Documented Code for datatool v3.4.4

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

2026-07-31

1 Introduction

This is the documented code for the datatool bundle. See datatool-user.pdf for the main user manual.

2 datatool-base.sty

This package provides all the basic commands needed by various packages in the datatool bundle. The datatool package was first released in 2007. The LaTeX kernel has changed significantly since then. I've started switching over to using LaTeX3, but it's still in a hybrid state. I have too many large packages and not enough time to do a complete rewrite.

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{datatool-base-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{datatool-base}[2026/07/31 v3.4.4 (NLCT)]
```

Required packages:

```
\RequirePackage{etoolbox}
\RequirePackage{amsmath}
\RequirePackage{xfor}
\RequirePackage{ifthen}
```

tracklang now loaded as from v3.0. Ideally it needs to be v1.6.3+ but doesn't matter so much if no localisation support requested.

```
\RequirePackage{tracklang}[2022/12/13]
```

Removed substr.sty.

```
\ExplSyntaxOn
```

Scratch variables. Integers:

```
\int_new:N \l__datatool_tmpa_int
\int_new:N \l__datatool_tmpb_int
\int_new:N \l__datatool_tmpc_int
\int_new:N \l__datatool_tmpd_int
```

Used to count or index items:

```
\int_new:N \l__datatool_count_int
```

Temporary storage of data type identifier:

```
\int_new:N \l__datatool_tmp_datatype_int
```

Token lists (the first one may have already been defined if datatool has been loaded):

```
\tl_clear_new:N \l__datatool_tmpa_tl
\tl_new:N \l__datatool_tmpb_tl
\tl_new:N \l__datatool_tmpc_tl
\tl_new:N \l__datatool_tmpd_tl
\tl_new:N \l__datatool_tmp_currency_tl
\tl_new:N \l__datatool_tmp_initial_tl
\tl_new:N \l__datatool_result_tl
\tl_new:N \l__datatool_resulta_tl
\tl_new:N \l__datatool_resultb_tl
\tl_new:N \l__datatool_dialect_tl
```

Floating point:

```
\fp_new:N \l__datatool_tmpa_fp
\fp_new:N \l__datatool_tmpb_fp
\fp_new:N \l__datatool_tmpc_fp
\fp_new:N \l__datatool_tmpd_fp
\fp_new:N \l__datatool_mean_fp
\fp_new:N \l__datatool_total_fp
\fp_new:N \l__datatool_datum_value_fp
```

Dimensions:

```
\dim_new:N \l__datatool_tmpa_dim
\dim_new:N \l__datatool_tmpb_dim
```

Strings:

```
\str_new:N \l__datatool_tmpa_str
\str_new:N \l__datatool_tmpb_str
\str_new:N \l__datatool_tmpc_str
```

Clist.

```
\clist_new:N \l__datatool_tmp_clist
```

Sequences.

```
\seq_new:N \l__datatool_tmp_seq
\seq_new:N \l__datatool_tmpa_seq
\seq_new:N \l__datatool_tmpb_seq
```

The following may also be used by the localisation files, so not a private variable, but still just a scratch variable.

```
\seq_new:N \l_datatool_regex_match_seq
\seq_new:N \l_datatool_timestamp_match_seq
```

Regular expressions

```
\regex_new:N \l__datatool_tmpa_regex
\regex_new:N \l__datatool_tmpb_regex
```

```
\ExplSyntaxOff
```

2.1 Package Options

Version 3.0 has switched to LaTeX3 key=value interface.

`\ifdtlverbose` Switch to govern verbose mode.

```
\newif\ifdtlverbose
```

`\if@dtl@utf8` Switch to determine UTF-8 setting. Deprecated.

```
\expandafter\newif\csname if@dtl@utf8\endcsname
```

`\TrackLangEncodingName` Check for `\TrackLangEncodingName`. This is used to input the appropriate encoding file, if available. If this command isn't available then it's likely that an old version of tracklang is installed.

```
\ExplSyntaxOn
\cs_if_exist:NF \TrackLangEncodingName
{
  \cs_if_exist:NTF \inputencodingname
  { \tl_set:NN \TrackLangEncodingName \inputencodingname }
  { \tl_set:Nn \TrackLangEncodingName { utf8 } }
}
```

Are we using a Unicode engine? (Affects regular expression matches with x`{`⟨*hex*⟩`}` for sort handler functions.)

```
\bool_lazy_any:nTF
 {
  { \sys_if_engine_luatex_p: }
  { \sys_if_engine_xetex_p: }
  { \sys_if_engine_uptex_p: }
 }
 {
   \cs_set_eq:NN \datatool_if_unicode_engine:TF \use_i:nn
   \cs_set_eq:NN \datatool_if_unicode_engine:T \use:n
   \cs_set_eq:NN \datatool_if_unicode_engine:F \use_none:n
 }
 {
   \cs_set_eq:NN \datatool_if_unicode_engine:TF \use_ii:nn
   \cs_set_eq:NN \datatool_if_unicode_engine:T \use_none:n
   \cs_set_eq:NN \datatool_if_unicode_engine:F \use:n
 }
```

`\@dtl@mathprocessor` As from v3.0, the default processor is lua if `\directlua` is defined or l3fp otherwise.

```
\ifdef\directlua
```

```
{\providecommand*{\@dtl@mathprocessor}{lua}}
{\providecommand*{\@dtl@mathprocessor}{l3fp}}
```

Determine whether or not to load locale support. This command will either expand to its argument or ignore it.

```
\newcommand\datatool@load@locales[1]{#1}
```

List of locales to track (in addition to any that might already be tracked):

```
\clist_new:N \l__datatool_extra_locales_clist
```

When to purify sort values:

```
\bool_new:N \l__datatool_initial_purify_early_bool
\bool_set_true:N \l__datatool_initial_purify_early_bool
```

List of types to reformat if auto-reformat settings are on:

```
\seq_new:N \l__datatool_auto_reformat_types_seq
\seq_set_from_clist:Nn \l__datatool_auto_reformat_types_seq
 { integer , decimal , si , currency , datetime , date , time }
```

Append to sequence:

```
\cs_new:Nn \__datatool_auto_reformat_types_do:n
 {
   \tl_if_eq:nnTF { #1 } { integer }
    {
      \seq_put_right:Nn \l__datatool_auto_reformat_types_seq
        { #1 }
    }
    {
      \tl_if_eq:nnTF { #1 } { decimal }
       {
         \seq_put_right:Nn \l__datatool_auto_reformat_types_seq
           { #1 }
       }
       {
         \tl_if_eq:nnTF { #1 } { si }
          {
            \seq_put_right:Nn \l__datatool_auto_reformat_types_seq
              { #1 }
          }
          {
            \tl_if_eq:nnTF { #1 } { datetime }
             {
               \seq_put_right:Nn \l__datatool_auto_reformat_types_seq
                 { #1 }
             }
             {
               \tl_if_eq:nnTF { #1 } { date }
                {
                  \seq_put_right:Nn \l__datatool_auto_reformat_types_seq
                    { #1 }
                }
```

```
                {
                  \tl_if_eq:nnTF { #1 } { time }
                   {
                     \seq_put_right:Nn \l__datatool_auto_reformat_types_seq
                       { #1 }
                   }
                   {
                     \PackageError { datatool-base }
                      {
                        Unsupported ~ auto-reformat ~ data ~ type ~
                         identifier  ~ ` #1 '
                      }
                      {
                        Supported ~ identifiers: ~
                        `integer', ~ `decimal', ~ `si', ~
                        `datetime', ~ `date', ~ `time'
                      }
                   }
                }
             }
          }
       }
    }
 }
\prg_new_conditional:Npnn \__datatool_if_auto_reformat_on:n #1
   { T, F, TF }
 {
   \seq_if_in:NnTF
      \l__datatool_auto_reformat_types_seq
     { #1 }
    {
       \tl_if_eq:nnTF { #1 } { si }
        {
           \bool_if:NTF \l__datatool_reformat_numeric_bool
            { \prg_return_true: }
            { \prg_return_false: }
        }
        {
          \int_if_exist:cTF { c_datatool_ #1 _int }
           {
             \exp_args:Nc \datatool_if_temporal_datum_type:nTF
               { c_datatool_ #1 _int }
              {
                \bool_if:NTF \l__datatool_reformat_datetime_bool
                 { \prg_return_true: }
                 { \prg_return_false: }
              }
              {
                \exp_args:Nc \datatool_if_numeric_datum_type:nTF
```

```
                { c_datatool_ #1 _int }
                {
                  \bool_if:NTF \l__datatool_reformat_numeric_bool
                   { \prg_return_true: }
                   { \prg_return_false: }
                }
                { \prg_return_false: }
              }
            }
            { \prg_return_false: }
         }
    }
    { \prg_return_false: }
 }
```

Similarly, but data type under consideration is in `\@dtl@datatype`:

```
\prg_new_conditional:Npnn \__datatool_if_auto_reformat_on:
   { T, F, TF }
 {
   \int_case:nnF { \@dtl@datatype }
    {
      { \c_datatool_integer_int }
       {
         \seq_if_in:NnTF
            \l__datatool_auto_reformat_types_seq
          { integer }
         {
           \bool_if:NTF \l__datatool_reformat_numeric_bool
            { \prg_return_true: }
            { \prg_return_false: }
         }
         { \prg_return_false: }
       }
      { \c_datatool_decimal_int }
       {
         \seq_if_in:NnTF
            \l__datatool_auto_reformat_types_seq
          { decimal }
         {
           \bool_if:NTF \l__datatool_reformat_numeric_bool
            { \prg_return_true: }
            { \prg_return_false: }
         }
         { \prg_return_false: }
       }
      { \c_datatool_currency_int }
       {
         \seq_if_in:NnTF
            \l__datatool_auto_reformat_types_seq
          { currency }
```

```
            {
              \bool_if:NTF \l__datatool_reformat_numeric_bool
               { \prg_return_true: }
               { \prg_return_false: }
            }
            { \prg_return_false: }
          }
         { \c_datatool_datetime_int }
          {
            \seq_if_in:NnTF
               \l__datatool_auto_reformat_types_seq
             { datetime }
            {
              \bool_if:NTF \l__datatool_reformat_datetime_bool
               { \prg_return_true: }
               { \prg_return_false: }
            }
            { \prg_return_false: }
          }
         { \c_datatool_date_int }
          {
            \seq_if_in:NnTF
               \l__datatool_auto_reformat_types_seq
             { date }
            {
              \bool_if:NTF \l__datatool_reformat_datetime_bool
               { \prg_return_true: }
               { \prg_return_false: }
            }
            { \prg_return_false: }
          }
         { \c_datatool_time_int }
          {
            \seq_if_in:NnTF
               \l__datatool_auto_reformat_types_seq
             { time }
            {
              \bool_if:NTF \l__datatool_reformat_datetime_bool
               { \prg_return_true: }
               { \prg_return_false: }
            }
            { \prg_return_false: }
          }
      }
      { \prg_return_false: }
  }
```

Option definitions. Note that all packages in the datatool bundle now all use the same option group.

```
\keys_define:nn { datatool }
```

```
{
   verbose .legacy_if_set:n = dtlverbose,
   utf8 .legacy_if_set:n = @dtl@utf8,
   math .choice: ,
   math / fp .code:n =
    { \tl_set:Nn \@dtl@mathprocessor { fp } } ,
   math / pgfmath .code:n =
    { \tl_set:Nn \@dtl@mathprocessor { pgfmath } } ,
   math / l3fp .code:n =
    { \tl_set:Nn \@dtl@mathprocessor { l3fp } } ,
   math / lua .code:n =
    { \tl_set:Nn \@dtl@mathprocessor { lua } } ,
   math .usage:n = load,
   lang-warn .choice: ,
   lang-warn / true .code:n =
    {
      \cs_set_eq:NN \datatool_locale_warn:nn \PackageWarning
    } ,
   lang-warn / false .code:n =
    {
      \PackageInfo
        {datatool-base}
        {
         localisation ~ warnings ~ switched ~
         off ~ (including ~ tracklang ~ warnings)
        }
      \cs_set_eq:NN  \datatool_locale_warn:nn \use_none:nn
      \TrackLangShowWarningsfalse
    } ,
   lang-warn .default:n = true ,
   lang-warn .initial:n = true ,
   lang-warn .usage:n = load ,
   nolocale .code:n =
    {
      \cs_set_eq:NN \datatool@load@locales \use_none:n
      \clist_clear:N \l__datatool_extra_locales_clist
    },
   nolocale .value_forbidden:n = true,
   nolocale .usage:n = load,
   locales .code:n =
     {
        \cs_set_eq:NN \datatool@load@locales \use:n
        \clist_set:Nn \l__datatool_extra_locales_clist { #1 }
     },
   locales .usage:n = load,
```

Synonym:

```
   lang .code:n =
     {
        \cs_set_eq:NN \datatool@load@locales \use:n
```

```
          \clist_set:Nn \l__datatool_extra_locales_clist { #1 }
        },
      lang .usage:n = load,
      lists .code:n = { \keys_set:nn { datatool / lists } { #1 } },
      initial-purify .choice:,
      initial-purify / early .code:n =
        { \bool_set_true:N \l__datatool_initial_purify_early_bool },
      initial-purify / late .code:n =
        { \bool_set_false:N \l__datatool_initial_purify_early_bool },
      auto-reformat-types .code:n =
        {
          \seq_clear:N \l__datatool_auto_reformat_types_seq
          \exp_args:Ne \clist_map_function:nN { #1 }
            \__datatool_auto_reformat_types_do:n
        } ,
      compare .code:n = { \keys_set:nn { datatool / compare } { #1 } },
      datetime .code:n = { \keys_set:nn { datatool / datetime } { #1 } },
      numeric .code:n = { \keys_set:nn { datatool / numeric } { #1 } },
    }
```

Numeric options (new to v3.0).

Should numeric values be reformatted after parsing?

```
\bool_new:N \l__datatool_reformat_numeric_bool
\bool_set_false:N \l__datatool_reformat_numeric_bool
```

`\DTLscinum` Used with auto-reformat for scientific notation.

```
\IfPackageLoadedTF { siunitx }
 {
   \newcommand \DTLscinum [ 1 ] { \num { #1 } }
 }
 {
   \newcommand \DTLscinum [ 1 ] { #1 }
   \AddToHook
     { package / siunitx / after }
     {
       \renewcommand \DTLscinum [ 1 ] { \num { #1 } }
     }
 }
```

Should region files override the default number chars?

```
\bool_new:N \l_datatool_region_set_numberchars_bool
\bool_set_true:N \l_datatool_region_set_numberchars_bool
```

Should region files override the default currency?

```
\bool_new:N \l_datatool_region_set_currency_bool
\bool_set_true:N \l_datatool_region_set_currency_bool
```

lcurrencysymbolprefixfmt

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

\DTLcurrCodeOrSymOrChar

```
\newcommand{\DTLcurrCodeOrSymOrChar}[3]{#2}

\keys_define:nn { datatool / numeric }
{
   auto-reformat .bool_set:N = \l__datatool_reformat_numeric_bool ,
```

This will also switch off any region number chars. Value needs to be in the form {⟨*grp-char*⟩}{⟨*decimal-char*⟩}

```
    set-number-chars .code:n =
     {
       \DTLsetnumberchars #1
       \bool_set_false:N \l_datatool_region_set_numberchars_bool
     } ,
```

Should the region set the number group and decimal characters:

```
    region-number-chars .choice: ,
    region-number-chars / true .code:n =
     {
       \bool_set_true:N \l_datatool_region_set_numberchars_bool
       \tl_if_empty:NF \l_datatool_current_region_tl
        {
          \cs_if_exist_use:cF
           { datatool \l_datatool_current_region_tl SetNumberChars }
           {
             \PackageWarning { datatool-base }
              {
                No ~ numberchars ~ hook ~ found ~ for ~ region ~
                  ` \l_datatool_current_region_tl '
              }
           }
        }
     } ,
    region-number-chars / false .code:n =
     {
       \bool_set_false:N \l_datatool_region_set_numberchars_bool
     } ,
    region-number-chars .default:n = true ,
```

Currency must already have been defined. This will also switch off any region currency.

```
    set-currency .code:n =
     {
       \tl_if_exist:cTF { dtl@curr@ \tl_to_str:n { #1 } @sym }
        {
          \tl_set:Ne \@dtl@currency { \exp_not:c { DTLcurr #1 } }
          \tl_set:Ne \DTLCurrencyCode { #1 }
          \tl_set:Ne \DTLfmtcurrency
            { \exp_not:c { dtl@curr@ #1 @fmt } }
          \bool_false:N \l_datatool_region_set_currency_bool
        }
        {
```

```
      \PackageError { datatool-base }
       {
         Currency ~ ` #1 ' ~ has ~ not ~ been ~ defined
       }
       {
         Currency ~ must ~ first ~ be ~ defined ~ with ~
         \tl_to_str:N \DTLdefcurrency
       }
    }
 } ,
region-currency .choice: ,
region-currency / true .code:n =
 {
   \bool_set_true:N \l_datatool_region_set_currency_bool
   \tl_if_empty:NF \l_datatool_current_region_tl
    {
      \cs_if_exist_use:cF
       { datatool \l_datatool_current_region_tl SetCurrency }
       {
         \PackageWarning { datatool-base }
          {
            No ~ currency ~ hook ~ found ~ for ~ region ~
              ` \l_datatool_current_region_tl '
          }
       }
    }
 } ,
region-currency / false .code:n =
 {
   \bool_set_false:N \l_datatool_region_set_currency_bool
 } ,
region-currency .default:n = true ,
```

Convenient method of redefining formatting command used by region files, but will only have an effect if the region uses it.

```
region-currency-prefix .choice: ,
region-currency-prefix / normal .code:n =
 {
   \cs_set_eq:NN \datatoolcurrencysymbolprefixfmt \use:n
 } ,
region-currency-prefix / smallcaps .code:n =
 {
   \renewcommand \datatoolcurrencysymbolprefixfmt [ 1 ]
    {
      \exp_args:Ne \textsc { \text_lowercase:n { ##1 } }
    }
 } ,
region-currency-prefix / smaller .code:n =
 {
   \renewcommand \datatoolcurrencysymbolprefixfmt [ 1 ]
```

```
        {
          \textsmaller { ##1 }
        }
     } ,
```

Redefine `\DTLcurrCodeOrSymOrChar`:

```
    currency-symbol-style .choice:,
    currency-symbol-style / iso .code:n =
     {
       \cs_set_eq:NN \DTLcurrCodeOrSymOrChar \use_i:nnn
     },
    currency-symbol-style / symbol .code:n =
     {
       \cs_set_eq:NN \DTLcurrCodeOrSymOrChar \use_ii:nnn
     },
    currency-symbol-style / string .code:n =
     {
       \cs_set_eq:NN \DTLcurrCodeOrSymOrChar \use_iii:nnn
     },
}
```

Date/time options (new to v3.0).

Should date/time values be parsed?

```
\bool_new:N \l__datatool_parse_datetime_bool
\bool_set_false:N \l__datatool_parse_datetime_bool
```

The next two conditionals are only relevant if the above is true. Should date/time values be parsed for ISO format?

```
\bool_new:N \l__datatool_parse_datetime_iso_bool
\bool_set_true:N \l__datatool_parse_datetime_iso_bool
```

Should date/time values be parsed for regional format? This will require the appropriate regional support.

```
\bool_new:N \l__datatool_parse_datetime_regional_bool
\bool_set_true:N \l__datatool_parse_datetime_regional_bool
```

Should parsed date/time values be reformatted?

```
\bool_new:N \l__datatool_reformat_datetime_bool
\bool_set_false:N \l__datatool_reformat_datetime_bool
```

LCurrentLocaleFormatDate

`\DTLCurrentLocaleFormatDate{`⟨*YYYY*⟩`}{`⟨*MM*⟩`}{`⟨*DD*⟩`}{`⟨*DOW*⟩`}`

Localisation support should redefine this.

```
\newcommand \DTLCurrentLocaleFormatDate [4]
 {
   \datatool_default_date_fmt:nnnn { #1 } { #2 } { #3 } { #4 }
 }

\cs_new_nopar:Nn \datatool_default_date_fmt:nnnn
 {
```

```
  \cs_if_exist:NTF \DTMdisplaydate
   {
     \tl_if_empty:nTF { #4 }
      {
        \DTMdisplaydate { #1 } { #2 } { #3 } { -1 }
      }
      {
        \DTMdisplaydate { #1 } { #2 } { #3 } { #4 }
      }
   }
   {
     \datatool_date_iso_fmt:nnnn { #1 } { #2 } { #3 } { #4 }
   }
}
```

\DataToolDateFmt

\DataToolDateFmt{⟨*YYYY*⟩}{⟨*MM*⟩}{⟨*DD*⟩}{⟨*DOW*⟩}

```
\newcommand \DataToolDateFmt [4]
{
  \DTLCurrentLocaleFormatDate { #1 } { #2 } { #3 } { #4 }
}
```

The following has a fourth argument for consistency.

```
\cs_new_nopar:Nn \datatool_date_iso_fmt:nnnn
{
  \datatool_date_iso_fmt:nnn { #1 } { #2 } { #3 }
}
\cs_new_nopar:Nn \datatool_date_iso_fmt:nnn
{
  \int_eval:n { #1 }
   - \datatool_two_digits:n { #2 }
   - \datatool_two_digits:n { #3 }
}
```

LCurrentLocaleFormatTime

\DTLCurrentLocaleFormatDate{⟨*hh*⟩}{⟨*mm*⟩}{⟨*ss*⟩}

Localisation support should redefine this.

```
\newcommand \DTLCurrentLocaleFormatTime [3]
{
  \datatool_default_time_fmt:nnn { #1 } { #2 } { #3 }
}

\cs_new_nopar:Nn \datatool_default_time_fmt:nnn
{
  \cs_if_exist:NTF \DTMdisplaytime
   {
     \tl_if_empty:nTF { #3 }
```

```
      {
        \DTMdisplaytime { #1 } { #2 } { 0 }
      }
      {
        \DTMdisplaytime { #1 } { #2 } { #3 }
      }
   }
   {
     \datatool_time_iso_fmt:nnn { #1 } { #2 } { #3 }
   }
}
```

\DataToolTimeFmt{⟨*hh*⟩}{⟨*mm*⟩}{⟨*ss*⟩}

\DataToolTimeFmt

```
\newcommand \DataToolTimeFmt [3]
 {
   \DTLCurrentLocaleFormatTime { #1 } { #2 } { #3 }
 }

\cs_new_nopar:Nn \datatool_time_iso_fmt:nnn
 {
   \datatool_two_digits:n { #1 }
   \c_colon_str
   \datatool_two_digits:n { #2 }
   \tl_if_empty:nF { #3 }
    {
      \c_colon_str
      \datatool_two_digits:n { #3 }
    }
 }
```

\DTLCurrentLocaleFormatTimeZone{⟨*TZh*⟩}{⟨*TZm*⟩}

rentLocaleFormatTimeZone

```
\newcommand \DTLCurrentLocaleFormatTimeZone [2]
 {
   \datatool_default_timezone_fmt:nn { #1 } { #2 }
 }

\cs_new:Nn \datatool_default_timezone_fmt:nn
 {
   \cs_if_exist:NTF \DTMdisplayzone
    {
      \DTMdisplayzone { #1 } { #2 }
    }
    {
      \datatool_timezone_iso_fmt:nn { #1 } { #2 }
    }
 }
```

\DataToolTimeZoneFmt

\DataToolTimeZoneFmt{⟨*TZh*⟩}{⟨*TZm*⟩}

```
\newcommand \DataToolTimeZoneFmt [2]
 {
   \DTLCurrentLocaleFormatTimeZone { #1 } { #2 }
 }

\cs_new_nopar:Nn \datatool_timezone_iso_fmt:nn
 {
   \datatool_signed_two_digits:n { #1 }
   \c_colon_str
   \datatool_two_digits:n { #2 }
 }
```

entLocaleTimeStampFmtSep Localisation files may redefine this.

```
\newcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
```

\DataToolTimeStampFmtSep Separator used in `\datatool_default_timestamp_fmt:nnnnnnn` between the date and time. Defaults to a space.

```
\newcommand \DataToolTimeStampFmtSep
 {
   \DTLCurrentLocaleTimeStampFmtSep
 }
```

\DataToolDateTimeFmt

\DataToolDateTimeFmt{⟨*date-args*⟩}{⟨*time-args*⟩}
{⟨*time-zone-args*⟩}

The ⟨*date-args*⟩ argument may either be empty or in the form {⟨*YYYY*⟩}{⟨*MM*⟩}{⟨*DD*⟩}{⟨*DOW*⟩} appropriate for `\DataToolDateFmt`. The ⟨*time-args*⟩ argument may either be empty or in the form {⟨*hh*⟩}{⟨*mm*⟩}{⟨*ss*⟩} appropriate for `\DataToolTimeFmt`. The ⟨*time-zone-args*⟩ argument may either be empty or in the form {⟨*TZh*⟩}{⟨*TZm*⟩} appropriate for `\DataToolTimeZoneFmt`.

```
\newcommand* \DataToolDateTimeFmt [3]
 {
   \tl_if_empty:nTF { #1 }
    {
```

No date arguments.

```
     \tl_if_empty:nF { #2 }
      {
```

Time arguments provided. (There should be three.)

```
        \DataToolTimeFmt #2
      }
     \tl_if_empty:nF { #3 }
      {
```

```
Time zone arguments provided. (There should be two.)
          \DataToolTimeZoneFmt #3
        }
      }
      {
Date arguments provided. (There should be four.)
        \tl_if_empty:nTF { #2 }
        {
No time arguments provided.
          \DataToolDateFmt #1
        }
        {
Time arguments provided. (There should be three.)
         \tl_if_empty:nTF { #3 }
          {
No time zone arguments provided.
            \DataToolTimeStampNoZoneFmt #1 #2
          }
          {
Time zone arguments provided. (There should be two.)
            \DataToolTimeStampWithZoneFmt #1 #2 #3
          }
        }
      }
  }
```

\DataToolTimeStampWithZoneFmt{⟨YYYY⟩}{⟨MM⟩}{⟨DD⟩}
{⟨DOW⟩}
{⟨hh⟩}{⟨mm⟩}{⟨ss⟩}

ToolTimeStampWithZoneFmt

```
\newcommand \DataToolTimeStampWithZoneFmt [9]
 {
   \DTLCurrentLocaleFormatTimeStampWithZone
     { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } { #7 } { #8 } { #9 }
 }
```

\DataToolTimeStampNoZoneFmt{⟨YYYY⟩}{⟨MM⟩}{⟨DD⟩}{⟨DOW⟩}
{⟨hh⟩}{⟨mm⟩}{⟨ss⟩}

taToolTimeStampNoZoneFmt

```
\newcommand \DataToolTimeStampNoZoneFmt [7]
 {
```

```
    \DTLCurrentLocaleFormatTimeStampNoZone
      { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } { #7 }
  }

\cs_new_nopar:Nn \datatool_default_timestamp_fmt:nnnnnnn
 {
   \cs_if_exist:NTF \DTMdisplay
    {
```

The datetime2 style will need to have the zone suppressed with showzone=false (if supported by the current style).

```
       \tl_if_empty:nTF { #4 }
        {
          \DTMdisplay
            { #1 } { #2 } { #3 } { -1 }
            { #5 } { #6 } { #7 }
            { 0 } { 0 }
        }
        {
          \DTMdisplay
            { #1 } { #2 } { #3 } { #4 }
            { #5 } { #6 } { #7 }
            { 0 } { 0 }
        }
    }
    {
      \datatool_default_date_fmt:nnnn
        { #1 } { #2 } { #3 } { #4 }
      \DataToolTimeStampFmtSep
      \datatool_default_time_fmt:nnn
        { #5 } { #6 } { #7 }
    }
 }

\cs_new_nopar:Nn \datatool_default_timestamp_fmt:nnnnnnnnn
 {
   \cs_if_exist:NTF \DTMdisplay
    {
      \tl_if_empty:nTF { #4 }
       {
         \DTMdisplay
           { #1 } { #2 } { #3 } { -1 }
           { #5 } { #6 } { #7 }
           { #8 } { #9 }
       }
       {
         \DTMdisplay
           { #1 } { #2 } { #3 } { #4 }
           { #5 } { #6 } { #7 }
           { #8 } { #9 }
       }
```

```
    }
    {
      \datatool_default_timestamp_fmt:nnnnnnn
        { #1 } { #2 } { #3 } { #4 }
        { #5 } { #6 } { #7 }
      \datatool_default_timezone_fmt:nn
        { #8 } { #9 }
    }
 }
\cs_new_nopar:Nn \datatool_timestamp_iso_fmt:nnnnnnn
 {
   \datatool_date_iso_fmt:nnnn { #1 } { #2 } { #3 } { #4 }
   T
   \datatool_time_iso_fmt:nnn { #5 } { #6 } { #7 }
 }
\cs_new_nopar:Nn \datatool_timestamp_iso_fmt:nnnnnnnnn
 {
   \datatool_timestamp_iso_fmt:nnnnnnn
    { #1 } { #2 } { #3 } { #4 }
    { #5 } { #6 } { #7 }
   \datatool_timezone_iso_fmt:nn { #8 } { #9 }
 }
```

eFormatTimeStampWithZone

\DTLCurrentLocaleFormatTimeStampWithZone{⟨*YYYY*⟩}
{⟨*MM*⟩}{⟨*DD*⟩}{⟨*DOW*⟩}
{⟨*hh*⟩}{⟨*mm*⟩}{⟨*ss*⟩}{⟨*tzh*⟩}{⟨*tzm*⟩}

Localisation support should redefine this.

```
\newcommand \DTLCurrentLocaleFormatTimeStampWithZone [9]
 {
   \datatool_default_timestamp_fmt:nnnnnnnnn
    { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } { #7 } { #8 } { #9 }
 }
```

aleFormatTimeStampNoZone

\DTLCurrentLocaleFormatTimeStampNoZone{⟨*YYYY*⟩}{⟨*MM*⟩}
{⟨*DD*⟩}{⟨*DOW*⟩}
{⟨*hh*⟩}{⟨*mm*⟩}{⟨*ss*⟩}

Localisation support should redefine this.

```
\newcommand \DTLCurrentLocaleFormatTimeStampNoZone [7]
 {
   \datatool_default_timestamp_fmt:nnnnnnn
      { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } { #7 }
 }
```

```
\keys_define:nn { datatool / datetime }
{
   parse .choice:,
   parse / false .code:n =
    {
      \bool_set_false:N \l__datatool_parse_datetime_bool
    } ,
   parse / true .code:n =
    {
      \bool_set_true:N \l__datatool_parse_datetime_bool
    } ,
   parse / parse-only .code:n =
    {
      \bool_set_true:N \l__datatool_parse_datetime_bool
      \bool_set_false:N \l__datatool_reformat_datetime_bool
    } ,
   parse / auto-reformat .code:n =
    {
      \bool_set_true:N \l__datatool_parse_datetime_bool
      \bool_set_true:N \l__datatool_reformat_datetime_bool
    } ,
   parse / iso-only .code:n =
    {
      \bool_set_true:N \l__datatool_parse_datetime_bool
      \bool_set_true:N \l__datatool_parse_datetime_iso_bool
      \bool_set_false:N \l__datatool_parse_datetime_regional_bool
    } ,
   parse / region-only .code:n =
    {
      \bool_set_true:N \l__datatool_parse_datetime_bool
      \bool_set_false:N \l__datatool_parse_datetime_iso_bool
      \bool_set_true:N \l__datatool_parse_datetime_regional_bool
    } ,
   parse / iso + region .code:n =
    {
      \bool_set_true:N \l__datatool_parse_datetime_bool
      \bool_set_true:N \l__datatool_parse_datetime_iso_bool
      \bool_set_true:N \l__datatool_parse_datetime_regional_bool
    } ,
   parse .default:n = true ,
   auto-reformat .choice: ,
   auto-reformat / false .code:n =
    {
      \bool_set_false:N \l__datatool_reformat_datetime_bool
    } ,
   auto-reformat / true .code:n =
    {
      \bool_set_true:N \l__datatool_reformat_datetime_bool
    } ,
   auto-reformat / region .code:n =
```

```
 {
   \bool_set_true:N \l__datatool_reformat_datetime_bool
   \renewcommand \DataToolDateFmt [ 4 ]
    {
      \DTLCurrentLocaleFormatDate { ##1 } { ##2 } { ##3 } { ##4 }
    }
   \renewcommand \DataToolTimeFmt [ 3 ]
    {
      \DTLCurrentLocaleFormatTime { ##1 } { ##2 } { ##3 }
    }
   \renewcommand \DataToolTimeZoneFmt [ 2 ]
    {
      \DTLCurrentLocaleFormatTimeZone { ##1 } { ##2 }
    }
   \renewcommand \DataToolTimeStampWithZoneFmt [ 9 ]
    {
      \DTLCurrentLocaleFormatTimeStampWithZone
        { ##1 } { ##2 } { ##3 } { ##4 }
        { ##5 } { ##6 } { ##7 }
        { ##8 } { ##9 }
    }
   \renewcommand \DataToolTimeStampNoZoneFmt [ 7 ]
    {
      \DTLCurrentLocaleFormatTimeStampNoZone
        { ##1 } { ##2 } { ##3 } { ##4 }
        { ##5 } { ##6 } { ##7 }
    }
   \renewcommand \DataToolTimeStampFmtSep
    {
     \DTLCurrentLocaleTimeStampFmtSep
    }
 } ,
auto-reformat / iso .code:n =
 {
   \bool_set_true:N \l__datatool_reformat_datetime_bool
   \renewcommand \DataToolDateFmt [ 4 ]
    {
      \datatool_date_iso_fmt:nnnn
        { ##1 } { ##2 } { ##3 } { ##4 }
    }
   \renewcommand \DataToolTimeFmt [ 3 ]
    {
      \datatool_time_iso_fmt:nnn
        { ##1 } { ##2 } { ##3 }
    }
   \renewcommand \DataToolTimeZoneFmt [ 2 ]
    {
      \datatool_timezone_iso_fmt:nn
        { ##1 } { ##2 }
    }
```

```
   \renewcommand \DataToolTimeStampWithZoneFmt [ 9 ]
    {
      \datatool_timestamp_iso_fmt:nnnnnnnnn
        { ##1 } { ##2 } { ##3 } { ##4 }
        { ##5 } { ##6 } { ##7 }
        { ##8 } { ##9 }
    }
   \renewcommand \DataToolTimeStampNoZoneFmt [ 7 ]
    {
      \datatool_timestamp_iso_fmt:nnnnnnn
        { ##1 } { ##2 } { ##3 } { ##4 }
        { ##5 } { ##6 } { ##7 }
    }
 } ,
auto-reformat / datetime2 .code:n =
 {
   \cs_if_exist:NTF \DTLdisplay
    {
      \bool_set_true:N \l__datatool_reformat_datetime_bool
      \renewcommand \DataToolDateFmt [ 4 ]
       {
         \tl_if_empty:nTF { ##4 }
          {
            \DTMdisplaydate { ##1 } { ##2 } { ##3 } { -1 }
          }
          {
            \DTMdisplaydate { ##1 } { ##2 } { ##3 } { ##4 }
          }
       }
      \renewcommand \DataToolTimeFmt [ 3 ]
       {
         \tl_if_empty:nTF { ##3 }
          {
            \DTMdisplaytime { ##1 } { ##2 } { 0 }
          }
          {
            \DTMdisplaytime { ##1 } { ##2 } { ##3 }
          }
       }
      \renewcommand \DataToolTimeZoneFmt [ 2 ]
       {
         \DTMdisplayzone { ##1 } { ##2 }
       }
      \renewcommand \DataToolTimeStampWithZoneFmt [ 9 ]
       {
         \tl_if_empty:nTF { ##4 }
          {
            \DTMdisplay
              { ##1 } { ##2 } { ##3 } {-1 }
              { ##5 } { ##6 } { ##7 }
```

```
                { ##8 } { ##9 }
            }
            {
              \DTMdisplay
                { ##1 } { ##2 } { ##3 } { ##4 }
                { ##5 } { ##6 } { ##7 }
                { ##8 } { ##9 }
            }
         }
        \renewcommand \DataToolTimeStampNoZoneFmt [ 7 ]
         {
           \tl_if_empty:nTF { ##4 }
            {
              \DTMdisplay
                { ##1 } { ##2 } { ##3 } {-1 }
                { ##5 } { ##6 } { ##7 }
                { 0 } { 0 }
            }
            {
              \DTMdisplay
                { ##1 } { ##2 } { ##3 } { ##4 }
                { ##5 } { ##6 } { ##7 }
                { 0 } { 0 }
            }
         }
      }
      {
         \PackageError { datatool-base }
          {
             Option ~ datetime ~ = ~ { ~ auto-reformat ~ = ~ datetime2 ~ } ~
             requires ~ datetime2 ~ package
          }
          {
            You ~ need ~ to ~ load ~ datetime2.sty ~ before ~ you ~
            set ~ this ~ option
          }
      }
   } ,
  auto-reformat .default:n = false ,
}
```

List options (new to v3.0).

If true, trim leading and trailing space around elements in the list-related commands below. Note the default changes the behaviour as from v3.0.

```
\bool_new:N \l__datatool_list_trim_bool
```

Reverse sort CSV lists:

```
\bool_new:N \l__datatool_sort_reverse_bool
```

Preprocess to create datum elements when sorting CSV lists:

```
\bool_new:N \l__datatool_sort_datum_bool
```

`\DTLlistand` Use either `\DTLandname` or `\&` in lists. (No difference if no language support provided.)

```
\bool_new:N \l_datatool_list_and_bool
\bool_set_true:N \l_datatool_list_and_bool
\newcommand \DTLlistand
 {
   \bool_if:NTF \l_datatool_list_and_bool
    { \DTLandname }
    { \& }
 }
```

`\ifDTLlistskipempty` If true, skip empty elements in the list-related commands below.

```
\newif\ifDTLlistskipempty
\DTLlistskipemptytrue

\keys_define:nn { datatool / lists }
{
   skip-empty .legacy_if_set:n = DTLlistskipempty ,
   trim .bool_set:N = \l__datatool_list_trim_bool ,
   trim .initial:n = true ,
   sort-reverse .bool_set:N = \l__datatool_sort_reverse_bool ,
   sort-reverse .initial:n = false ,
   sort-datum .bool_set:N = \l__datatool_sort_datum_bool ,
   sort-datum .initial:n = false ,
   and .choice: ,
   and / word .code:n =
    {
      \bool_true:N \l_datatool_list_and_bool
    } ,
   and / symbol .code:n =
    {
      \bool_false:N \l_datatool_list_and_bool
    } ,
}
```

String comparison options.

`\ifdtlcompareskipcs` If true, `\dtlcompare` should skip control sequences.

```
\newif\ifdtlcompareskipcs
\dtlcompareskipcsfalse
```

If the following boolean is true, `\dtlcompare` should expand control sequences.

```
\bool_new:N \l__datatool_compare_expand_cs_bool

\keys_define:nn { datatool / compare }
{
   expand-cs .bool_set:N = \l__datatool_compare_expand_cs_bool,
   skip-cs .legacy_if_set:n = dtlcompareskipcs,
}
```

`\DTLsetup` Provide user interface command to set options later.

```
\DeclareDocumentCommand \DTLsetup { m }
 { \keys_set:nn { datatool } { #1 } }
```

Use `\DTLsetLocaleOptions` for options provided by localisation files.

```
\ExplSyntaxOff
```

Process options:

```
\ProcessKeyOptions[datatool]
```

Load file dealing with numerical processes. As from version 3.0, these are in def files not packages.

```
\InputIfFileExists
 {datatool-\@dtl@mathprocessor.def}
 {}
 {
   \InputIfFileExists
    {datatool-l3fp.def}
    {
       \PackageError{datatool}
       {%
         Missing file `datatool-\@dtl@mathprocessor.def'
         for math=\@dtl@mathprocessor. Falling back on
         math=l3fp
       }
       {%
         Check that your TeX distribution contains the file
         `datatool-\@dtl@mathprocessor.def'
       }
    }
    {
       \PackageError{datatool}
       {%
         Missing file `datatool-\@dtl@mathprocessor.def'
         for math=\@dtl@mathprocessor. No math commands
         available!
       }
       {%
         Something is wrong with the datatool installation
       }
    }
 }
```

2.2 Utilities

`\dtl@message` `\dtl@message{`⟨*message string*⟩`}`

Displays message only if the verbose option is set.

```
\newcommand*{\dtl@message}[1]{%
  \ifdtlverbose\typeout{#1}\fi
}
```

`\@dtl@toks`

```
\newtoks\@dtl@toks
```

`\@dtl@tmpcount` Define temporary count register

```
\newcount\@dtl@tmpcount
```

`\dtl@tmplength` Define temporary length register. TODO: rename `\l__datatool_tmp_dim`?

```
\newlength\dtl@tmplength
```

Provide a way of measuring the height of text with problematic commands locally disabled:

```
\ExplSyntaxOn
\cs_new:Nn \datatool_measure_height:Nn
{
```

NB `\settoheight` automatically scopes the content.

```
  \settoheight #1 { \l_datatool_measure_hook_tl #2 }
}
\cs_new:Nn \datatool_measure_width:Nn
{
  \settowidth #1 { \l_datatool_measure_hook_tl #2 }
}
\cs_new:Nn \datatool_measure_depth:Nn
{
  \settodepth #1 { \l_datatool_measure_hook_tl #2 }
}
```

`\datatool_measure:NNNn` ⟨*wd-dim*⟩ ⟨*ht-dim*⟩ ⟨*dp-dim*⟩ {⟨*text*⟩}

Measure all dimensions.

```
\cs_new:Nn \datatool_measure:NNNn
 {
   \hbox_set:Nn \l__datatool_measure_box
    { \l_datatool_measure_hook_tl #4 }
   \dim_set:Nn #1 { \box_wd:N \l__datatool_measure_box }
   \dim_set:Nn #2 {\box_ht:N \l__datatool_measure_box }
   \dim_set:Nn #3 { \box_dp:N \l__datatool_measure_box }
 }
```

`\datatool_measure_ht_plus_dp:Nn` ⟨*dim*⟩ {⟨*text*⟩}

Measure the height plus depth.

```
\cs_new:Nn \datatool_measure_ht_plus_dp:Nn
 {
   \hbox_set:Nn \l__datatool_measure_box
    { \l_datatool_measure_hook_tl #2 }
   \dim_set:Nn #1
    {
      \box_ht:N \l__datatool_measure_box
      + \box_dp:N \l__datatool_measure_box
    }
 }
\box_new:N \l__datatool_measure_box
```

Hook:

```
\tl_new:N \l_datatool_measure_hook_tl
\tl_set:Nn \l_datatool_measure_hook_tl
{
  \cs_set_eq:NN \label \use_none:n
  \cs_set_eq:NN \ref \use_none:n
  \cs_set_eq:NN \pageref \use_none:n
  \cs_set_eq:NN \refstepcounter \__datatool_local_stepcounter:n
  \cs_set_eq:NN \stepcounter \__datatool_local_stepcounter:n
  \cs_set_eq:NN \hypertarget \use_ii:nn
  \cs_set_eq:NN \hyperlink \use_ii:nn
}
```

Local step counter for use in the above. This doesn't trigger an error if the counter is undefined nor does it reset dependent counter.

```
\cs_new:Nn \__datatool_local_stepcounter:n
{
  \int_if_exist:cT { c@ #1 } { \int_incr:c { c@ #1 } }
}
```

Swap values in integer variables:

```
\cs_new:Nn \datatool_swap_ints:NN
{
  \exp_args:Nee \__datatool_swap_ints:nnNN
    { \int_use:N #1 } { \int_use:N #2 } #1 #2
}
\cs_new:Nn \__datatool_swap_ints:nnNN
{
  \int_set:Nn #3 { #2 }
  \int_set:Nn #4 { #1 }
}
```

\dtlifintopenbetween

\dtlifintopenbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

If the values may be decimal, use \dtlifnumopenbetween instead.

Version 3.0 rewritten to use LaTeX3 commands. This means it can now fully expand.

```
\newcommand{\dtlifintopenbetween}[5]{%
  \bool_lazy_and:nnTF
    { \int_compare_p:nNn { #1 } > { #2 } }
    { \int_compare_p:nNn { #1 } < { #3 } }
  { #4 } { #5 }
}
```

\dtlifintclosedbetween

\dtlifintclosedbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

If the values may be decimal, use \dtlifnumclosedbetween instead.

Version 3.0 rewritten to use LaTeX3 commands. This means it can now fully expand. It's simpler to perform the reverse test (if outside the range).

```
\newcommand{\dtlifintclosedbetween}[5]{%
  \bool_lazy_or:nnTF
    { \int_compare_p:nNn { #1 } < { #2 } }
    { \int_compare_p:nNn { #1 } > { #3 } }
  { #5 } { #4 }
}
```

This is mainly for formatting time zone hours with a required leading sign:

```
\cs_new:Nn \datatool_signed_two_digits:n
 {
   \int_compare:nNnTF { #1 } < { \c_zero_int }
    {
      - \datatool_two_digits:e { \int_abs:n { #1 } }
    }
    {
      + \datatool_two_digits:n { #1 }
    }
 }
```

Expanding first reduces computations if the argument is an expression rather than just a number.

```
\cs_generate_variant:Nn \datatool_signed_two_digits:n { e, V }
```

Argument an integer variable:

```
\cs_new:Nn \datatool_signed_two_digits:N
 {
   \int_compare:nNnTF { #1 } < { \c_zero_int }
    {
      - \datatool_two_digits:e { \int_abs:n { #1 } }
    }
    {
      + \datatool_two_digits:N #1
    }
 }
```

Similar but omit sign for positive numbers. Note that `\two@digits` can discard a following space since it uses `\number`. This is avoided here by using `\int_eval:n`.

```
\cs_new:Nn \datatool_two_digits:n
 {
   \int_compare:nNnTF { #1 } < { \c_zero_int }
    {
      - \datatool_two_digits:e { \int_abs:n { #1 } }
    }
    {
      \int_compare:nNnT { #1 } < { 10 } { 0 }
      \int_eval:n { #1 }
    }
 }
\cs_generate_variant:Nn \datatool_two_digits:n { e, V }
```

Argument an integer variable:

```
\cs_new:Nn \datatool_two_digits:N
 {
   \int_compare:nNnTF { #1 } < { \c_zero_int }
    {
      - \datatool_two_digits:e { \int_abs:n { #1 } }
    }
    {
      \int_compare:nNnT { #1 } < { 10 } { 0 }
      \int_use:N #1
    }
 }
```

`\dtlpadleadingzeros`

`\dtlpadleadingzeros{`⟨*num-digits*⟩`}{`⟨*value*⟩`}`

Pad leading zeros. This is designed for sorting numbers by character code (and so needs to be expandable). Zero-padding helps place them in numeric order. The ⟨*num-digits*⟩ argument should be between 1 and 6.

```
\newcommand{\dtlpadleadingzeros}[2]{
  \fp_compare:nNnTF { #2 } < { \c_zero_fp }
  { \dtlpadleadingzerosminus } { \dtlpadleadingzerosplus }
  \__datatool_int_leading_zeros:ee
    { \int_eval:n { #1 } }
    { \fp_to_int:n { floor ( abs ( #2 ) ) } }
  \fp_to_decimal:n { abs ( #2 ) }
}
```

`\dtlpadleadingzerosminus`

```
\newcommand{\dtlpadleadingzerosminus}{-}
```

`\dtlpadleadingzerosplus`

```
\newcommand{\dtlpadleadingzerosplus}{}
```

```
\cs_new:Nn \__datatool_int_leading_zeros:nn
{
 \bool_lazy_and:nnTF
    { \int_compare_p:nNn { #1 } > { \c_one_int } }
    { \int_compare_p:nNn { #2 } < { 10 } }
  { \prg_replicate:nn { #1 - \c_one_int } { 0 } }
  {
    \bool_lazy_and:nnTF
      { \int_compare_p:nNn { #1 } > { 2 } }
      { \int_compare_p:nNn { #2 } < { 100 } }
     { \prg_replicate:nn { #1 - 2 } { 0 } }
     {
       \bool_lazy_and:nnTF
         { \int_compare_p:nNn { #1 } > { 3 } }
         { \int_compare_p:nNn { #2 } < { 1000 } }
        { \prg_replicate:nn { #1 - 3 } { 0 } }
        {
          \bool_lazy_and:nnTF
            { \int_compare_p:nNn { #1 } > { 4 } }
            { \int_compare_p:nNn { #2 } < { 10000 } }
           { \prg_replicate:nn { #1 - 4 } { 0 } }
           {
             \bool_lazy_and:nnTF
               { \int_compare_p:nNn { #1 } > { 5 } }
               { \int_compare_p:nNn { #2 } < { 100000 } }
              { \prg_replicate:nn { #1 - 5 } { 0 } }
              {
                \bool_lazy_and:nnT
                  { \int_compare_p:nNn { #1 } > { 6 } }
                  { \int_compare_p:nNn { #2 } < { 1000000 } }
                 { \prg_replicate:nn { #1 - 6 } { 0 } }
              }
           }
        }
     }
  }
}
\cs_generate_variant:Nn \__datatool_int_leading_zeros:nn { en, ee }
```

Pad trailing zeros for a plain decimal number (stored in a token list variable):

```
\cs_new:Nn \datatool_pad_trailing_zeros:Nn
{
  \int_compare:nNnT { #2 } > { \c_zero_int }
  {
    \tl_if_in:NnTF #1 { . }
    {
      \exp_after:wN \__datatool_split_decimal:wNN #1 \q_stop
        \l__datatool_prefix_tl
        \l__datatool_suffix_tl
      \int_step_inline:nnn { \tl_count:N \l__datatool_suffix_tl + 1 } { #2 }
```

```
        {
          \tl_put_right:Nn #1 { 0 }
        }
    }
    {
      \tl_put_right:Nn #1 { . }
      \int_step_inline:nn { #2 }
        {
          \tl_put_right:Nn #1 { 0 }
        }
    }
  }
}
\cs_generate_variant:Nn \datatool_pad_trailing_zeros:Nn { cn }
\cs_new:Npn \__datatool_split_decimal:wNN #1 . #2 \q_stop #3 #4
{
  \tl_set:Nn #3 { #1 }
  \tl_set:Nn #4 { #2 }
}
```

2.2.1 General List Utilities

`\@dtl@assigntmpseq` Assign `\l__datatool_tmp_seq` taking into account the trim spaces and skip empty elements settings. The argument may be a single token (a command whose definition is a comma-separated list), in which case it will be expanded once, or the argument should be a comma-separated list. Any nested use will need to be scoped.

```
\newcommand{\@dtl@assigntmpseq}[1]{
  \tl_if_single_token:nTF { #1 }
  {
    \exp_args:No \__datatool_create_tmp_list:n { #1 }
  }
  {
    \__datatool_create_tmp_list:n { #1 }
  }
}
\cs_new:Nn \__datatool_create_tmp_list:n
 {
   \bool_if:NTF \l__datatool_list_trim_bool
   {
     \seq_set_from_clist:Nn \l__datatool_tmp_seq { #1 }
   }
   {
     \seq_set_split_keep_spaces:Nnn \l__datatool_tmp_seq { , } { #1 }
   }
   \ifDTLlistskipempty
    \seq_remove_all:Nn \l__datatool_tmp_seq {}
   \fi
 }
```

\DTLifinlist

```
\DTLifinlist{⟨element⟩}{⟨list⟩}{⟨true part⟩}{⟨false part⟩}
```

If ⟨*element*⟩ is contained in the comma-separated list given by ⟨*list*⟩, then do ⟨*true part*⟩ otherwise do false part. The ⟨*list*⟩ may be a command whose definition is a comma-separated list. Rewritten in v3.0 to use LaTeX3. This is fractionally slower than the old definition but more reliable.

```
\newcommand*{\DTLifinlist}[4]{
 \@dtl@assigntmpseq{#2}
 \seq_if_in:NnTF \l__datatool_tmp_seq { #1 } { #3 } { #4 }
}
```

\DTLlistelement

```
\DTLlistelement{⟨list⟩}{⟨idx⟩}
```

Does the ⟨*idx*⟩th element in the list. The index should start from 1 for the first element.

```
\newrobustcmd{\DTLlistelement}[2]{%
 \@dtl@assigntmpseq{#1}
 \seq_item:Nn \l__datatool_tmp_seq { #2 }
}
```

\DTLfetchlistelement

```
\DTLfetchlistelement{⟨list⟩}{⟨idx⟩}{⟨cs⟩}
```

Fetches the ⟨*idx*⟩th element in the list and stores in ⟨*cs*⟩. The index should start from 1 for the first element.

```
\newrobustcmd{\DTLfetchlistelement}[3]{%
 \@dtl@assigntmpseq{#1}
 \tl_set:Nx #3 { \seq_item:Nn \l__datatool_tmp_seq { #2 } }
}
```

\DTLnumitemsinlist

```
\DTLnumitemsinlist{⟨list⟩}{⟨cmd⟩}
```

Counts number of elements in list and stores result in control sequence ⟨*cmd*⟩.

```
\newrobustcmd{\DTLnumitemsinlist}[2]{%
  \@dtl@assigntmpseq{#1}
  \tl_set:Nx #2 { \seq_count:N \l__datatool_tmp_seq }
}
```

DefaultLocaleWordHandler

```
\newcommand{\DTLDefaultLocaleWordHandler}[1]{%
  \DTLCurrentLocaleWordHandler{#1}%
  \appto#1{\datatoolctrlboundary}%
}
```

CurrentLocaleWordHandler

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

\ExplSyntaxOff
```

The original sorting algorithm required a comparison command. This isn't a problem for a simple case-sensitive character code sort, but it's less efficient if the sort value needs to be obtained by processing the original value. If this is done by the comparison command, then it has to be done every time a comparison is made (for both values). It's more efficient to process the sort values first and then sort, but this means keeping track of the original value.

Provide a convenient way of locally redefining commands while obtaining the sort value.

\dtlSortWordCommands

```
\newcommand*{\dtlSortWordCommands}{%
 \begingroup\makeatletter
 \dtl@SortWordCommands
}
```

\dtl@SortWordCommands

```
\newcommand{\dtl@SortWordCommands}[1]{%
  \gappto\dtl@SortWordCommands@hook{#1}%
  \endgroup
}
```

\datatoolasciistart Does nothing but influences sorting.

```
\newcommand{\datatoolasciistart}{}
```

\datatoolasciiend Does nothing but influences sorting.

```
\newcommand{\datatoolasciiend}{}
```

\datatoolctrlboundary Does nothing but influences sorting.

```
\newcommand{\datatoolctrlboundary}{}
```

\dtltexorsort

```
\newcommand{\dtltexorsort}[2]{#1}
```

tl@SortWordCommands@hook

```
\begingroup
\ExplSyntaxOn
\char_set_catcode_other:n {0}
\char_set_catcode_other:n {28}
\char_set_catcode_other:n {29}
\char_set_catcode_other:n {30}
\char_set_catcode_other:n {31}
\char_set_catcode_other:n {127}
\ExplSyntaxOff
\gdef\dtl@SortWordCommands@hook{%
```

```
  \def\datatoolasciistart{^^@}%
  \def\datatoolpersoncomma{^^1c}%
  \def\datatoolplacecomma{^^1d}%
  \def\datatoolsubjectcomma{^^1e}%
  \def\datatoolparenstart{^^1f}%
  \def\datatoolctrlboundary{^^1f}%
  \def\datatoolparen{\datatoolctrlboundary\@gobble}%
  \def\datatoolasciiend{^^7f}%
  \let\nobreakspace\space
  \let\ \space
  \edef\${\expandafter\@gobble\string\$}%
  \edef\_{\expandafter\@gobble\string\_}%
  \edef\#{\expandafter\@gobble\string\#}%
  \edef\%{\expandafter\@gobble\string\%}%
  \edef\&{\expandafter\@gobble\string\&}%
  \let\dtltexorsort\@secondoftwo
  \def\TeX{TeX}%
  \def\LaTeX{LaTeX}%
  \datatoolSetCurrencySort
}
\endgroup

\ExplSyntaxOn
```

Apply basic sort pre-processing (no locale):

```
\cs_new:Nn \datatool_sort_preprocess:Nn
 {
   \datatool_sort_preprocess:NnN #1 { #2 }
    \l_datatool_sort_to_lowercase_bool
 }
```

Version 3.2: added variant that takes boolean final argument to determine whether or not to convert to lower case.

```
\cs_new:Nn \datatool_sort_preprocess:NnN
 {
   \group_begin:
    \dtl@SortWordCommands@hook
    \bool_if:NTF #3
     {
       \datatool_sort_handler_preprocess:Nn #1 { \text_lowercase:n { #2 } }
     }
     {
       \datatool_sort_handler_preprocess:Nn #1 { #2 }
     }
    \exp_args:NNNe
   \group_end:
    \tl_set:Nn #1 { \exp_args:NV \text_purify:n #1 }
 }
```

\DTLsortwordlist

```
\DTLsortwordlist{⟨clist-var⟩}{⟨handler-cs⟩}
```

The first argument is a macro containing the list. The second argument is a handler macro for converting the original value into a byte sequence, which needs to have the form `\cs {`⟨*original*⟩`}{`⟨*tl*⟩`}`, where ⟨*original*⟩ is the original value and ⟨*tl*⟩ is a token list in which to store the byte sequence.

```
\seq_new:N \l__datatool_wordlist_seq
\tl_new:N \l__datatool_word_tl
\NewDocumentCommand \DTLsortwordlist { m m }
{
  \datatool_sortwordlist:NNN #1 #2 \__datatool_append_sorteditem:w
}
```

Syntax: ⟨*clist-var*⟩ ⟨*handler-cs*⟩ ⟨*append-cs*⟩

The ⟨*append-cs*⟩ function is used after sorting to append the sort markup to a token list register with each item in the form `\__datatool_sorted:nnn {`⟨*actual*⟩`} {`⟨*sort value*⟩`} {`⟨*letter group*⟩`}`. The ⟨*append-cs*⟩ function should have the syntax ⟨*sort value*⟩`\q_mark`⟨*actual*⟩`\q_stop` ⟨*tl var*⟩

```
\cs_new:Nn \datatool_sortwordlist:NNN
 {
  \__datatool_sortword_list:NN #1 #2
  \__datatool_finish_sortword_list:NN #1 #3
 }
```

Syntax: ⟨*clist-var*⟩ ⟨*handler-cs*⟩

```
\cs_new:Nn \__datatool_sortword_list:NN
{
```

Scope to localise the effect of the hook.

```
  \group_begin:
  \dtl@SortWordCommands@hook
  \seq_clear:N \l__datatool_wordlist_seq
```

Convert clist to temporary sequence variable, according to the current settings.

```
  \exp_args:No \__datatool_create_tmp_list:n { #1 }
  \__datatool_sortword_seq:NN \l__datatool_tmp_seq #2
  \exp_args:No
   \__datatool_start_sortword_list:n
     { \l__datatool_wordlist_seq }
}
```

Alternative function if list is already in a sequence. Syntax: ⟨*seq-var*⟩ ⟨*handler-cs*⟩

```
\cs_new:Nn \datatool_sortwordseq:NN
 {
   \datatool_sortwordseq:NNN #1 #2 \__datatool_append_sorted_seq_item:w
 }
```

Syntax: ⟨*seq-var*⟩ ⟨*handler-cs*⟩ ⟨*append-cs*⟩

```
\cs_new:Nn \datatool_sortwordseq:NNN
 {
```

```
  \__datatool_sortword_seqlist:NN #1 #2
  \__datatool_finish_sortword_seq:NN #1 #3
 }
```
Syntax: ⟨*seq-var*⟩ ⟨*handler-cs*⟩
```
\cs_new:Nn \__datatool_sortword_seqlist:NN
{
```
Scope to localise the effect of the hook.
```
  \group_begin:
  \dtl@SortWordCommands@hook
  \seq_clear:N \l__datatool_wordlist_seq
  \__datatool_sortword_seq:NN #1 #2
  \exp_args:No
   \__datatool_start_sortword_list:n
     { \l__datatool_wordlist_seq }
}
```
Syntax: ⟨*seq-var*⟩ ⟨*handler-cs*⟩ Inner workings.
```
\cs_new:Nn \__datatool_sortword_seq:NN
{
  \seq_map_inline:Nn #1
   {
     \bool_if:NTF \l__datatool_sort_datum_bool
     {
       \DTLparse \l__datatool_tmpa_tl { ##1 }
       \tl_set:No \l__datatool_word_tl { ##1 }
       #2 { ##1 } { \l__datatool_word_tl }
       \exp_args:Noo \__datatool_sortword_append:nn
        { \l__datatool_word_tl }
        { \l__datatool_tmpa_tl }
     }
     {
       \tl_set:Nn \l__datatool_word_tl { ##1 }
       #2 { ##1 } { \l__datatool_word_tl }
       \exp_args:No \__datatool_sortword_append:nn
        { \l__datatool_word_tl } { ##1 }
     }
   }
}
```
End group to cancel effect of sort hook. This allows the original definitions to be used by the fallback function.
```
\cs_new:Nn \__datatool_start_sortword_list:n
{
   \group_end:
   \tl_set:Nn \l__datatool_wordlist_seq { #1 }
   \seq_sort:Nn \l__datatool_wordlist_seq
   {
     \__datatool_compare_sortitem:w ##1 ##2
   }
}
```

Add sort element to sequence.

```
\cs_new:Nn \__datatool_sortword_append:nn
 {
    \seq_put_right:Nn \l__datatool_wordlist_seq
     { #1 \q_mark #2 \q_stop }
 }
```

Comparator used by `\DTLsortwordlist`

```
\__datatool_compare_sortitem:w
⟨sort1⟩\q_mark ⟨actual1⟩\q_stop
⟨sort2⟩\q_mark ⟨actual2⟩\q_stop
```

```
\cs_new:Npn \__datatool_compare_sortitem:w
 #1\q_mark#2\q_stop#3\q_mark#4\q_stop
 {
   \__datatool_compare_sortitem:nnnn
    { #1 } { #2 } { #3 } { #4 }
   \int_compare:nNnTF
       { \dtl@sortresult } < { \c_zero_int }
    {
      \sort_return_same:
    }
    {
      \int_compare:nNnTF
         { \dtl@sortresult } > { \c_zero_int }
       {
         \sort_return_swapped:
       }
       {
         \bool_if:NTF \l__datatool_sort_reverse_bool
          {
            \__datatool_fallback_action:nnnn { #4 } { #2 }
             { \sort_return_swapped: }
             { \sort_return_same: }
          }
          {
            \__datatool_fallback_action:nnnn { #2 } { #4 }
             { \sort_return_swapped: }
             { \sort_return_same: }
          }
       }
    }
 }
```

Syntax: {⟨*sort A*⟩}{⟨*actual A*⟩}{⟨*sort B*⟩}{⟨*actual B*⟩} If the actual values are in the datum format, a numeric comparison can be used if the datum format indicates the original value was identified as numeric.

```
\cs_new:Nn \__datatool_compare_sortitem:nnnn
```

```
{
  \tl_clear:N \l__datatool_tmpc_tl
  \tl_clear:N \l__datatool_tmpd_tl
  \int_set_eq:NN \@dtl@datatype \c_datatool_string_int
  \tl_if_head_eq_meaning:nNTF { #2 }  \__datatool_datum:nnnn
   {
     \__datatool_get_datum:w #2 \q_nil \q_stop
     \int_set_eq:NN
       \l__datatool_tmp_datatype_int
       \@dtl@datatype
     \tl_set:No \l__datatool_tmpc_tl
       { \datatool_datum_currency:Nnnnn #2 }
   }
   {
     \datatool_if_null:nT { #2 }
      {
        \int_set_eq:NN
          \l__datatool_tmp_datatype_int
          \c_datatool_unknown_int
      }
   }
  \tl_if_head_eq_meaning:nNTF { #4 }  \__datatool_datum:nnnn
   {
     \__datatool_get_datum:w #4 \q_nil \q_stop
     \tl_set:No \l__datatool_tmpd_tl
       { \datatool_datum_currency:Nnnnn #4 }
   }
   {
     \datatool_if_null:nT { #4 }
      {
        \int_set_eq:NN
          \@dtl@datatype
          \c_datatool_unknown_int
      }
   }
```

If they have different currency symbols, treat them as strings.

```
  \tl_if_eq:NNF \l__datatool_tmpc_tl \l__datatool_tmpd_tl
   {
     \int_set_eq:NN
       \l__datatool_tmp_datatype_int \c_datatool_string_int
     \int_set_eq:NN
       \@dtl@datatype \c_datatool_string_int
   }
  \int_compare:nNnTF
    { \l__datatool_tmp_datatype_int }
      =
    { \c_datatool_unknown_int }
   {
```

The first value has an unknown type.

```
      \int_compare:nNnTF
       { \@dtl@datatype } > { \c_datatool_string_int }
      {
```

The first value has an unknown type. The second is numeric, so treat the first as zero.

```
        \__datatool_numcmp:Nnn \dtl@sortresult
         { 0 }
         { \datatool_datum_value:Nnnnn #4 }
      }
      {
```

The first value has an unknown type. If the second is not numeric so use a string comparison.

```
        \__datatool_strcmp:Nnn \dtl@sortresult { #1 } { #3 }
      }
    }
    {
```

The first value has a known type.

```
      \int_case:nnF { \@dtl@datatype }
      {
        { \c_datatool_unknown_int }
          {
```

The first value is known. The second is unknown.

```
            \int_compare:nNnTF
               { \l__datatool_tmp_datatype_int }
                 >
               { \c_datatool_string_int }
             {
```

The first value is numeric. The second is unknown so treat as 0.

```
               \__datatool_numcmp:Nnn \dtl@sortresult
                { \datatool_datum_value:Nnnnn #2 }
                { 0 }
             }
             {
```

The first value is not numeric. The second is unknown so use a string comparison.

```
               \__datatool_strcmp:Nnn \dtl@sortresult { #1 } { #3 }
             }
          }
        { \c_datatool_string_int }
          {
```

The first value is known. The second is a string. Use a string comparison

```
            \__datatool_strcmp:Nnn \dtl@sortresult { #1 } { #3 }
          }
      }
      {
```

The first value is known. The second is numeric.

```
        \int_compare:nNnTF
           { \l__datatool_tmp_datatype_int }
            >
           { \c_datatool_string_int }
         {
```

Both values are numeric.

```
           \__datatool_numcmp:Nnn \dtl@sortresult
            { \datatool_datum_value:Nnnnn #2 }
            { \datatool_datum_value:Nnnnn #4 }
         }
         {
```

The first value is a string. The second is numeric. Use a string comparison.

```
           \__datatool_strcmp:Nnn \dtl@sortresult { #1 } { #3 }
         }
      }
    }
   \bool_if:NT \l__datatool_sort_reverse_bool
    {
      \int_compare:nNnTF
        { \dtl@sortresult } < { \c_zero_int }
       {
         \int_set:Nn \dtl@sortresult { \c_one_int }
       }
       {
         \int_set:Nn \dtl@sortresult { - \dtl@sortresult }
       }
    }
 }
```

Finish off after sorting. The first argument is the clist in which to store the sorted values. The second argument is the function that extracts the required information and appends it to the clist.

```
 \cs_new:Nn \__datatool_finish_sortword_list:NN
  {
    \clist_clear:N #1
    \seq_map_inline:Nn \l__datatool_wordlist_seq
    {
```

The argument ##1 should be in the form ⟨*sort*⟩`\q_mark` ⟨*actual*⟩`\q_stop{`⟨*tl*⟩`}`

```
      #2 ##1 #1
    }
  }
```

Similarly, but for a sequence variable.

```
 \cs_new:Nn \__datatool_finish_sortword_seq:NN
  {
    \seq_clear:N #1
    \seq_map_inline:Nn \l__datatool_wordlist_seq
    {
```

The argument ##1 should be in the form ⟨*sort*⟩\q_mark ⟨*actual*⟩\q_stop{⟨*tl*⟩}

```
      #2 ##1 #1
    }
  }
```

This is similar to the datum marker but allows the sort value and initial letter (which may be required for letter groups) to be available. The three arguments are: original item, sort string, letter group.

```
\cs_new:Nn \__datatool_sorted:nnn { \exp_not:n { #1 }}
```

Unlike the datum markers, these are more likely to be passed directly (for example, in the argument of \do) so the ⟨*sort marker*⟩ is expected to be in the form __datatool_sorted:nnn{⟨*actual*⟩}{⟨*sort*⟩}{⟨*letter*⟩}. If using \@for to iterate over the list then the loop control sequence will need to be expanded first.

\DTLsortedactual{⟨*sort marker*⟩}

\DTLsortedactual

The original item.

```
\newcommand{\DTLsortedactual} [1] { \use_ii:nnnn #1 }
```

\DTLsortedvalue{⟨*sort marker*⟩}

\DTLsortedvalue

The sort value.

```
\newcommand{\DTLsortedvalue} [1] { \use_iii:nnnn #1 }
```

\DTLsortedletter{⟨*sort marker*⟩}

\DTLsortedletter

The letter group.

```
\newcommand{\DTLsortedletter} [1] { \use_iv:nnnn #1 }
```

__datatool_append_sorteditem:w ⟨*sort*⟩\q_mark ⟨*actual*⟩\q_stop ⟨*clist_var*⟩

```
\cs_new:Npn \__datatool_append_sorteditem:w
  #1 \q_mark #2 \q_stop #3
{
  \tl_clear:N \l__datatool_tmpa_tl
```

```
  \DTLassignlettergroup { #2 } { #1 } { \l__datatool_tmpa_tl }
  \clist_put_right:Nx #3
    { \exp_not:N \__datatool_sorted:nnn
        { \exp_not:n { #2 } } { \exp_not:n { #1 } }
        { \exp_not:o { \l__datatool_tmpa_tl } }
    }
}
\cs_new:Npn \__datatool_append_sorted_seq_item:w
  #1 \q_mark #2 \q_stop #3
{
  \tl_clear:N \l__datatool_tmpa_tl
  \DTLassignlettergroup { #2 } { #1 } { \l__datatool_tmpa_tl }
  \seq_put_right:Nx #3
    { \exp_not:N \__datatool_sorted:nnn
        { \exp_not:n { #2 } } { \exp_not:n { #1 } }
        { \exp_not:o { \l__datatool_tmpa_tl } }
    }
}
```

\DTLassignlettergroup

\DTLassignlettergroup{⟨*actual*⟩}{⟨*sort value*⟩}{⟨*tl*⟩}

Obtains the letter group from the sort value.

```
\newcommand{\DTLassignlettergroup} [3]
 {
   \@dtl@datatype = \c_datatool_string_int
   \tl_if_head_eq_meaning:nNT { #1 } \__datatool_datum:nnnn
    {
      \__datatool_get_datum:w #1 \q_nil \q_stop
    }
   \int_case:nn { \@dtl@datatype }
   {
     { \c_datatool_string_int }
      {
       \DTLCurrentLocaleGetGroupString { #1 } { #2 } #3
       \exp_args:NV \datatool_get_first_grapheme:nN #3 #3
       \exp_args:NV \datatool_if_letter:nTF #3
        {
           \exp_args:NV \DTLCurrentLocaleGetInitialLetter #3 { #3 }
           \DTLPreProcessLetterGroup { #3 }
           \tl_set:Nx #3 { \exp_not:N \dtllettergroup { #3 } }
        }
        {
          \DTLPreProcessNonLetterGroup { #3 }
          \tl_set:Nx #3 { \exp_not:N \dtlnonlettergroup { #3 } }
        }
      }
     { \c_datatool_integer_int }
      {
```

```
        \DTLPreProcessIntegerGroup
          { \l__datatool_datum_value_tl } { #1 }
        \tl_set:Nx #3
         { \exp_not:N \dtlnumbergroup { \l__datatool_datum_value_tl } }
       }
      { \c_datatool_decimal_int }
       {
        \DTLPreProcessDecimalGroup
          { \l__datatool_datum_value_tl } { #1 }
        \tl_set:Nx #3
         { \exp_not:N \dtlnumbergroup { \l__datatool_datum_value_tl } }
       }
      { \c_datatool_currency_int }
       {
        \DTLPreProcessCurrencyGroup
          { \l__datatool_datum_currency_tl }
          { \l__datatool_datum_value_tl }
          { #1 }
        \tl_set:Nx #3
         {
           \exp_not:N \dtlcurrencygroup
             { \exp_not:o { \l__datatool_datum_currency_tl } }
             { \l__datatool_datum_value_tl }
         }
       }
      { \c_datatool_datetime_int }
       {
        \DTLPreProcessDateTimeGroup
          { \l__datatool_datum_value_tl } { #1 }
        \tl_set:Nx #3
         { \exp_not:N \dtldatetimegroup { \l__datatool_datum_value_tl } }
       }
      { \c_datatool_date_int }
       {
        \DTLPreProcessDateGroup
          { \l__datatool_datum_value_tl } { #1 }
        \tl_set:Nx #3
         { \exp_not:N \dtldategroup { \l__datatool_datum_value_tl } }
       }
      { \c_datatool_time_int }
       {
        \DTLPreProcessTimeGroup
          { \l__datatool_datum_value_tl } { #1 }
        \tl_set:Nx #3
         { \exp_not:N \dtltimegroup { \l__datatool_datum_value_tl } }
       }
    }
  }
```

rentLocaleGetGroupString

```
\DTLCurrentLocaleGetGroupString{⟨actual⟩}{⟨sort value⟩}{⟨tl⟩}
```

May consist of more than one character. This determines whether to use the sort or actual value to obtain the letter group. The returned value may be truncated as only the initial part is of interest.

```
\newcommand{\DTLCurrentLocaleGetGroupString}[3]
{
  \tl_set:Nn #3 { #1 }
}
```

DTLPreProcessLetterGroup Hook to pre-process letter group. The argument is a token list variable containing the current value.

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

PreProcessNonLetterGroup Hook to pre-process non-letter group. The argument is a token list variable containing the current value.

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

TLPreProcessIntegerGroup Hook to pre-process integer number group. The first argument is a token list variable containing the current value. The second argument is the actual value. The hook may adjust the token list variable.

```
\newcommand{\DTLPreProcessIntegerGroup}[2]{}
```

TLPreProcessDecimalGroup Hook to pre-process decimal number group. The first argument is a token list variable containing the current value. The second argument is the actual value. The hook may adjust the token list variable.

```
\newcommand{\DTLPreProcessDecimalGroup}[2]{}
```

LPreProcessCurrencyGroup

```
\DTLPreProcessCurrencyGroup{⟨sym tl var⟩}{⟨num tl var⟩}{⟨actual⟩}
```

Hook to pre-process currency number group. The ⟨*num tl var*⟩ is a token list variable containing the current numeric value. The ⟨*sym tl var*⟩ is a token list variable containing the currency symbol. The ⟨*actual*⟩ argument is the actual value. The hook may adjust the token list variable.

```
\newcommand{\DTLPreProcessCurrencyGroup}[3]{}
```

LPreProcessDateTimeGroup Hook to pre-process datetime group. The first argument is a token list variable containing the current value. The second argument is the actual value. The hook may adjust the token list variable.

```
\newcommand{\DTLPreProcessDateTimeGroup}[2]{}
```

\DTLPreProcessDateGroup Hook to pre-process date group. The first argument is a token list variable containing the current value. The second argument is the actual value. The hook may adjust the token list variable.

```
\newcommand{\DTLPreProcessDateGroup}[2]{}
```

\DTLPreProcessTimeGroup Hook to pre-process time group. The first argument is a token list variable containing the current value. The second argument is the actual value. The hook may adjust the token list variable.

```
\newcommand{\DTLPreProcessTimeGroup}[2]{}
```

\dtllettergroup

```
\newcommand{\dtllettergroup}[1]{ \text_titlecase_first:n { #1 } }
```

\dtlnonlettergroup

```
\newcommand{\dtlnonlettergroup}[1]{\detokenize{#1}}
```

\dtlnumbergroup

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

\dtlcurrencygroup

```
\newcommand{\dtlcurrencygroup}[2]{#1}
```

\dtldatetimegroup

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

\dtldategroup

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

\dtltimegroup

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

\dtlfallbackaction

```
\dtlfallbackaction{⟨val1⟩}{⟨val2⟩}{⟨swap code⟩}{⟨no swap code⟩}
```

Used when two sort values are identical. This uses the original values for comparison.

```
\newcommand{\dtlfallbackaction}[4]{
  \DTLifstringgt { #1 } { #2 } { #3 } { #4 }
}
\cs_new:Nn \__datatool_fallback_action:nnnn
{
  \dtlfallbackaction { #1 } { #2 } { #3 } { #4 }
}
```

Word sort doesn't discard spaces or punctuation. So the first word of a compound word or phrase will come before a longer word that happens to have the other word at the start. This means that, for example, "sea lion" will come before "seal" because the space character comes before the character "l". This isn't exactly the same as a character sort as the localisation handler may convert the token list to series of bytes that alters the ordering.

Letter sort is only really concerned with alphanumerics (letters and digits). Spaces and punctuation aren't considered relevant to order. Commands will be expanded first, then stripped after the locale handler has been applied. This will ensure that any accent commands, such as `\'e`, will be converted to UTF-8 before the locale handler is applied.

```
\cs_new:Nn \datatool_sort_handler_preprocess:Nn
{
  \protected@edef #1 { #2 }
}
```

Handlers should set this to true if the sort value is converted to lowercase and false otherwise.

```
\bool_new:N \l_datatool_sort_to_lowercase_bool
\bool_set_true:N \l_datatool_sort_to_lowercase_bool
```

\DTLsortwordhandler

\DTLsortwordhandler{⟨*original*⟩}{⟨*cs*⟩}

```
\newcommand{\DTLsortwordhandler}[2]{
  \bool_set_true:N \l_datatool_sort_to_lowercase_bool
  \datatool_sort_handler_preprocess:Nn
     #2 { #1 }
  \DTLDefaultLocaleWordHandler { #2 }
```

v3.1: Case change has been moved from pre-process to here:

```
  \tl_set:Ne #2 { \text_lowercase:n { \text_purify:n { #2 } } }
}
```

\DTLsortwordcasehandler

\DTLsortwordcasehandler{⟨*original*⟩}{⟨*cs*⟩}

Case-sensitive handler.

```
\newcommand{\DTLsortwordcasehandler}[2]{
  \bool_set_false:N \l_datatool_sort_to_lowercase_bool
  \datatool_sort_handler_preprocess:Nn
    #2 { #1 }
  \DTLDefaultLocaleWordHandler { #2 }
  \tl_set:Ne #2 { \text_purify:n { #2 } }
}
```

Regular expression for content that should be stripped for letter sort.

```
\regex_new:N \l_datatool_letter_regex
\regex_set:Nn \l_datatool_letter_regex { [\-[:space:]] }
```

\DTLsortletterhandler

\DTLsortletterhandler{⟨*original*⟩}{⟨*cs*⟩}

Hyphens and spaces are discarded from the sort value.

```
\newcommand{\DTLsortletterhandler}[2]{
  \bool_set_true:N \l_datatool_sort_to_lowercase_bool
  \datatool_sort_handler_preprocess:Nn
    #2 { #1 }
  \regex_replace_all:NnN \l_datatool_letter_regex {} #2
  \DTLDefaultLocaleWordHandler { #2 }
```

v3.1: Case change has been moved from pre-process to here:

```
  \tl_set:Ne #2 { \text_lowercase:n { \text_purify:n { #2 } } }
}
```

DTLsortlettercasehandler

\DTLsortlettercasehandler{⟨*original*⟩}{⟨*cs*⟩}

Case-sensitive handler.

```
\newcommand{\DTLsortlettercasehandler}[2]{
  \bool_set_false:N \l_datatool_sort_to_lowercase_bool
  \datatool_sort_handler_preprocess:Nn
    #2 { #1 }
  \regex_replace_all:NnN \l_datatool_letter_regex {} #2
  \DTLDefaultLocaleWordHandler { #2 }
  \tl_set:Ne #2 { \text_purify:n { #2 } }
}
```

\dtlsortlist

\dtlsortlist{⟨*list-cs*⟩}{⟨*criteria cmd*⟩}

Sorts the given comma-separated list according to the ⟨*criteria*⟩ command, which must take three arguments.

```
\newcommand{\dtlsortlist}[2]{%
  \exp_args:No \__datatool_create_tmp_list:n { #1 }
  \seq_set_eq:NN \l__datatool_wordlist_seq \l__datatool_tmp_seq
  \seq_sort:Nn \l__datatool_wordlist_seq
  {
    \bool_if:NTF \l__datatool_sort_reverse_bool
    {
      #2 { \dtl@sortresult } { ##2 } { ##1 }
    }
    {
      #2 { \dtl@sortresult } { ##1 } { ##2 }
```

```
    }
    \ifnum\dtl@sortresult > 0\relax
      \sort_return_swapped:
    \else
      \sort_return_same:
    \fi
  }
  \tl_clear:N #1
  \seq_map_inline:Nn \l__datatool_wordlist_seq
  {
    \tl_if_empty:NF #1
     {
       \tl_put_right:Nn #1 { , }
     }
    \tl_if_in:nnTF { ##1 } { , }
    {
      \tl_put_right:Nn #1 { { ##1 } }
    }
    {
      \tl_put_right:Nn #1 { ##1 }
    }
  }
}
```

\dtlinsertinto

\dtlinsertinto{⟨*element*⟩}{⟨*sorted-list*⟩}{⟨*criteria cmd*⟩}

Globally inserts ⟨*element*⟩ into the sorted list ⟨*sorted-list*⟩ according to the criteria given by ⟨*criteria cmd*⟩, which should be a command that takes three arguments {⟨*reg*⟩}{⟨*A*⟩}{⟨*B*⟩}, where ⟨*reg*⟩ is a count register in which to store the result, ⟨*A*⟩ is the first element and ⟨*B*⟩ is the second element to compare.

```
\newrobustcmd{\dtlinsertinto}[3]{%
  \exp_args:No \__datatool_create_tmp_list:n { #2 }
  \seq_set_eq:NN \l__datatool_wordlist_seq \l__datatool_tmp_seq
  \tl_gclear:N #2
  \@dtl@insertdonefalse
  \seq_map_inline:Nn \l__datatool_wordlist_seq
  {
    \tl_if_empty:NF #2
     {
       \tl_gput_right:Nn #2 { , }
     }
    \if@dtl@insertdone
    \else
      #3 { \dtl@sortresult } { ##1 } { #1 }
      \ifnum\dtl@sortresult > 0\relax
        \@dtl@insertdonetrue
        \tl_if_in:nnTF { #1 } { , }
```

```
        {
          \tl_gput_right:Nn #2 { { #1 } , }
        }
        {
          \tl_gput_right:Nn #2 { #1 , }
        }
      \fi
    \fi
    \tl_if_in:nnTF { ##1 } { , }
    {
      \tl_gput_right:Nn #2 { { ##1 } }
    }
    {
      \tl_gput_right:Nn #2 { ##1 }
    }
  }
  \if@dtl@insertdone
  \else
    \tl_if_empty:NF #2
     {
       \tl_gput_right:Nn #2 { , }
     }
    \tl_if_in:nnTF { #1 } { , }
     {
       \tl_gput_right:Nn #2 { { #1 } }
     }
     {
       \tl_gput_right:Nn #2 { #1 }
     }
  \fi
}
```

\edtlinsertinto

\edtlinsertinto{⟨*element*⟩}{⟨*sorted-list*⟩}{⟨*criteria cmd*⟩}

First expands ⟨*element*⟩ before inserting into the list.

```
\newcommand*{\edtlinsertinto}[3]{%
  \exp_args:Nx \dtlinsertinto {#1} {#2} {#3}%
}
\ExplSyntaxOff
```

\if@dtl@insertdone Define conditional to indicate whether the new entry has been inserted into the sorted list.

```
\newif\if@dtl@insertdone
```

\dtl@sortresult Define \dtl@sortresult to be set by comparison macro. TODO: replace with L3 int var? (NB used by glossaries.sty)

```
\newcount\dtl@sortresult
```

```
\ExplSyntaxOn
```

\DTLshufflelist

\DTLshufflelist⟨*clist-var*⟩

Shuffle a comma-separated list. This just converts the list to a sequence and shuffles that using `seq_shuffle:N` and then converts that back to a comma-separated list.

```
\NewDocumentCommand \DTLshufflelist { m }
{
```

Convert clist to temporary sequence variable, according to the current settings.

```
  \group_begin:
  \exp_args:No \__datatool_create_tmp_list:n { #1 }
  \seq_shuffle:N \l__datatool_tmp_seq
  \clist_clear:N \l__datatool_tmp_clist
  \seq_map_inline:Nn \l__datatool_tmp_seq
   {
     \clist_put_right:Nn \l__datatool_tmp_clist { ##1 }
   }
  \exp_args:NNNV
  \group_end:
  \tl_set:Nn #1 \l__datatool_tmp_clist
}

\ExplSyntaxOff
```

This next command is based on the list iteration exercise given at http://www.dickimaw-books.com/latex/admin/html/foreachtips.shtml#oxfordcomma It's designed to format a comma-separated list using `\DTLlistformatsep` between each item except for the last. The separator for the last pair is `\DTLlistformatlastsep` if the list only contains two items or `\DTLlistformatoxford\DTLlistformatlastsep` if the list contains three or more items. Each item is formatted according to `\DTLlistformatitem`.

\DTLlistformatsep

```
\newcommand*{\DTLlistformatsep}{, }
```

\DTLlistformatoxford

```
\newcommand*{\DTLlistformatoxford}{}
```

\DTLandname

```
\ifdef\andname
{\newcommand*{\DTLandname}{\andname}}
{\newcommand*{\DTLandname}{\&}}
```

\DTLlistformatlastsep

```
\newcommand*{\DTLlistformatlastsep}{ \DTLlistand\space}
```

\DTLlistformatitem

```
\newcommand*{\DTLlistformatitem}[1]{#1}
```

\@dtl@formatlist@handler

```
\newcommand*{\@dtl@formatlist@handler}[1]{%
  \@dtl@formatlist@itemsep
  \@dtl@formatlist@lastitem
  \renewcommand{\@dtl@formatlist@lastitem}{%
    \renewcommand{\@dtl@formatlist@itemsep}{%
      \DTLlistformatsep
      \renewcommand*{\@dtl@formatlist@prelastitemsep}{%
        \DTLlistformatoxford}}%
    \renewcommand{\@dtl@formatlist@prelastitem}{%
      \@dtl@formatlist@prelastitemsep
       \DTLlistformatlastsep}%
    \DTLlistformatitem{#1}%
  }%
}%
```

\DTLformatlist Formats the comma-separated list supplied in its argument. The unstarred version adds grouping.

```
\newrobustcmd*{\DTLformatlist}{%
 \@ifstar{\s@dtlformatlist}{\@dtlformatlist}%
}
```

\s@dtlformatlist Starred version of `\DTLformatlist` doesn't add grouping.

```
\ExplSyntaxOn
\newcommand*{\s@dtlformatlist}[1]{%
  \def\@dtl@formatlist@itemsep{}%
  \def\@dtl@formatlist@lastitem{}%
  \def\@dtl@formatlist@prelastitem{}%
  \def\@dtl@formatlist@prelastitemsep{}%
  \@dtl@assigntmpseq{#1}%
  \seq_map_function:NN \l__datatool_tmp_seq \@dtl@formatlist@handler
  \@dtl@formatlist@prelastitem\@dtl@formatlist@lastitem
}
\ExplSyntaxOff
```

\@dtlformatlist Unstarred version of `\DTLformatlist` adds grouping.

```
\newcommand*{\@dtlformatlist}[1]{{\s@dtlformatlist{#1}}}
```

2.2.2 General Token Utilities

These commands are for altering the content of token registers. They're mostly used by datatool for the internal database structure.

```
\ExplSyntaxOn
```

\@dtl@toks@gput@right@cx

```
\dtl@toks@gput@right@cx{⟨toks name⟩}{⟨stuff⟩}
```

Globally appends stuff to token register \⟨*toks name*⟩ Deprecated. TODO remove

```
\newcommand{\@dtl@toks@gput@right@cx}[2]{%
 \__datatool_token_register_gput_right:cx #1 { #2 }
}
```

l@toks@gconcat@middle@cx

\dtl@toks@gconcat@middle@cx{⟨*toks name*⟩}{⟨*before toks*⟩}{⟨*stuff*⟩}{⟨*after toks*⟩}

Globally sets token register \⟨*toks name*⟩ to the contents of ⟨*before toks*⟩ concatenated with ⟨*stuff*⟩ (expanded) and the contents of ⟨*after toks*⟩ Deprecated. TODO remove

```
\newcommand{\@dtl@toks@gconcat@middle@cx}[4]{%
  \__datatool_token_register_gset:cx
    { #1 } { \the #2 #3 \the #4 }
}
```

Use a token register.

```
\cs_new:Nn \__datatool_token_register_use:N
{
  \the #1
}
\cs_generate_variant:Nn
  \__datatool_token_register_use:N { c }
```

Set a token register equal to another.

```
\cs_new:Nn \__datatool_token_register_set_eq:NN
{
  #1 = #2
}
\cs_generate_variant:Nn
   \__datatool_token_register_set_eq:NN
 { cN, Nc, cc }
```

Globally set a token register equal to another.

```
\cs_new:Nn \__datatool_token_register_gset_eq:NN
{
  \global #1 = #2
}
\cs_generate_variant:Nn
   \__datatool_token_register_gset_eq:NN
 { cN, Nc, cc }
```

Set the contents of a token register.

```
\cs_new:Nn \__datatool_token_register_set:Nn
{
  #1 = { #2 }
}
\cs_generate_variant:Nn
   \__datatool_token_register_set:Nn
   { cn, cx, co, cV, No, Nx, NV }
```

Globally set the contents of a token register.

```
\cs_new:Nn \__datatool_token_register_gset:Nn
{
  \global #1 = { #2 }
}
\cs_generate_variant:Nn
   \__datatool_token_register_gset:Nn
   { cn, cx, co, cV, No, Nx }
```

Append content to a token register.

```
\cs_new:Nn \__datatool_token_register_put_right:Nn
{
  \__datatool_token_register_set:No #1
   { \the #1  #2 }
}
\cs_generate_variant:Nn
   \__datatool_token_register_put_right:Nn
   { cn, cx, co, cV, No, Nx }
```

Globally append content to a token register.

```
\cs_new:Nn \__datatool_token_register_gput_right:Nn
{
  \__datatool_token_register_gset:No #1
   { \the #1  #2 }
}
\cs_generate_variant:Nn
   \__datatool_token_register_gput_right:Nn
   { cn, cx, co, cV, No, Nx }
```

Concatenate the content of two registers with extra stuff in between

```
\cs_new:Nn \__datatool_token_register_concat_middle:NNnN
{
  \__datatool_token_register_set:Nx
    #1 { \the #2 \exp_not:n { #3 } \the #4 }
}
\cs_generate_variant:Nn
   \__datatool_token_register_concat_middle:NNnN
   { cNnN, cNxN, NNxN }
```

Globally concatenate.

```
\cs_new:Nn \__datatool_token_register_gconcat_middle:NNnN
{
  \__datatool_token_register_gset:Nx
    #1 { \the #2 \exp_not:n { #3 } \the #4 }
}
\cs_generate_variant:Nn
   \__datatool_token_register_gconcat_middle:NNnN
   { cNnN, cNxN, NNxN }
```

2.3 Encodings

Load file dealing with non-ASCII characters, but first initialise ASCII defaults. Each character is represented by both a string variable (for use by the sort comparator) and a token list variable (for typesetting, if required). The difference in expansion is more noticeable with input than with native Unicode engines.

Define new variables with default ASCII value.

Regular expression to match apostrophe.

```
\regex_new:N \l_datatool_apos_regex
\regex_set:Nn \l_datatool_apos_regex { \' }
```

Most symbols are currency signs, but mid-point is also added as it's sometimes used as a number separator.

```
\cs_new:Nn \__datatool_new_symbol:nn
 {
   \__datatool_new_symbol:nnn { #1 } { #2 } { #2 }
 }
\cs_new:Nn \__datatool_new_symbol:nnn
 {
   \tl_new:c { l_datatool_ #1 _tl }
   \tl_set:cn { l_datatool_ #1 _tl } { #2 }
   \str_new:c { l_datatool_ #1 _str }
   \str_set:cn { l_datatool_ #1 _str } { #3 }
 }
\cs_generate_variant:Nn \__datatool_new_symbol:nnn { nnV }
```

Regular expression to search for these symbols, which may be used in localisation files. Only those characters supported by the encoding will be added.

```
\regex_new:N \l_datatool_currencysigns_regex
\regex_set:Nn \l_datatool_currencysigns_regex { \$ }
```

Allow encoding file to change the values.

```
\cs_new:Nn \datatool_set_symbol:nn
 {
   \str_set:cn { l_datatool_ #1 _str } { #2 }
   \tl_set:cn { l_datatool_ #1 _tl } { #2 }
 }
\cs_generate_variant:Nn \datatool_set_symbol:nn { ne }
```

Designed for inputenc to set by character code:

```
\cs_new:Nn \datatool_set_symbol_from_charcode:nn
 {
   \str_set:ce { l_datatool_ #1 _str }
    { \char_generate:nn { #2 } { 12 } }
   \exp_args:Nno \tl_set:co { l_datatool_ #1 _tl }
    { \char_generate:nn { #2 } { 13 } }
 }
```

Just for currency signs so that they can be added to the regular expression and list of known currencies:

```
\cs_new:Nn \datatool_set_currencysign:nn
```

```
 {
   \datatool_set_symbol:nn { #1 } { #2 }
   \regex_set:Nn \l_datatool_currencysigns_regex
     {
        \ur { l_datatool_currencysigns_regex }
      | \u { l_datatool_ #1 _str }
     }
   \DTLnewcurrencysymbol { #2 }
 }
\cs_generate_variant:Nn \datatool_set_currencysign:nn { ne }
\cs_new:Nn \datatool_set_currencysign_from_charcode:nn
 {
   \datatool_set_symbol_from_charcode:nn { #1 } { #2 }
   \regex_set:Nn \l_datatool_currencysigns_regex
     {
        \ur { l_datatool_currencysigns_regex }
      | \u { l_datatool_ #1 _str }
     }
 }
```

These may be redefined by an encoding file, if available. See section 2.3.

```
\__datatool_new_symbol:nn { cent } { c }
\__datatool_new_symbol:nn { pound } { L }
\__datatool_new_symbol:nnV { currency } { \# } \c_hash_str
\__datatool_new_symbol:nn { yen } { Y }
\__datatool_new_symbol:nn { middot } { . }
\__datatool_new_symbol:nn { florin } { f }
\__datatool_new_symbol:nn { baht } { B }
\__datatool_new_symbol:nn { ecu } { CE }
\__datatool_new_symbol:nn { colonsign } { C }
\__datatool_new_symbol:nn { cruzerio } { Cr }
\__datatool_new_symbol:nn { frenchfranc } { F }
\__datatool_new_symbol:nn { lira } { L }
\__datatool_new_symbol:nn { mill } { m }
\__datatool_new_symbol:nn { naira } { N }
\__datatool_new_symbol:nn { peseta } { Pts }
\__datatool_new_symbol:nn { rupee } { Rs }
\__datatool_new_symbol:nn { won } { W }
\__datatool_new_symbol:nn { shekel } { S }
\__datatool_new_symbol:nn { dong } { d }
\__datatool_new_symbol:nn { euro } { E }
\__datatool_new_symbol:nn { kip } { K }
\__datatool_new_symbol:nn { tugrik } { T }
\__datatool_new_symbol:nn { drachma } { Dr }
\__datatool_new_symbol:nn { germanpenny } { p }
\__datatool_new_symbol:nn { peso } { P }
\__datatool_new_symbol:nn { guarani } { G. }
\__datatool_new_symbol:nn { austral } { A }
\__datatool_new_symbol:nn { hryvnia } { S }
\__datatool_new_symbol:nn { cedi } { C }
```

```
\__datatool_new_symbol:nn { livretournois } { lt }
\__datatool_new_symbol:nn { spesmilo } { Sm }
\__datatool_new_symbol:nn { tenge } { T }
\__datatool_new_symbol:nn { indianrupee } { R }
\__datatool_new_symbol:nn { turkishlira } { L }
\__datatool_new_symbol:nn { nordicmark } { M }
\__datatool_new_symbol:nn { manat } { M }
\__datatool_new_symbol:nn { ruble } { R }
\__datatool_new_symbol:nn { lari } { L }
\__datatool_new_symbol:nn { bitcoin } { B }
\__datatool_new_symbol:nn { som } { c }
```

An internal list that stores all known currencies. This sequence variable and `\DTLnewcurrencysymbol` need to be defined before the ldf encoding file is input.

```
\seq_new:N \l__datatool_known_currencies_seq
```

A list of all defined currency labels:

```
\seq_new:N \l_datatool_currencies_seq
```

A list of all defined currency labels registered by regions:

```
\seq_new:N \l_datatool_regional_currencies_seq
```

Map region to currency code:

```
\prop_new:N \l_datatool_regional_currencies_prop
```

`\DTLnewcurrencysymbol`

`\DTLaddcurrency{`⟨*symbol*⟩`}`

Adds ⟨*symbol*⟩ to the list of known currencies

```
\NewDocumentCommand \DTLnewcurrencysymbol { m }
{
  \seq_if_in:NnF \l__datatool_known_currencies_seq { #1 }
   {
     \seq_put_right:Nn \l__datatool_known_currencies_seq { #1 }
   }
}

\ExplSyntaxOff

\InputIfFileExists
 {datatool-\TrackLangEncodingName .ldf}
{}
{
  \PackageInfo{datatool-base}
   {Missing file `datatool-\TrackLangEncodingName .ldf'.
    Falling back on US-ASCII for \string\datatool\string_...\string_str commands}
}
```

2.4 Locale Dependent Information

```
\ExplSyntaxOn
```

`\@dtl@decimal` The current decimal character was stored in `\@dtl@decimal` prior to version 3.0. As from v3.0, there are separate variables for formatting and parsing.

```
\tl_new:N \l__datatool_decimal_tl
```

Parsing may either be a token list:

```
\tl_new:N \l__datatool_decimal_parse_tl
```

or a regular expression:

```
\regex_new:N \l__datatool_decimal_parse_regex
```

`\@dtl@numbergroupchar` The current number group character was stored in `\@dtl@numbergroupchar` prior to version 3.0. As from v3.0, there are separate variables for formatting and parsing.

```
\tl_new:N \l__datatool_numbergroup_tl
```

Parsing may either be a token list:

```
\tl_new:N \l__datatool_numbergroup_parse_tl
```

or a regular expression:

```
\regex_new:N \l__datatool_numbergroup_parse_regex
```

Regular expressions for parsing numeric values. For parsing locale numbers:

```
\regex_new:N \l_datatool_locale_numeric_regex
\regex_new:N \l_datatool_locale_fractional_regex
\regex_new:N \l_datatool_locale_fractional_nozero_regex
```

TeX's maximum integer limit is 0x7FFFFFFF so provide a regular expression to capture large integers that would otherwise trigger an error. (For example, a numeric ID or a bare telephone number without parentheses or hyphens could well exceed this value.)

```
\regex_new:N \l_datatool_locale_bigint_regex
```

For splitting standard decimals into number groups (dot for decimal separator and no number groups):

```
\regex_const:Nn \c_datatool_decimal_grps_regex
  { \A ([+\-])? 0* (\d{1,3}) \. (\d+) \Z }
\regex_const:Nn \c_datatool_decimal_grps_i_regex
  { \A ([+\-])? 0* (\d{1,3}) (\d{3}) \. (\d+) \Z }
\regex_const:Nn \c_datatool_decimal_grps_ii_regex
  { \A ([+\-])? 0* (\d{1,3}) (\d{3}) (\d{3}) \. (\d+) \Z }
\regex_const:Nn \c_datatool_decimal_grps_iii_regex
  { \A ([+\-])? 0* (\d+) (\d{3}) (\d{3}) (\d{3}) \. (\d+) \Z }
```

Similarly for integers:

```
\regex_const:Nn \c_datatool_integer_grps_i_regex
  { \A ([+\-])? 0* (\d{1,3}) (\d{3}) \Z }
\regex_const:Nn \c_datatool_integer_grps_ii_regex
  { \A ([+\-])? 0* (\d{1,3}) (\d{3}) (\d{3}) \Z }
\regex_const:Nn \c_datatool_integer_grps_iii_regex
  { \A ([+\-])? 0* (\d+) (\d{3}) (\d{3}) (\d{3}) \Z }
\regex_const:Nn \c_datatool_integer_no_grps_regex
  { \A ([+\-])? (\d{1,3}) \Z }
```

Trimming:

```
\regex_const:Nn \c_datatool_numeric_leading_zeros_regex
  { \A ([+\-])? 0* (\d+ (?: (\.\d+) )? ) \Z }
\regex_const:Nn \c_datatool_integer_leading_zeros_regex
  { \A ([+\-])? 0* (\d+ ) \Z }
\regex_const:Nn \c_datatool_decimal_redundant_zeros_regex
  { \A ([+\-])? 0* (\d+ \. \d+ ) 0* \Z }
\regex_const:Nn \c_datatool_decimal_implicit_zero_regex
  { \A ([+\-])? ( \. \d+ ) 0* \Z }
```

Any unformatted number:

```
\regex_const:Nn \c_datatool_any_numeric_regex
  { \A ([+\-])? \d* \. ? \d+ ( [eE] [+\-]? \d+ ) ? \Z }
```

Get fractional part:

```
\cs_new:Nn \datatool_decimal_frac:n
 {
   \__datatool_decimal_frac:w #1 . 0 . 0 \q_stop
 }
\cs_generate_variant:Nn \datatool_decimal_frac:n { e , o }
```

Internal command:

```
\cs_new:Npn \__datatool_decimal_frac:w #1.#2.#3 \q_stop
{
  \tl_if_head_eq_meaning:nNT { #1 } - { - }
  0 . #2
}
```

Right zero pad or truncate decimals.

```
\cs_new:Npn \__datatool_decimal_places_i:w #1.#2#3\q_stop
 {
   #1.#2
 }
\cs_new:Npn \__datatool_decimal_places_ii:w #1.#2#3#4\q_stop
 {
   #1.#2#3
 }
\cs_new:Npn \__datatool_decimal_places_iii:w #1.#2#3#4#5\q_stop
 {
   #1.#2#3#4
 }
\cs_new:Npn \__datatool_decimal_places_iv:w #1.#2#3#4#5#6\q_stop
 {
   #1.#2#3#4#5
 }
\cs_new:Npn \__datatool_decimal_places_v:w #1.#2#3#4#5#6#7\q_stop
 {
   #1.#2#3#4#5#6
 }
\cs_new:Npn \__datatool_decimal_places_vi:w #1.#2#3#4#5#6#7#8\q_stop
 {
```

```
    #1.#2#3#4#5#6#7
  }
 \cs_new:Npn \__datatool_decimal_places_vii:w #1.#2#3#4#5#6#7#8#9\q_stop
  {
    #1.#2#3#4#5#6#7#8
  }
```

NB First check that the number is a decimal. Expand to padded value:

```
 \cs_new:Nn \__datatool_decimal_trunc_pad_zeros:nn
  {
    \int_case:nnF { #2 }
     {
       { 1 }
         { \__datatool_decimal_places_i:w #1 0 \q_stop }
       { 2 }
         { \__datatool_decimal_places_ii:w #1 00 \q_stop }
       { 3 }
         { \__datatool_decimal_places_iii:w #1 000 \q_stop }
       { 4 }
         { \__datatool_decimal_places_iv:w #1 0000 \q_stop }
       { 5 }
         { \__datatool_decimal_places_v:w #1 00000 \q_stop }
       { 6 }
         { \__datatool_decimal_places_vi:w #1 000000 \q_stop }
     }
     {
       \__datatool_decimal_places_vii:w #1 0000000 \q_stop
     }
  }
```

\DTLsetnumberchars

\DTLsetnumberchars{⟨*number group char*⟩}{⟨*decimal char*⟩}

This sets the decimal character and number group characters.

```
 \NewDocumentCommand \DTLsetnumberchars { m m }
  {
    \datatool_set_numberchars:nn { #1 } { #2 }
  }

 \cs_new:Nn \datatool_set_numberchars:nn
  {
    \datatool_set_numberchars:nnnn { #1 } { #2 } { #1 } { #2 }
  }
 \cs_generate_variant:Nn \datatool_set_numberchars:nn
  { nV , Vn , VV }
```

Syntax: {⟨*format number group char*⟩} {⟨*format decimal char*⟩} {⟨*parse number group char*⟩} {⟨*parse decimal char*⟩}

```
 \cs_new:Nn \datatool_set_numberchars:nnnn
```

```
 {
  \tl_set:Nn \l__datatool_numbergroup_tl { #1 }
  \tl_set:Nn \l__datatool_numbergroup_parse_tl { #3 }
  \tl_set:Nn \l__datatool_decimal_tl { #2 }
  \tl_set:Nn \l__datatool_decimal_parse_tl { #4 }
```

Regular expressions used for parsing.

```
  \regex_set:Nn \l_datatool_locale_bigint_regex
    {
       \A [+\-]?\d* [2-9]
         \u{l__datatool_numbergroup_parse_tl}? \d{3}
         \u{l__datatool_numbergroup_parse_tl}? \d{3}
         \u{l__datatool_numbergroup_parse_tl}? \d{3}
          \Z
    }
  \regex_set:Nn \l_datatool_locale_numeric_regex
    {
       \A ([+\-]?\d+)
         (?:\u{l__datatool_numbergroup_parse_tl}(\d{3}))?
         (?:\u{l__datatool_numbergroup_parse_tl}(\d{3}))?
         (?:\u{l__datatool_numbergroup_parse_tl}(\d{3}))?
         (?:\u{l__datatool_decimal_parse_tl}(\d+))? \Z
    }
  \regex_set:Nn \l_datatool_locale_fractional_regex
    {
       \A ([+\-]?\d+) \u{l__datatool_decimal_parse_tl} (\d+) \Z
    }
  \regex_set:Nn \l_datatool_locale_fractional_nozero_regex
    {
       \A ([+\-]?) \u{l__datatool_decimal_parse_tl} (\d+) \Z
    }
}
\cs_generate_variant:Nn \datatool_set_numberchars:nnnn
 { VVVV , eeee , VnVn , nnVn }
```

Set the default.

```
\DTLsetnumberchars{,}{.}
```

As above but where a regular expression is needed for parsing.

```
\cs_new:Nn \datatool_set_numberchars_regex:nnnn
 {
  \tl_set:Nn \l__datatool_numbergroup_tl { #1 }
  \tl_clear:N \l__datatool_numbergroup_parse_tl
  \regex_set:Nn \l__datatool_numbergroup_parse_regex { #3 }
  \tl_set:Nn \l__datatool_decimal_tl { #2 }
  \tl_clear:N \l__datatool_decimal_parse_tl
  \regex_set:Nn \l__datatool_decimal_parse_regex { #4 }
```

Regular expressions used for parsing.

```
  \regex_set:Nn \l_datatool_locale_bigint_regex
    {
```

```
      \A [+\-]?\d* [2-9]
        \ur{l__datatool_numbergroup_parse_regex}? \d{3}
        \ur{l__datatool_numbergroup_parse_regex}? \d{3}
        \ur{l__datatool_numbergroup_parse_regex}? \d{3}
         \Z
    }
  \regex_set:Nn \l_datatool_locale_numeric_regex
    {
      \A ([+\-]?\d+)
        (?:\ur{l__datatool_numbergroup_parse_regex}(\d{3}))?
        (?:\ur{l__datatool_numbergroup_parse_regex}(\d{3}))?
        (?:\ur{l__datatool_numbergroup_parse_regex}(\d{3}))?
        (?:\ur{l__datatool_decimal_parse_regex}(\d+))? \Z
    }
  \regex_set:Nn \l_datatool_locale_fractional_regex
    {
      \A ([+\-]?\d+) \ur{l__datatool_decimal_parse_regex} (\d+) \Z
    }
  \regex_set:Nn \l_datatool_locale_fractional_nozero_regex
    {
      \A ([+\-]?) \ur{l__datatool_decimal_parse_regex} (\d+) \Z
    }
}
\cs_generate_variant:Nn \datatool_set_numberchars_regex:nnnn
 { VVnn , Vnnn , nVnn }
```

As above but regex is needed for number group and token list for decimal group when parsing.

```
\cs_new:Nn \datatool_set_numberchars_regex_tl:nnnn
 {
  \tl_set:Nn \l__datatool_numbergroup_tl { #1 }
  \tl_clear:N \l__datatool_numbergroup_parse_tl
  \regex_set:Nn \l__datatool_numbergroup_parse_regex { #3 }
  \tl_set:Nn \l__datatool_decimal_tl { #2 }
  \tl_set:Nn \l__datatool_decimal_parse_tl { #4 }
```

Regular expressions used for parsing.

```
  \regex_set:Nn \l_datatool_locale_bigint_regex
    {
      \A [+\-]?\d* [2-9]
        \ur{l__datatool_numbergroup_parse_regex}? \d{3}
        \ur{l__datatool_numbergroup_parse_regex}? \d{3}
        \ur{l__datatool_numbergroup_parse_regex}? \d{3}
         \Z
    }
  \regex_set:Nn \l_datatool_locale_numeric_regex
    {
      \A ([+\-]?\d+)
        (?:\ur{l__datatool_numbergroup_parse_regex}(\d{3}))?
        (?:\ur{l__datatool_numbergroup_parse_regex}(\d{3}))?
        (?:\ur{l__datatool_numbergroup_parse_regex}(\d{3}))?
```

```
          (?:\u{l__datatool_decimal_parse_tl}(\d+))? \Z
      }
    \regex_set:Nn \l_datatool_locale_fractional_regex
      {
         \A ([+\-]?\d+) \u{l__datatool_decimal_parse_tl} (\d+) \Z
      }
    \regex_set:Nn \l_datatool_locale_fractional_nozero_regex
      {
         \A ([+\-]?) \u{l__datatool_decimal_parse_tl} (\d+) \Z
      }
 }
 \cs_generate_variant:Nn \datatool_set_numberchars_regex_tl:nnnn
  { VVnn , Vnnn , nVnn , nVnV , nnnV }
```

As above but token list is needed for number group and regex for decimal group when parsing.

```
 \cs_new:Nn \datatool_set_numberchars_tl_regex:nnnn
  {
   \tl_set:Nn \l__datatool_numbergroup_tl { #1 }
   \tl_set:Nn \l__datatool_numbergroup_parse_tl { #3 }
   \tl_set:Nn \l__datatool_decimal_tl { #2 }
   \tl_clear:N \l__datatool_decimal_parse_tl
   \regex_set:Nn \l__datatool_decimal_parse_regex { #4 }
```

Regular expressions used for parsing.

```
   \regex_set:Nn \l_datatool_locale_bigint_regex
     {
        \A [+\-]?\d* [2-9]
          \u{l__datatool_numbergroup_parse_tl}? \d{3}
          \u{l__datatool_numbergroup_parse_tl}? \d{3}
          \u{l__datatool_numbergroup_parse_tl}? \d{3}
           \Z
     }
   \regex_set:Nn \l_datatool_locale_numeric_regex
     {
        \A ([+\-]?\d+)
          (?:\u{l__datatool_numbergroup_parse_tl}(\d{3}))?
          (?:\u{l__datatool_numbergroup_parse_tl}(\d{3}))?
          (?:\u{l__datatool_numbergroup_parse_tl}(\d{3}))?
          (?:\ur{l__datatool_decimal_parse_regex}(\d+))? \Z
     }
   \regex_set:Nn \l_datatool_locale_fractional_regex
     {
        \A ([+\-]?\d+) \ur{l__datatool_decimal_parse_regex} (\d+) \Z
     }
   \regex_set:Nn \l_datatool_locale_fractional_nozero_regex
     {
        \A ([+\-]?) \ur{l__datatool_decimal_parse_regex} (\d+) \Z
     }
 }
 \cs_generate_variant:Nn \datatool_set_numberchars_tl_regex:nnnn
```

```
 { VVnn , Vnnn , nVnn , VnVn , nnVn }
```

Convenient shortcut to set thin-space number group:

```
\datatool_if_unicode_engine:TF
 {
  \cs_new:Nn \datatool_set_thinspace_group_decimal_char:n
   {
     \datatool_set_numberchars_regex_tl:nnnn
       { \, } { #1 } { \c{,} | \s | \x{2009} } { #1 }
   }
 }
 {
  \cs_new:Nn \datatool_set_thinspace_group_decimal_char:n
   {
     \datatool_set_numberchars_regex_tl:nnnn
       { \, } { #1 } { \c{,} | \s | \x{E2} \x{80} \x{89} } { #1 }
   }
 }
\cs_generate_variant:Nn \datatool_set_thinspace_group_decimal_char:n { V }
```

Similarly for apostrophe:

```
\datatool_if_unicode_engine:TF
 {
  \regex_const:Nn \c_datatool_apostrophe_regex
    { \x{27} | \x{2019} }
 }
 {
  \regex_const:Nn \c_datatool_apostrophe_regex
    { \x{27} | \x{E2} \x{80} \x{99} }
 }
\cs_new:Nn \datatool_set_apos_group_decimal_char:n
 {
   \datatool_set_numberchars_regex_tl:nnnn
     { ' } { #1 } { \ur{c_datatool_apostrophe_regex} } { #1 }
 }
\cs_generate_variant:Nn \datatool_set_apos_group_decimal_char:n { V }
```

Similarly for underscore number group character.

```
\cs_new:Nn \datatool_set_underscore_group_decimal_char:n
 {
   \datatool_set_numberchars_regex_tl:nnnn
    { \_ } { #1 } { \c{_} | \x{5F} } { #1 }
 }
\cs_generate_variant:Nn \datatool_set_underscore_group_decimal_char:n { V }
```

2.4.1 Determining Data Types

The control sequence `\__datatool_datum:nnnn` parses its argument to determine the data type of its argument. Each data type is represented by one of the following values: 0 (string), 1 (integer), 2 (decimal), 3 (currency), 4 (datetime), 5 (date), or 6

(time). The unknown data type -1 is used when the data is empty. Numeric values are expected to use the number group and decimal characters currently in effect.

The temporal data types (datetime, date and time) are new to v3.0 and have a numeric value. This is mainly provided for the benefit of datatooltk as imported data typically treats date/time data as numeric and it may be useful to retain the numeric value and ISO format. The Julian day number (an integer) is used to represent a date, the Julian time (a decimal between -0.5 an 0.5, where 0 is midday) is used to represent a time, and the Julian date (the sum of the Julian day number and the Julian time) is used to represent a timestamp (datetime).

`\@dtl@datatype` Scratch variable used to represent the data type identifier. This may be replaced with an integer variable at some point.

```
\newcount\@dtl@datatype
```

Data type identifiers.

```
\int_const:Nn \c_datatool_unknown_int {-1}
\int_const:Nn \c_datatool_string_int {0}
\int_const:Nn \c_datatool_integer_int {1}
\int_const:Nn \c_datatool_decimal_int {2}
\int_const:Nn \c_datatool_currency_int {3}
\int_const:Nn \c_datatool_datetime_int {4}
\int_const:Nn \c_datatool_date_int {5}
\int_const:Nn \c_datatool_time_int {6}
```

Maximum known value. This will be updated if any new types are added in future.

```
\cs_new:Nn \datatool_max_known_type:
 {
   \c_datatool_time_int
 }
```

`\DTLgetDataTypeName` Get textual name for data type (expandable):

```
\newcommand*{\DTLgetDataTypeName}[1]
{
  \int_case:nnF { #1 }
   {
     { \c_datatool_unknown_int }
       { \DTLdatatypeunsetname }
     { \c_datatool_string_int }
       { \DTLdatatypestringname }
     { \c_datatool_integer_int }
       { \DTLdatatypeintegername }
     { \c_datatool_decimal_int }
       { \DTLdatatypedecimalname }
     { \c_datatool_currency_int }
       { \DTLdatatypecurrencyname }
     { \c_datatool_datetime_int }
       { \DTLdatatypedatetimename }
     { \c_datatool_date_int }
       { \DTLdatatypedatename }
```

```
      { \c_datatool_time_int }
        { \DTLdatatypetimename }
    }
    {
      \DTLdatatypeinvalidname
    }
}

\DTLdatatypeunsetname
\newcommand \DTLdatatypeunsetname { unset }

\DTLdatatypestringname
\newcommand \DTLdatatypestringname { string }

\DTLdatatypeintegername
\newcommand \DTLdatatypeintegername { integer }

\DTLdatatypedecimalname
\newcommand \DTLdatatypedecimalname { decimal }

\DTLdatatypecurrencyname
\newcommand \DTLdatatypecurrencyname { currency }

\DTLdatatypedatetimename
\newcommand \DTLdatatypedatetimename { date-time }

\DTLdatatypedatename
\newcommand \DTLdatatypedatename { date }

\DTLdatatypetimename
\newcommand \DTLdatatypetimename { time }

\DTLdatatypeinvalidname
\newcommand \DTLdatatypeinvalidname { invalid }
```

Test for for datum types. The argument should be a number. Test if the number is a recognised type (including “unknown”):

```
\prg_new_conditional:Npnn \datatool_if_valid_datum_type:n #1
   { p, T, F, TF }
 {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn { #1 } < { \c_datatool_unknown_int } }
      { \int_compare_p:nNn { #1 } > { \datatool_max_known_type: } }
    { \prg_return_false: }
    { \prg_return_true: }
 }
```

Test if the argument is a numeric datum type ID (currency, dates and times are considered numeric, in addition to integers and decimals):

```
\prg_new_conditional:Npnn \datatool_if_numeric_datum_type:n #1
   { p, T, F, TF }
 {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn { #1 } < { \c_datatool_integer_int } }
      { \int_compare_p:nNn { #1 } > { \c_datatool_time_int } }
    { \prg_return_false: }
    { \prg_return_true: }
 }
```

Test if the argument is a temporal datum type ID (timestamp, date or time):

```
\prg_new_conditional:Npnn \datatool_if_temporal_datum_type:n #1
   { p, T, F, TF }
 {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn { #1 } < { \c_datatool_datetime_int } }
      { \int_compare_p:nNn { #1 } > { \c_datatool_time_int } }
    { \prg_return_false: }
    { \prg_return_true: }
 }
```

Test if the argument is an integer or decimal datum type ID (not currency or temporal):

```
\prg_new_conditional:Npnn \datatool_if_number_only_datum_type:n #1
   { p, T, F, TF }
 {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn { #1 } = { \c_datatool_integer_int } }
      { \int_compare_p:nNn { #1 } = { \c_datatool_decimal_int } }
    { \prg_return_true: }
    { \prg_return_false: }
 }
```

Test if the argument is any datum type that has an integer value (integer or date):

```
\prg_new_conditional:Npnn \datatool_if_any_int_datum_type:n #1
   { p, T, F, TF }
 {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn { #1 } = { \c_datatool_integer_int } }
      { \int_compare_p:nNn { #1 } = { \c_datatool_date_int } }
    { \prg_return_true: }
    { \prg_return_false: }
 }
```

The null commands were originally in datatool but as from v3.0, the lower-level commands are now in datatool-base since commands like `\DTLdatumvalue` need to check for null. The person package, which may be used with just datatool-base and not datatool, performs a null test.

`\@dtlnovalue` The old internal command `\@dtlnovalue` has been replaced with the constant `\c_datatool_nullvalue_tl`. This is similar in concept to `c_novalue_tl`

but is intended to represent missing values in datatool return values, particularly when querying database values. This starts with a space in the event that a null value is expanded in a context that tests for an initial letter without checking the category code. It ends with a space that has a letter category code. This mixture of category codes is designed to make it harder to accidentally identify the string " Undefined Value " as null.

Note that neither the old internal command nor this new constant are intended to be included in a list of data or in a database. DatatoolTk does still use the old command name as a marker for its own internal use, although it shouldn't make its way into an output file.

```
\tl_const:Nx \c_datatool_nullvalue_tl
 {
   \c_catcode_other_space_tl
   \tl_to_str:n { Undefined }
   \c_catcode_other_space_tl
   \tl_to_str:n { Value }
   \char_generate:nn { 32 } { 11 }
 }
```

`\dtlnovalue`

```
\newcommand* \dtlnovalue { \c_datatool_nullvalue_tl }
```

`\DTLstringnull` String null value:

```
\newcommand* \DTLstringnull { \@dtlstringnull }
```

`\@dtlstringnull` String null value:

```
\newcommand* \@dtlstringnull {NULL}
```

`\DTLnumbernull` Number null value:

```
\newcommand* \DTLnumbernull { \@dtlnumbernull }
```

`\@dtlnumbernull` Number null value:

```
\newcommand*{\@dtlnumbernull}{0}
```

Constant representing an empty datum value with unknown type.

```
\tl_const:Nn \c_datatool_empty_datum_tl
 {
   \__datatool_datum:nnnn { } { } { } { \c_datatool_unknown_int }
 }
```

`\DTLifnull` is still defined in datatool. Provide low-level commands. Test token for null. NB this doesn't test against the expansion text of the string and number null values as they could easily be valid non-null values, but it will test against the actual null constant value.

```
\prg_new_conditional:Npnn \datatool_if_null:N #1
  { p, T, F, TF }
{
  \bool_lazy_any:nTF
```

```
    {
      { \tl_if_eq_p:NN #1 \dtlnovalue }
      { \tl_if_eq_p:NN #1 \DTLstringnull }
      { \tl_if_eq_p:NN #1 \DTLnumbernull }
      { \tl_if_eq_p:NN #1 \c_datatool_nullvalue_tl }
    }
    { \prg_return_true: }
    { \prg_return_false: }
}
\cs_generate_variant:Nn \datatool_if_null:NTF { cTF }
```

Test for null where the argument may be a single token or may be a token list. If the argument is a single token, then the test is the same as the above otherwise the result will be false.

```
\prg_new_conditional:Npnn \datatool_if_null:n #1
  { T, F, TF }
{
  \tl_if_single_token:nTF { #1 }
  {
    \bool_lazy_any:nTF
     {
       { \tl_if_eq_p:NN #1 \dtlnovalue }
       { \tl_if_eq_p:NN #1 \DTLstringnull }
       { \tl_if_eq_p:NN #1 \DTLnumbernull }
       { \tl_if_eq_p:NN #1 \c_datatool_nullvalue_tl }
     }
     { \prg_return_true: }
     {
       \tl_if_eq:NnTF \dtlnovalue { #1 }
       { \prg_return_true: }
       {
         \tl_if_eq:NnTF \DTLstringnull { #1 }
         { \prg_return_true: }
         {
           \tl_if_eq:NnTF \DTLnumbernull { #1 }
           { \prg_return_true: }
           { \prg_return_false: }
         }
       }
     }
  }
  {
    \tl_if_eq:NnTF \c_datatool_nullvalue_tl { #1 }
     { \prg_return_true: }
     { \prg_return_false: }
  }
}
```

Test for null or empty. This includes testing for the empty datum.

```
\prg_new_conditional:Npnn \datatool_if_null_or_empty:N #1
```

```
  { p, T, F, TF }
{
  \bool_lazy_any:nTF
   {
     { \tl_if_empty_p:N #1 }
     { \tl_if_eq_p:NN #1 \dtlnovalue }
     { \tl_if_eq_p:NN #1 \DTLstringnull }
     { \tl_if_eq_p:NN #1 \DTLnumbernull }
     { \tl_if_eq_p:NN #1 \c_datatool_nullvalue_tl }
     { \tl_if_eq_p:NN #1 \c_datatool_empty_datum_tl }
   }
   { \prg_return_true: }
   { \prg_return_false: }
}
```

Test for null or empty where the argument may be a single token or may be a token list. If the argument is a single token, then the test is the same as the above. If the argument isn't a single token the result will be false unless the argument is empty.

```
\prg_new_conditional:Npnn \datatool_if_null_or_empty:n #1
  { T, F, TF }
{
  \tl_if_single_token:nTF { #1 }
  {
    \bool_lazy_any:nTF
     {
       { \tl_if_empty_p:N #1 }
       { \tl_if_eq_p:NN #1 \dtlnovalue }
       { \tl_if_eq_p:NN #1 \DTLstringnull }
       { \tl_if_eq_p:NN #1 \DTLnumbernull }
       { \tl_if_eq_p:NN #1 \c_datatool_nullvalue_tl }
       { \tl_if_eq_p:NN #1 \c_datatool_empty_datum_tl }
     }
     { \prg_return_true: }
     {
       \tl_if_eq:NnTF \dtlnovalue { #1 }
       { \prg_return_true: }
       {
         \tl_if_eq:NnTF \DTLstringnull { #1 }
         { \prg_return_true: }
         {
           \tl_if_eq:NnTF \DTLnumbernull { #1 }
           { \prg_return_true: }
           { \prg_return_false: }
         }
       }
     }
  }
  {
    \tl_if_empty:nTF { #1 }
    { \prg_return_true: }
```

```
    {
      \tl_if_eq:NnTF \c_datatool_nullvalue_tl { #1 }
       { \prg_return_true: }
       {
         \tl_if_eq:NnTF \c_datatool_empty_datum_tl { #1 }
         { \prg_return_true: }
         { \prg_return_false: }
       }
    }
  }
}
```

Datum marker. The first argument is the original content. The second should expand to a (non-locale) numeric representation or empty for non-numeric data. (The second argument may contain additional content that will ordinarily be discarded on expansion, but may be retrieved if required.) The third argument is the currency symbol or empty if not currency. The final argument is the integer data type identifier.

```
\cs_new:Nn \__datatool_datum:nnnn { \exp_not:n { #1 } }
```

This means that the datum control sequence content consists of five things: the datum marker and its four arguments.

The weird datum is designed to allow easy lookup in databases. It uses a similar trick to etoolbox that uses the bar | character with category code 3 as a marker.

```
\group_begin:
```

Locally change category code of |

```
  \char_set_catcode_math_toggle:N \|
```

Create a constant token list that expands to this character with this category code.

```
  \tl_const:Nn \c__datatool_datum_weird_marker_tl { | }
```

Define the weird function to expand to the regular datum function. This is designed for expandable contexts to perform a quick conversion without expanding the arguments.

```
  \cs_new:Npn \__datatool_datum:w | #1 | #2 | #3 | #4 |
   {
     \exp_not:N \__datatool_datum:nnnn
      { \exp_not:n { #1 } }
      { \exp_not:n { #2 } }
      { \exp_not:n { #3 } }
      { \exp_not:n { #4 } }
   }
```

Allow marker to expand.

```
  \cs_new:Npn \__datatool_datum_use:w | #1 | #2 | #3 | #4 |
   {
     \__datatool_datum:nnnn
       { #1 } { #2 } { #3 } { #4 }
   }
```

Only the first argument (but don't expand that):

```
  \cs_new:Npn \__datatool_datum_use_i:w | #1 | #2 | #3 | #4 |
   {
```

```
    \exp_not:n { #1 }
  }
```
Define the reverse. Four arguments provided (datum marker not included). This only prevents the string and currency arguments from expanding.
```
  \cs_new:Nn \__datatool_weird_datum:nnnn
   {
     \exp_not:N \__datatool_datum:w
       | \exp_not:n { #1 }
       | #2
       | \exp_not:n { #3 }
       | #4 |
  }
```
Five arguments provided, starting with the datum marker that needs to be discarded.
```
  \cs_new:Nn \__datatool_weird_datum:Nnnnn
   {
     \exp_not:N \__datatool_datum:w
       | \exp_not:n { #2 }
       | \exp_not:n { #3 }
       | \exp_not:n { #4 }
       | \exp_not:n { #5 } |
  }
```
Parse content that starts with `\__datatool_weird_datum:w`.
```
 \cs_new:Npn \__datatool_get_weird_datum:w
   \__datatool_datum:w | #1 | #2 | #3 | #4 | #5 \q_stop
 {
    \quark_if_nil:nTF { #5 }
    {
      \@dtl@datatype = #4
      \tl_set:Nn \l__datatool_datum_original_value_tl { #1 }
      \tl_set:Nn \l__datatool_datum_value_tl { #2 }
      \tl_set:Nn \l__datatool_datum_currency_tl { #3 }
    }
    {
      \@dtl@datatype = \c_datatool_string_int
    }
 }
```
Restore category code.
```
 \group_end:
```
Test for equality where one or other value might be in either datum format.
Syntax: ⟨*tl var1*⟩ ⟨*tl var2*⟩ {⟨*true*⟩}{⟨*false*⟩}
```
 \prg_new_conditional:Npnn \datatool_if_value_eq:NN #1 #2
    { T, F, TF }
 {
    \exp_args:NV \tl_if_head_eq_meaning:nNTF
       #1 \__datatool_datum:w
     {
       \exp_args:NV \tl_if_head_eq_meaning:nNTF
```

```
        #2 \__datatool_datum:w
      {
        \datatool_if_value_eq:eeTF
         { #1 } { #2 }
         { \prg_return_true: }
         { \prg_return_false: }
      }
      {
        \datatool_if_value_eq:eVTF
         { #1 } #2
         { \prg_return_true: }
         { \prg_return_false: }
      }
    }
    {
      \exp_args:NV \tl_if_head_eq_meaning:nNTF
        #2 \__datatool_datum:w
      {
        \datatool_if_value_eq:VeTF
          #1 { #2 }
         { \prg_return_true: }
         { \prg_return_false: }
      }
      {
        \datatool_if_value_eq:VVTF #1 #2
         { \prg_return_true: }
         { \prg_return_false: }
      }
    }
  }
```

Syntax: ⟨*tl var*⟩ {⟨*tl*⟩}{⟨*true*⟩}{⟨*false*⟩}

```
 \prg_new_conditional:Npnn \datatool_if_value_eq:Nn #1 #2
    { T, F, TF }
  {
    \exp_args:NV \tl_if_head_eq_meaning:nNTF
       #1 \__datatool_datum:w
     {
       \datatool_if_value_eq:enTF
        { #1 } { #2 }
        { \prg_return_true: }
        { \prg_return_false: }
     }
     {
       \datatool_if_value_eq:VnTF #1 { #2 }
        { \prg_return_true: }
        { \prg_return_false: }
     }
  }
```

Syntax: {⟨*tl*⟩} ⟨*tl var*⟩ {⟨*true*⟩}{⟨*false*⟩}

```
\prg_new_conditional:Npnn \datatool_if_value_eq:nN #1 #2
   { T, F, TF }
 {
   \exp_args:NV \tl_if_head_eq_meaning:nNTF
      #2 \__datatool_datum:w
    {
      \datatool_if_value_eq:neTF
       { #1 } { #2 }
       { \prg_return_true: }
       { \prg_return_false: }
    }
    {
      \datatool_if_value_eq:nVTF { #1 } #2
       { \prg_return_true: }
       { \prg_return_false: }
    }
 }
```

Syntax: {⟨*tl1*⟩}{⟨*tl2*⟩}{⟨*true*⟩}{⟨*false*⟩}

```
\prg_new_conditional:Npnn \datatool_if_value_eq:nn #1 #2
   { T, F, TF }
 {
   \tl_if_eq:nnTF { #1 } { #2 }
    { \prg_return_true: }
    {
      \tl_if_head_eq_meaning:nNTF { #1 } \__datatool_datum:nnnn
       {
         \tl_if_head_eq_meaning:nNTF { #2 } \__datatool_datum:nnnn
          {
```

Both in datum format. If both numeric use numeric comparison, otherwise use string comparison.

```
              \bool_lazy_and:nnTF
               {
                 \int_compare_p:nNn
                   { \__datatool_datum_type:n { #1 } }
                     >
                   { \c_datatool_string_int }
               }
               {
                 \int_compare_p:nNn
                   { \__datatool_datum_type:n { #2 } }
                     >
                   { \c_datatool_string_int }
               }
               {
```

Both numeric. Test both the value and currency symbol.

```
                 \tl_if_eq:eeTF
                  { \__datatool_datum_currency:n { #1 } }
```

```
                { \__datatool_datum_currency:n { #2 } }
                {
                  \fp_compare:nNnTF
                    { \__datatool_datum_value:n { #1 } }
                    =
                    { \__datatool_datum_value:n { #2 } }
                   { \prg_return_true: }
                   { \prg_return_false: }
                }
                { \prg_return_false: }
              }
              {
```

One or other isn't numeric. Compare string values only.

```
                \tl_if_eq:eeTF
                  { \__datatool_datum_string:n { #1 } }
                  { \__datatool_datum_string:n { #2 } }
                 { \prg_return_true: }
                 { \prg_return_false: }
              }
            }
            {
```

First in datum format, second isn't. Compare string values only

```
              \tl_if_eq:enTF { #1 } { #2 }
               { \prg_return_true: }
               { \prg_return_false: }
            }
          }
          {
```

First isn't in datum format. Compare string values only

```
            \tl_if_eq:enTF { #1 } { #2 }
             { \prg_return_true: }
             { \prg_return_false: }
          }
      }
  }
\cs_generate_variant:Nn \datatool_if_value_eq:nnTF
 { VVTF , VnTF , nVTF, eeTF, neTF, enTF, eVTF, VeTF }
```

Function to convert a token list variable so that it contains a weird datum function if the original contained a datum marker but don't parse and convert otherwise.

```
\cs_new:Nn \__datatool_to_weird_datum_no_parse:N
{
  \exp_args:NV \__datatool_to_weird_datum_no_parse:nN #1 #1
}
\cs_new:Nn \__datatool_to_weird_datum_no_parse:nN
{
  \tl_if_head_eq_meaning:nNTF
    { #1 } \__datatool_datum:nnnn
```

```
    {
      \tl_set:Nx #2 { \__datatool_weird_datum:Nnnnn #1 }
    }
    {
      \tl_set:Nn #2 { #1 }
    }
}
```

Function to convert a token list variable so that it contains a weird datum function.

```
\cs_new:Nn \__datatool_to_weird_datum:N
{
  \exp_args:NV \__datatool_to_weird_datum:nN #1 #1
}
\cs_new:Nn \__datatool_to_weird_datum:nN
{
  \tl_if_head_eq_meaning:nNTF
     { #1 } \__datatool_datum:w
    {
```

Already in weird format.

```
      \tl_set:Nn #2 { #1 }
    }
    {
```

Convert to datum.

```
      \__datatool_to_datum:nN { #1 } #2
```

Leave null unprocessed.

```
      \datatool_if_null:NF #2
       {
```

At this point the token list variable should start with the datum marker but check to make sure.

```
         \tl_if_head_eq_meaning:nNT
           { #1 } \__datatool_datum:nnnn
          {
            \tl_put_left:Nn #2 { \__datatool_weird_datum:Nnnnn }
            \tl_set:Nx #2 { #2 }
          }
       }
  }
}
```

Function to convert a token list variable so that it contains a normal datum function.

```
\cs_new:Nn \__datatool_to_datum:N
 {
   \exp_args:NV \__datatool_to_datum:nN #1 #1
 }
```

Function to convert a token list so that it contains a normal datum function and save it in the token list variable.

```
\cs_new:Nn \__datatool_to_datum:nN
```

```
 {
  \datatool_if_null:nTF { #1 }
   {
```
Set variable to null.
```
     \tl_set_eq:NN #2 \dtlnovalue
   }
   {
    \tl_if_head_eq_meaning:nNTF
       { #1 } \__datatool_datum:w
     {
```
Convert from weird format:
```
       \tl_set:Nx #2 { #1 }
     }
     {
       \tl_if_head_eq_meaning:nNTF
         { #1 } \__datatool_datum:nnnn
        {
```
Already in the correct format:
```
          \tl_set:Nn #2 { #1 }
        }
        {
```
Needs parsing.
```
          \__datatool_parse_datum:n { #1 }
          \tl_if_empty:NTF \l__datatool_datum_update_value_tl
           {
             \int_compare:nNnTF
               { \@dtl@datatype }
                 =
               { \c_datatool_unknown_int }
             {
```
NB this may be blank (only a space) not empty, but it's still classed as unknown.
```
               \tl_set:Nn #2
                {
                  \__datatool_datum:nnnn { #1 } { } { }
                  { \c_datatool_unknown_int }
                }
             }
             {
               \tl_set_eq:NN #2 \l__datatool_datum_original_value_tl
             }
          }
          {
            \tl_set_eq:NN #2 \l__datatool_datum_update_value_tl
          }
       }
     }
   }
```

```
}
```
Similar but don't parse. Only convert if it starts with weird form.
```
\cs_new:Nn \__datatool_rm_weird_datum:N
{
  \exp_args:NV \__datatool_rm_weird_datum:nN #1 #1
}
\cs_new:Nn \__datatool_rm_weird_datum:nN
{
 \tl_if_head_eq_meaning:nNTF
    { #1 } \__datatool_datum:w
  {
```
Convert from weird format:
```
    \tl_set:Nx #2 { #1 }
  }
  {
    \tl_set:Nn #2 { #1 }
  }
}
```
Remove either datum.
```
\cs_new:Nn \__datatool_rm_datum:N
{
  \exp_args:NV \__datatool_rm_datum:nN #1 #1
}
\cs_new:Nn \__datatool_rm_datum:nN
{
 \__datatool_rm_weird_datum:nN { #1 } #2
 \exp_args:NV \tl_if_head_eq_meaning:nNT
    #2 \__datatool_datum:nnnn
  {
```
Convert from datum format:
```
    \tl_set:Nx #2 { #2 }
  }
}
```
Used to encapsulate floating point datum value. The first argument is the (non-locale) original floating point value. The second is the l3fp content to allow for reconstruction. The third is the decimal value for use where scientific notation can't be parsed. For example, with fp functions.
```
\tl_if_eq:NnTF \@dtl@mathprocessor { fp }
 {
   \cs_new:Nn \datatool_datum_fp:nnn { #3 }
 }
 {
   \cs_new:Nn \datatool_datum_fp:nnn { #1 }
 }
```
Pick out the l3fp content:
```
\cs_new:Nn \datatool_datum_fp:Nnnn { \exp_not:n { #3 } }
```

```
\cs_new:Nn \datatool_set_fp:Nn
 {
  \tl_if_single_token:nTF { #2 }
   {
     \exp_args:No \tl_if_head_eq_meaning:nNTF { #2 } \__datatool_datum:nnnn
      {
        \exp_last_unbraced:NV \__datatool_get_datum:w #2 \q_nil \q_stop
      }
      {
        \exp_args:No \tl_if_head_eq_meaning:nNTF { #2 }  \__datatool_datum:w
         {
           \exp_last_unbraced:NV \__datatool_get_weird_datum:w #2 \q_nil \q_stop
         }
         {
           \exp_args:No \__datatool_parse_datum:n { #2 }
         }
      }
   }
   {
      \tl_if_head_eq_meaning:nNTF { #2 } \__datatool_datum:nnnn
      {
        \__datatool_get_datum:w #2 \q_nil \q_stop
      }
      {
        \tl_if_head_eq_meaning:nNTF { #2 } \__datatool_datum:w
         {
           \__datatool_get_weird_datum:w #2 \q_nil \q_stop
         }
         {
           \__datatool_parse_datum:n { #2 }
         }
      }
   }
  \datatool_if_numeric_datum_type:nF { \@dtl@datatype }
   {
     \exp_args:Nx \__datatool_parse_datum:n
      { \tl_trim_spaces:e { \text_expand:n { #2 } } }
   }
  \tl_if_empty:NTF \l__datatool_datum_value_tl
   {
     \fp_zero:N #1
   }
   {
     \exp_args:NV \tl_if_head_eq_meaning:nNTF
       \l__datatool_datum_value_tl
      { \datatool_datum_fp:nnn }
      {
        \tl_set:Nx #1
         {
           \exp_after:wN \datatool_datum_fp:Nnnn \l__datatool_datum_value_tl
```

```
        }
      }
      {
        \fp_set:Nn #1 { \l__datatool_datum_value_tl }
      }
    }
  }
\cs_generate_variant:Nn \datatool_set_fp:Nn { NV, No }
```

\DTLusedatum

\DTLusedatum{⟨*tl*⟩}

```
\cs_new:Nn \__datatool_datum_string:n { \datatool_datum_string:Nnnnn #1 }
\newcommand{\DTLusedatum} [1]
 {
   \tl_if_eq:NNTF #1 \dtlnovalue
   { \dtlnovalue }
   { \exp_args:NV \__datatool_datum_string:n #1 }
 }
```

At the time of writing there were no `\use...` commands for five arguments. There now are, but these are provided anyway in case of a slightly older kernel and they are better semantically.

```
\cs_new:Nn \datatool_datum_string:Nnnnn { #2 }
```

\DTLdatumvalue

\DTLdatumvalue{⟨*tl*⟩}

```
\cs_new:Nn \__datatool_datum_value:n { \datatool_datum_value:Nnnnn #1 }
\newcommand{\DTLdatumvalue} [1]
 {
   \tl_if_eq:NNTF #1 \dtlnovalue
   { \DTLnumbernull }
   { \exp_args:NV \__datatool_datum_value:n #1 }
 }
```

```
\cs_new:Nn \datatool_datum_value:Nnnnn { #3 }
```

\DTLdatumcurrency

\DTLdatumcurrency{⟨*tl*⟩}

```
\cs_new:Nn \__datatool_datum_currency:n { \datatool_datum_currency:Nnnnn #1 }
\newcommand{\DTLdatumcurrency} [1]
 {
   \tl_if_eq:NNTF #1 \dtlnovalue
```

```
    { \dtlnovalue }
    { \exp_args:NV \__datatool_datum_currency:n #1 }
  }

\cs_new:Nn \datatool_datum_currency:Nnnnn { \exp_not:n { #4 } }
```

\DTLdatumtype

\DTLdatumtype{⟨*tl*⟩}

```
\cs_new:Nn \__datatool_datum_type:n { \datatool_datum_type:Nnnnn #1 }
\newcommand{\DTLdatumtype} [1]
 {
   \tl_if_eq:NNTF #1 \dtlnovalue
   { \int_use:N \c_datatool_unknown_int }
   { \int_eval:n { \exp_args:NV \__datatool_datum_type:n #1 } }
 }

\cs_new:Nn \datatool_datum_type:Nnnnn { #5 }
```

Debugging:

```
\cs_new:Nn \datatool_datum_show:N
 {
  \int_compare:nNnTF { \tl_count:N #1 } = { 5 }
   {
     \exp_after:wN \__datatool_datum_show:NNnnnn
      \exp_after:wN #1 #1
   }
   {
    \datatool_if_null:NTF #1
     {
       \msg_show:nnnV
        { datatool-base }
        { show-datum-var-null }
        { #1 }
        #1
     }
     {
       \tl_if_empty:NTF #1
        {
          \msg_show:nnn
           { datatool-base }
           { show-datum-var-empty }
           { #1 }
        }
        {
         \PackageError { datatool-base }
          { \tl_to_str:N #1 \c_space_tl is ~ not ~ a ~ datum ~ variable }
          { }
        }
```

```
    }
  }
}
\cs_new:Nn \__datatool_datum_show:NNnnnn
{
 \tl_if_eq:NNTF #2 \__datatool_datum:nnnn
  {
    \int_case:nnF { #6 }
     {
       { \c_datatool_unknown_int }
        {
          \msg_show:nnnn
            { datatool-base }
            { show-datum-var-unset }
            { #1 }
            { #3 }
        }
       { \c_datatool_string_int }
        {
          \msg_show:nnnn
            { datatool-base }
            { show-datum-var-string }
            { #1 }
            { #3 }
        }
       { \c_datatool_integer_int }
        {
          \msg_show:nnnnn
            { datatool-base }
            { show-datum-var-integer }
            { #1 }
            { #3 }
            { #4 }
        }
       { \c_datatool_decimal_int }
        {
          \msg_show:nnnnn
            { datatool-base }
            { show-datum-var-decimal }
            { #1 }
            { #3 }
            { #4 }
        }
       { \c_datatool_currency_int }
        {
          \msg_show:nnnnnn
            { datatool-base }
            { show-datum-var-currency }
            { #1 }
```

```
            { #3 }
            { #4 }
            { #5 }
        }
      { \c_datatool_datetime_int }
        {
          \msg_show:nnnnn
            { datatool-base }
            { show-datum-var-datetime }
            { #1 }
            { #3 }
            { #4 }
        }
      { \c_datatool_date_int }
        {
          \msg_show:nnnnn
            { datatool-base }
            { show-datum-var-date }
            { #1 }
            { #3 }
            { #4 }
        }
      { \c_datatool_time_int }
        {
          \msg_show:nnnnn
            { datatool-base }
            { show-datum-var-time }
            { #1 }
            { #3 }
            { #4 }
        }
    }
    {
      \PackageError { datatool-base }
        {
          datum ~ variable ~
          \tl_to_str:N #1 \c_space_tl has ~ invalid ~ data ~ type: ~
          \int_eval:n { #6 }
        }
        { }
    }
}
{
 \tl_if_eq:NNTF #2  \__datatool_datum:w
    {
      \int_case:nnF { #6 }
        {
          { \c_datatool_unknown_int }
           {
             \msg_show:nnnn
```

```
      { datatool-base }
      { show-weird-datum-var-unset }
      { #1 }
      { #3 }
  }
{ \c_datatool_string_int }
  {
    \msg_show:nnnn
      { datatool-base }
      { show-weird-datum-var-string }
      { #1 }
      { #3 }
  }
{ \c_datatool_integer_int }
  {
    \msg_show:nnnnn
      { datatool-base }
      { show-weird-datum-var-integer }
      { #1 }
      { #3 }
      { #4 }
  }
{ \c_datatool_decimal_int }
  {
    \msg_show:nnnnn
      { datatool-base }
      { show-weird-datum-var-decimal }
      { #1 }
      { #3 }
      { #4 }
  }
{ \c_datatool_currency_int }
  {
    \msg_show:nnnnnn
      { datatool-base }
      { show-weird-datum-var-currency }
      { #1 }
      { #3 }
      { #4 }
      { #5 }
  }
{ \c_datatool_datetime_int }
  {
    \msg_show:nnnnn
      { datatool-base }
      { show-weird-datum-var-datetime }
      { #1 }
      { #3 }
      { #4 }
  }
```

```
            { \c_datatool_date_int }
             {
               \msg_show:nnnnn
                 { datatool-base }
                 { show-weird-datum-var-date }
                 { #1 }
                 { #3 }
                 { #4 }
             }
            { \c_datatool_time_int }
             {
               \msg_show:nnnnn
                 { datatool-base }
                 { show-weird-datum-var-time }
                 { #1 }
                 { #3 }
                 { #4 }
             }
          }
          {
            \PackageError { datatool-base }
             {
               weird ~ datum ~ variable ~
               \tl_to_str:N #1 \c_space_tl has ~ invalid ~ data ~ type: ~
               \int_eval:n { #6 }
             }
             { }
          }
       }
       {
         \PackageError { datatool-base }
          {
            Can't ~ show ~ \tl_to_str:N #1 : ~
            \tl_to_str:N #2 \c_space_tl is ~ not ~ a ~ datum ~ marker
          }
          { }
       }
    }
  }
```

Messages:

```
 \msg_new:nnn { datatool-base }
  { show-datum-var-null }
  {
    variable ~ \tl_to_str:N #1 \c_space_tl is ~ null \\
    > ~ #2
  }
 \msg_new:nnn { datatool-base }
  { show-datum-var-empty }
  {
```

```
   variable ~ \tl_to_str:N #1 \c_space_tl is ~ empty
 }
\msg_new:nnn { datatool-base }
 { show-datum-var-unset }
 {
   Showing ~ unset ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ ` #2 '
 }
\msg_new:nnn { datatool-base }
 { show-weird-datum-var-unset }
 {
   Showing ~ unset ~ weird ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ ` #2 '
 }
\msg_new:nnn { datatool-base }
 { show-datum-var-string }
 {
   Showing ~ string ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2
 }
\msg_new:nnn { datatool-base }
 { show-weird-datum-var-string }
 {
   Showing ~ string ~ weird ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2
 }
\msg_new:nnn { datatool-base }
 { show-datum-var-integer }
 {
   Showing ~ integer ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
\msg_new:nnn { datatool-base }
 { show-weird-datum-var-integer }
 {
   Showing ~ integer ~ weird ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
\msg_new:nnn { datatool-base }
 { show-datum-var-decimal }
 {
   Showing ~ decimal ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
```

```
\msg_new:nnn { datatool-base }
 { show-weird-datum-var-decimal }
 {
   Showing ~ decimal ~ weird ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
\msg_new:nnn { datatool-base }
 { show-datum-var-currency }
 {
   Showing ~ currency ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3 \\
   > ~ currency ~ symbol ~ #4
 }
\msg_new:nnn { datatool-base }
 { show-weird-datum-var-currency }
 {
   Showing ~ currency ~ weird ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3 \\
   > ~ currency ~ symbol ~ #4
 }
\msg_new:nnn { datatool-base }
 { show-datum-var-datetime }
 {
   Showing ~ datetime ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
\msg_new:nnn { datatool-base }
 { show-weird-datum-var-datetime }
 {
   Showing ~ datetime ~ weird ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
\msg_new:nnn { datatool-base }
 { show-datum-var-date }
 {
   Showing ~ date ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
\msg_new:nnn { datatool-base }
 { show-weird-datum-var-date }
 {
```

```
    Showing ~ date ~ weird ~ datum ~ variable ~ \tl_to_str:N #1 \\
    > ~ string ~ value ~ #2 \\
    > ~ numeric ~ value ~ #3
  }
\msg_new:nnn { datatool-base }
 { show-datum-var-time }
 {
   Showing ~ time ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
\msg_new:nnn { datatool-base }
 { show-weird-datum-var-time }
 {
   Showing ~ time ~ weird ~ datum ~ variable ~ \tl_to_str:N #1 \\
   > ~ string ~ value ~ #2 \\
   > ~ numeric ~ value ~ #3
 }
```

The above document commands can be used on ⟨*cs*⟩ obtained from parsing data.
Document commands:

\DTLparse

\DTLparse{⟨*cs*⟩}{⟨*arg*⟩}

Note that any dates or times will be treated as strings.

```
\NewDocumentCommand \DTLparse { m m }
 {
   \__datatool_parse:Nn #1 { #2 }
 }
```

Internal function easier to remember the argument order:

```
\cs_new:Nn \__datatool_parse:Nn
{
  \tl_clear:N \l__datatool_datum_value_tl
  \tl_clear:N \l__datatool_datum_currency_tl
  \tl_clear:N \l__datatool_datum_update_value_tl
  \datatool_if_null:nTF { #2 }
   {
     \tl_set_eq:NN #1 \dtlnovalue
     \tl_set:Nn \l__datatool_datum_original_value_tl { #2 }
     \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int
   }
   {
     \tl_if_head_eq_meaning:nNTF
       { #2 } \__datatool_datum:w
      {
```

Convert from weird format:

```
        \tl_set:Nx #1 { #2 }
      }
      {
        \tl_set:Nn #1 { #2 }
      }
     \exp_args:NV \tl_if_head_eq_meaning:nNTF
      #1 \__datatool_datum:nnnn
      {
        \int_set:Nn \@dtl@datatype
          { \exp_args:NV \__datatool_datum_type:n #1 }
        \tl_set:Nx \l__datatool_datum_original_value_tl { #1 }
        \tl_set:Nx \l__datatool_datum_currency_tl
          { \exp_args:NV \__datatool_datum_currency:n #1  }
        \tl_set:Nx \l__datatool_datum_value_tl
          { \exp_args:NV \__datatool_datum_value:n #1 }
      }
      {
       \exp_args:NV \__datatool_parse_datum:n #1
       \tl_if_empty:NTF \l__datatool_datum_update_value_tl
        {
           \int_compare:nNnTF
             { \@dtl@datatype }
               =
             { \c_datatool_unknown_int }
           {
```

NB this may be blank (only a space) not empty, but it's still classed as unknown.

```
             \tl_set:Nn #1
              {
                \__datatool_datum:nnnn { #2 } { } { }
                { \c_datatool_unknown_int }
              }
           }
           {
             \tl_set_eq:NN #1 \l__datatool_datum_original_value_tl
           }
        }
        {
          \tl_set_eq:NN #1 \l__datatool_datum_update_value_tl
        }
     }
   }
 }
\cs_generate_variant:Nn \__datatool_parse:Nn
  { NV }
\cs_new:Nn \__datatool_parse:N
 {
   \exp_args:NNV \__datatool_parse:Nn #1 #1
 }
```

\DTLxparse

\DTLxparse{⟨*cs*⟩}{⟨*arg*⟩}

As above but expands second argument.

```
\NewDocumentCommand \DTLxparse { m m }
 {
   \exp_args:NNx \__datatool_parse:Nn #1 { #2 }
 }
```

Low-level set datum parts.

\datatool_set_datum:Nnnnn ⟨*datum-var*⟩ {⟨*string*⟩} {⟨*value*⟩} {⟨*currency symbol*⟩} {⟨*type*⟩}

The type may be an integer expression

```
\cs_new:Nn \datatool_set_datum:Nnnnn
 {
   \int_case:nnF { #5 }
    {
      { \c_datatool_unknown_int }
       {
         \tl_set:Nn #1
          {
            \__datatool_datum:nnnn
              { #2 } { } { } { \c_datatool_unknown_int }
          }
       }
      { \c_datatool_string_int }
       {
         \tl_set:Nn #1
          {
            \__datatool_datum:nnnn
              { #2 } { } { } { \c_datatool_string_int }
          }
       }
      { \c_datatool_integer_int }
       {
         \tl_set:Nn #1
          {
            \__datatool_datum:nnnn
              { #2 } { #3 } { } { \c_datatool_integer_int }
          }
       }
      { \c_datatool_decimal_int }
       {
         \tl_set:Nn #1
          {
            \__datatool_datum:nnnn
              { #2 } { #3 } { } { \c_datatool_decimal_int }
```

```
          }
        }
       { \c_datatool_currency_int }
        {
          \tl_set:Nn #1
           {
             \__datatool_datum:nnnn
               { #2 } { #3 } { #4 } { \c_datatool_currency_int }
           }
        }
       { \c_datatool_datetime_int }
        {
          \tl_set:Nn #1
           {
             \__datatool_datum:nnnn
               { #2 } { #3 } { } { \c_datatool_datetime_int }
           }
        }
       { \c_datatool_date_int }
        {
          \tl_set:Nn #1
           {
             \__datatool_datum:nnnn
               { #2 } { #3 } { } { \c_datatool_date_int }
           }
        }
       { \c_datatool_time_int }
        {
          \tl_set:Nn #1
           {
             \__datatool_datum:nnnn
               { #2 } { #3 } { } { \c_datatool_time_int }
           }
        }
    }
    {
      \PackageError { datatool-base }
       { Unsupported ~ datum ~ type ~ \int_eval:n { #5 } }
       { }
    }
 }
\cs_generate_variant:Nn \datatool_set_datum:Nnnnn
  { Neeen , Nnenn , NnVVn , NnVnn }
```

No sanity check:

```
\cs_new:Nn \__datatool_set_datum:Nnnnn
 {
  \tl_set:Nn #1
   {
     \__datatool_datum:nnnn { #2 } { #3 } { #4 } { #5 }
```

```
    }
  }
\cs_generate_variant:Nn \__datatool_set_datum:Nnnnn
  { Neeen , NnVnn , NeVnn }
```

Document-level commands:

\DTLsetintegerdatum

`\DTLsetintegerdatum{⟨cs⟩}{⟨formatted value⟩}{⟨value⟩}`

Sets ⟨*cs*⟩ to an integer datum.

```
\NewDocumentCommand \DTLsetintegerdatum { m m m }
 {
   \tl_set:Nn #1
    {
      \__datatool_datum:nnnn
        { #2 } { #3 } { }
        { \c_datatool_integer_int }
    }
 }
```

\DTLxsetintegerdatum

`\DTLxsetintegerdatum{⟨cs⟩}{⟨formatted value⟩}{⟨value⟩}`

As above but expands ⟨*formatted value*⟩ and ⟨*value*⟩.

```
\NewDocumentCommand \DTLxsetintegerdatum { m m m }
 {
   \exp_args:Neee \DTLsetintegerdatum { \exp_not:N #1 }
    { #2 } { #3 }
 }
```

\DTLsetfpdatum

`\DTLsetdecimaldatum{⟨cs⟩}{⟨formatted value⟩}{⟨value⟩}`

Sets ⟨*cs*⟩ to a floating point datum.

```
\NewDocumentCommand \DTLsetfpdatum { m m m }
 {
   \fp_set:Nn \l__datatool_datum_value_fp { #3 }
   \tl_set:Nx #1
    {
      \exp_not:N \__datatool_datum:nnnn
       { \exp_not:n { #2 } }
       {
         \exp_not:N \datatool_datum_fp:nnn
          { #3 }
          { \exp_not:V \l__datatool_datum_value_fp }
          { \fp_to_decimal:N \l__datatool_datum_value_fp }
       }
```

```
        { }
        { \c_datatool_decimal_int }
    }
}
```

\DTLsetdecimaldatum

\DTLsetdecimaldatum{⟨*cs*⟩}{⟨*formatted value*⟩}{⟨*value*⟩}

Sets ⟨*cs*⟩ to a decimal datum.

```
\NewDocumentCommand \DTLsetdecimaldatum { m m m }
 {
   \tl_set:Nn #1
    {
      \__datatool_datum:nnnn
       { #2 } { #3 } { }
       { \c_datatool_decimal_int }
    }
 }
```

\DTLxsetdecimaldatum

\DTLsxetdecimaldatum{⟨*cs*⟩}{⟨*formatted value*⟩}{⟨*value*⟩}

As above but expands ⟨*formatted value*⟩ and ⟨*value*⟩.

```
\NewDocumentCommand \DTLxsetdecimaldatum { m m m }
 {
   \exp_args:Neee \DTLsetdecimaldatum { \exp_not:N #1 }
    { #2 } { #3 }
 }
```

\DTLsetcurrencydatum

\DTLsetcurrencydatum{⟨*cs*⟩}{⟨*formatted value*⟩}{⟨*value*⟩}{⟨*currency symbol*⟩}

Sets ⟨*cs*⟩ to a currency datum.

```
\NewDocumentCommand \DTLsetcurrencydatum { m m m m }
 {
   \tl_set:Nn #1
    {
      \__datatool_datum:nnnn
       { #2 } { #3 } { #4 }
       { \c_datatool_currency_int }
    }
 }
```

\DTLxsetcurrencydatum

\DTLxsetcurrencydatum{⟨*cs*⟩}{⟨*formatted value*⟩}{⟨*value*⟩}{⟨*currency symbol*⟩}

As above but expands ⟨*formatted value*⟩, ⟨*value*⟩ and ⟨*currency symbol*⟩.

```
\NewDocumentCommand \DTLxsetcurrencydatum { m m m m }
 {
   \exp_args:Neee \DTLsetcurrencydatum { \exp_not:N #1 }
    { #2 } { #3 } { #4 }
 }
```

\DTLsettemporaldatum

\DTLsettemporaldatum{⟨*cs*⟩}{⟨*formatted value*⟩}{⟨*ISO*⟩}

Sets ⟨*cs*⟩ to a date/time datum. The actual type is determined by parsing ⟨*ISO*⟩. If the formatted value should be automatically calculated, use `\DTLsetup`{datetime=reformat}`\DTLparse`{⟨*cs*⟩}{⟨*ISO*⟩} and adjust the formatting commands as applicable.

```
\NewDocumentCommand \DTLsettemporaldatum { m m m }
 {
   \__datatool_parse_datetime:nTF { #3 }
    {
       \__datatool_set_datetime_value:Nn #1 { #2 }
    }
    {
       \PackageError { datatool-base }
        { Invalid ~ date/time format ~ ` \tl_to_str:n { #3 } '}
        {
          The ~ date/time ~ value ~ needs ~ to ~ be ~ a ~
          date ~ in ~ the ~ form ~ YYYY-MM-DD ~ or ~ a ~
          time ~ in ~ the ~ form ~ hh:mm ~ or ~ hh:mm:ss ~ or ~ a ~
          timestamp ~ in ~ the ~ form ~ YYYY-MM-DD ~ hh:mm:ssTZ ~
          where ~ the ~ timezone ~ TZ ~ is ~ optional ~
          and ~ may ~ be ~ the ~ letter ~
          `Z' ~ or ~ in ~ the ~ form ~ [+-]hh:mm ~
          (if ~ omitted ~ `Z' ~ is ~ assumed). ~ The ~
          separator ~ may ~ either ~ be ~ a ~ space ~ or ~
          the ~ letter ~ `T'
        }
    }
 }
```

\DTLxsettemporaldatum

\DTLxsetdatetimedatum{⟨*cs*⟩}{⟨*formatted value*⟩}{⟨*ISO*⟩}

As above but expands all but the first argument.

```
\NewDocumentCommand \DTLxsettemporaldatum { m m m }
 {
   \exp_args:Neee \DTLsettemporaldatum { \exp_not:N #1 }
    { #2 } { #3 }
 }
```

\DTLsetstringdatum

\DTLsetstringdatum{⟨*cs*⟩}{⟨*string*⟩}

Sets ⟨*cs*⟩ to a string datum.

```
\NewDocumentCommand \DTLsetstringdatum { m m }
 {
   \tl_set:Nn #1
    {
      \__datatool_datum:nnnn
      { #2 } { } { }
      { \c_datatool_string_int }
    }
 }
```

\DTLxsetstringdatum

\DTLxsetstringdatum{⟨*cs*⟩}{⟨*string*⟩}

As above but expands ⟨*string*⟩.

```
\NewDocumentCommand \DTLxsetstringdatum { m m }
 {
   \exp_args:NNe \DTLxsetstringdatum #1 { #2 }
 }
```

Temporary storage used when extracting data. The original value:

```
\tl_new:N \l__datatool_datum_original_value_tl
```

The numeric value:

```
\tl_new:N \l__datatool_datum_value_tl
```

The currency symbol:

```
\tl_new:N \l__datatool_datum_currency_tl
```

The currency code:

```
\tl_new:N \l__datatool_datum_currency_code_tl
```

If the data hasn't been parsed yet, the following is set to the tagged information:

```
\tl_new:N \l__datatool_datum_update_value_tl
```

Regular expression for scientific notation. Version 3.4: bug fix (expression needs to be anchored)

```
\regex_const:Nn \c_datatool_sci_regex
 { \A [+\-]? [0-9]* \.? [0-9]+ \s* [eE] [+\-]? [0-9]+ \Z }
```

Determine data type. For temporal data types, the argument must match the relevant format. See `\__datatool_parse_datetime_if_enabled:n` and `\__datatool_parse_datetime:n` conditionals.

```
\cs_new:Nn \__datatool_parse_datum:n
 {
```

Initialise to unknown.

```
\@dtl@datatype = \c_datatool_unknown_int
\tl_set:Nn \l__datatool_datum_original_value_tl { #1 }
\tl_clear:N \l__datatool_datum_value_tl
\tl_clear:N \l__datatool_datum_currency_tl
\tl_clear:N \l__datatool_datum_currency_code_tl
\tl_clear:N \l__datatool_datum_update_value_tl
\tl_if_blank:nF { #1 }
{
```

Has the argument already had its data type set?

```
  \tl_if_head_eq_meaning:nNTF { #1 }  \__datatool_datum:nnnn
  { \__datatool_get_datum:w #1 \q_nil \q_stop }
  {
    \tl_if_head_eq_meaning:nNTF { #1 }  \__datatool_datum:w
    { \__datatool_get_weird_datum:w #1 \q_nil \q_stop }
    {
```

Doesn't start with the special marker, so it needs parsing. First check for scientific notation.

```
     \regex_match:NnTF \c_datatool_sci_regex { #1 }
      {
        \fp_set:Nn \l__datatool_datum_value_fp { #1 }
        \@dtl@datatype = \c_datatool_decimal_int
        \tl_set:Nx \l__datatool_datum_value_tl
         {
           \exp_not:N \datatool_datum_fp:nnn
            { #1 }
            { \exp_not:V \l__datatool_datum_value_fp }
            { \fp_to_decimal:N \l__datatool_datum_value_fp }
         }
        \__datatool_if_auto_reformat_on:nTF { si }
         {
```

Auto-reformat on.

```
           \tl_set:Nx \l__datatool_datum_update_value_tl
            {
              \exp_not:N \__datatool_datum:nnnn
               {
                 \exp_not:N \DTLscinum
                   { \exp_not:n { #1 } }
               }
               { \exp_not:V \l__datatool_datum_value_tl }
               { }
               { \exp_not:N \c_datatool_decimal_int }
            }
         }
         {
           \tl_set:Nx \l__datatool_datum_update_value_tl
            {
              \exp_not:N \__datatool_datum:nnnn
```

```
                { \exp_not:n { #1 } }
                { \exp_not:V \l__datatool_datum_value_tl }
                { }
                { \exp_not:N \c_datatool_decimal_int }
              }
            }
          }
          {
```

Next for date/time if applicable.

```
           \__datatool_parse_datetime_if_enabled:nTF { #1 }
           {
             \__datatool_if_auto_reformat_on:TF
              { \__datatool_set_datetime_value: }
              { \__datatool_set_datetime_value:n { #1 } }
           }
           {
```

Not date/time. Next check if it starts with a + or - to allow for a sign before the currency symbol.

```
            \bool_lazy_or:nnTF
              { \tl_if_head_eq_meaning_p:nN { #1 } + }
              { \tl_if_head_eq_meaning_p:nN { #1 } - }
             {
               \exp_args:Nxx \__datatool_parse_item:nn
                { \tl_tail:n { #1 } }
                { \tl_head:n { #1 } }
             }
             {
               \__datatool_parse_item:nn { #1 } { }
             }
           }
         }
       }
     }
   }
```

Parse remaining after prefix check:

```
\cs_new:Nn \__datatool_parse_item:nn
{
  \tl_set:Nn \l__datatool_datum_original_value_tl { #1 }
```

Check if it starts with a known currency marker.

```
  \tl_if_head_eq_meaning:nNTF { #1 } { \DTLcurrency }
   { \__datatool_get_currency_datum:w #1 \q_nil \q_stop }
   {
     \tl_if_head_eq_meaning:nNTF { #1 } { \DTLfmtcurrency }
     { \__datatool_get_fmtcurrency_datum:w #1 \q_nil \q_stop }
     {
       \tl_if_head_eq_meaning:nNTF { #1 } { \DTLfmtcurr }
```

```
        { \__datatool_get_fmtcurr_datum:w #1 \q_nil \q_stop }
        {
```

Doesn't start with `\DTLcurrency` or `\DTLfmtcurrency` or `\DTLfmtcurr`. Does it start with a known currency symbol?

```
          \__datatool_check_known_currencies:
          \tl_if_empty:NTF \l__datatool_datum_currency_tl
          {
```

Not currency. Is it numeric?

```
            \__datatool_parse_numeric:n { #1 }
            \int_compare:nNnT
              { \@dtl@datatype } = { \c_datatool_unknown_int }
            {
              \int_set_eq:NN \@dtl@datatype \c_datatool_string_int
            }
          }
          {
```

Starts (or ends) with a currency symbol. Is it followed by a numeric value?

```
            \tl_trim_spaces:N \l__datatool_suffix_tl
            \__datatool_parse_numeric:N \l__datatool_suffix_tl
            \int_compare:nNnTF
              { \@dtl@datatype } = { \c_datatool_string_int }
            {
              \tl_clear:N \l__datatool_datum_currency_tl
            }
            {
              \int_set_eq:NN \@dtl@datatype \c_datatool_currency_int
            }
          }
        }
      }
    }
```

Check for a double sign which will be treated as a string.

```
    \tl_if_empty:nF { #2 }
     {
       \int_compare:nNnT
         { \@dtl@datatype } > { \c_datatool_string_int }
       {
         \bool_lazy_or:nnTF
           { \tl_if_head_eq_meaning_p:nN { #1 } + }
           { \tl_if_head_eq_meaning_p:nN { #1 } - }
          {
            \int_set_eq:NN \@dtl@datatype \c_datatool_string_int
          }
          {
            \tl_if_eq:nnT { #2 } { - }
             {
               \tl_set:Nx \l__datatool_datum_value_tl
```

```
              { \fp_to_tl:n { - ( \l__datatool_datum_value_tl ) } }
          }
        }
      }
    }
   \int_case:nn { \@dtl@datatype }
   {
     { \c_datatool_string_int }
      {
```

Non-numeric.

```
       \tl_set:Nn \l__datatool_datum_update_value_tl
        {
          \__datatool_datum:nnnn { #2 #1 } { } { }
            { \c_datatool_string_int }
        }
      }
     { \c_datatool_integer_int }
      {
```

Integer.

```
       \__datatool_if_auto_reformat_on:nTF { integer }
        {
```

Auto-reformat on.

```
         \tl_set:Nx
           \l__datatool_datum_update_value_tl
           { \int_eval:n { \l__datatool_datum_value_tl } }
         \regex_replace_case_once:nN
          {
            \c_datatool_integer_grps_i_regex
               { \1 \2 \u{l__datatool_numbergroup_tl} \3 }
            \c_datatool_integer_grps_ii_regex
               { \1 \2 \u{l__datatool_numbergroup_tl} \3
                    \u{l__datatool_numbergroup_tl} \4 }
            \c_datatool_integer_grps_iii_regex
               { \1 \2 \u{l__datatool_numbergroup_tl} \3
                    \u{l__datatool_numbergroup_tl} \4
                    \u{l__datatool_numbergroup_tl} \5 }
          }
         \l__datatool_datum_update_value_tl
         \tl_set:Ne \l__datatool_datum_update_value_tl
          {
            \exp_not:N \__datatool_datum:nnnn
             {
               \exp_not:V \l__datatool_datum_update_value_tl
             }
             { \int_eval:n { \l__datatool_datum_value_tl } } { }
             { \c_datatool_integer_int }
          }
        }
```

```
          {
           \tl_set:Nx \l__datatool_datum_update_value_tl
            {
              \exp_not:N \__datatool_datum:nnnn
               { \exp_not:n { #2 #1 } }
               { \int_eval:n { \l__datatool_datum_value_tl } } { }
               { \c_datatool_integer_int }
            }
          }
        }
       { \c_datatool_decimal_int }
        {
```

Decimal.

```
          \tl_set_eq:NN
            \l__datatool_datum_update_value_tl
            \l__datatool_datum_value_tl
          \fp_set:Nn \l__datatool_datum_value_fp
            { \l__datatool_datum_value_tl }
          \tl_set:Nx \l__datatool_datum_value_tl
           {
            \exp_not:N \datatool_datum_fp:nnn
             { \exp_not:V \l__datatool_datum_value_tl }
             { \exp_not:V \l__datatool_datum_value_fp }
             { \fp_to_decimal:N \l__datatool_datum_value_fp }
           }
          \__datatool_if_auto_reformat_on:nTF { decimal }
           {
```

Auto-reformat on.

```
              \regex_replace_case_once:nN
                {
                  \c_datatool_decimal_grps_regex
                    { \1 \2 \u{l__datatool_decimal_tl} \3 }
                  \c_datatool_decimal_grps_i_regex
                     { \1 \2 \u{l__datatool_numbergroup_tl} \3 \u{l__datatool_decimal_tl} \
                  \c_datatool_decimal_grps_ii_regex
                     { \1 \2 \u{l__datatool_numbergroup_tl} \3
                          \u{l__datatool_numbergroup_tl} \4
                          \u{l__datatool_decimal_tl} \5 }
                  \c_datatool_decimal_grps_iii_regex
                     { \1 \2 \u{l__datatool_numbergroup_tl} \3
                          \u{l__datatool_numbergroup_tl} \4
                          \u{l__datatool_numbergroup_tl} \5
                          \u{l__datatool_decimal_tl} \6 }
                }
               \l__datatool_datum_update_value_tl
              \tl_set:Ne \l__datatool_datum_update_value_tl
               {
                 \exp_not:N \__datatool_datum:nnnn
                  {
```

```
                \exp_not:V \l__datatool_datum_update_value_tl
              }
              { \exp_not:V \l__datatool_datum_value_tl }
              { }
              { \c_datatool_decimal_int }
            }
          }
          {
```

Auto-reformat off.

```
            \tl_set:Nx \l__datatool_datum_update_value_tl
             {
               \exp_not:N \__datatool_datum:nnnn
                { \exp_not:n { #2 #1 } }
                { \exp_not:V \l__datatool_datum_value_tl }
                { }
                { \c_datatool_decimal_int }
             }
          }
       }
      { \c_datatool_currency_int }
       {
```

Currency.

```
         \__datatool_if_auto_reformat_on:nTF { currency }
         {
```

Auto-reformat on.

```
           \__datatool_decimal_to_currency:VVNV
             \l__datatool_datum_currency_tl
             \l__datatool_datum_value_tl
             \l__datatool_datum_update_value_tl
             \l__datatool_datum_currency_code_tl
         }
         {
```

Auto-reformat off.

```
          \tl_set:Nx \l__datatool_datum_update_value_tl
           {
             \exp_not:N \__datatool_datum:nnnn
               { \exp_not:n { #2 #1 } }
               { \l__datatool_datum_value_tl }
               { \exp_not:V \l__datatool_datum_currency_tl }
               { \c_datatool_currency_int }
           }
         }
       }
   }
  \tl_set:Nn \l__datatool_datum_original_value_tl { #2 #1 }
}
```

Parse for a numeric value, using the current group and decimal characters.

```
\tl_new:N \l__datatool_parser_num_tl
\cs_new:Nn \__datatool_parse_numeric:N
 {
   \exp_args:No \__datatool_parse_numeric:n { #1 }
 }
\cs_new:Nn \__datatool_parse_numeric:n
 {
   \tl_set:Nn \l__datatool_parser_num_tl { #1 }
   \tl_if_empty:nF { #1 }
   {
     \regex_match:NnTF \l_datatool_locale_bigint_regex { #1 }
      {
        \@dtl@datatype = \c_datatool_string_int
      }
      {
        \regex_replace_case_once:nNTF
         {
           \l_datatool_locale_numeric_regex
           { \c{q_mark} \1 \2 \3 \4 . \5 \c{q_stop}}
           \l_datatool_locale_fractional_regex
           { \c{q_mark} \1 . \2 \c{q_stop}}
           \l_datatool_locale_fractional_nozero_regex
           { \c{q_mark} \1 0 . \2 \c{q_stop}}
         }
         \l__datatool_parser_num_tl
         { \exp_after:wN \__datatool_parse_number:w \l__datatool_parser_num_tl }
         { \@dtl@datatype = \c_datatool_string_int }
      }
   }
 }
\cs_new:Npn \__datatool_parse_number:w \q_mark #1.#2\q_stop
{
  \tl_if_empty:nTF { #2 }
  {
    \@dtl@datatype = \c_datatool_integer_int
    \tl_set:Nn \l__datatool_datum_value_tl { #1 }
  }
  {
    \@dtl@datatype = \c_datatool_decimal_int
    \tl_if_empty:nTF { #1 }
     { \tl_set:Nn \l__datatool_datum_value_tl { 0.#2 } }
     { \tl_set:Nn \l__datatool_datum_value_tl { #1.#2 } }
  }
}
```

Check for known currencies.

```
\cs_new:Nn \__datatool_check_known_currencies:
 {
```

Check regions first (this will obtain the currency code to save searching for it again).

```
   \tl_clear:N \l__datatool_datum_currency_code_tl
```

```
    \prop_map_function:NN \l_datatool_regional_currencies_prop
      \__datatool_prop_parse_currency_do:nn
    \tl_if_empty:NT \l__datatool_datum_currency_tl
     {
      \seq_map_function:NN
       \l__datatool_known_currencies_seq
       \__datatool_seq_parse_currency_do:n
     }
```

If suffix is blank then it may just be the symbol.

```
    \exp_args:NV \tl_if_blank:nT \l__datatool_suffix_tl
     {
       \tl_clear:N \l__datatool_datum_currency_tl
     }
  }
```

Handler macro for currency iterator for property map.

```
 \cs_new:Nn \__datatool_prop_parse_currency_do:nn
 {
   \__datatool_if_starts_or_ends_with:VvTF
       \l__datatool_datum_original_value_tl
       { dtl@curr@ #2 @tl }
    {
      \tl_set_eq:NN \l__datatool_datum_currency_tl \l__datatool_prefix_tl
      \tl_set:Nn \l__datatool_datum_currency_code_tl { #2 }
    }
    {
      \__datatool_if_starts_or_ends_with:VvTF
         \l__datatool_datum_original_value_tl
         { dtl@curr@ #2 @sym }
       {
         \tl_set_eq:NN \l__datatool_datum_currency_tl \l__datatool_prefix_tl
         \tl_set:Nn \l__datatool_datum_currency_code_tl { #2 }
       }
       {
         \cs_if_exist:cT { datatool #1 symbolprefix }
          {
            \__datatool_if_starts_or_ends_with:VeTF
               \l__datatool_datum_original_value_tl
               { #1 \exp_not:v { dtl@curr@ #2 @tl } }
             {
               \tl_set_eq:NN \l__datatool_datum_currency_tl \l__datatool_prefix_tl
               \tl_set:Nn \l__datatool_datum_currency_code_tl { #2 }
             }
             {
               \__datatool_if_starts_or_ends_with:VeTF
                   \l__datatool_datum_original_value_tl
                   { #1 \exp_not:v { dtl@curr@ #2 @sym } }
                {
                  \tl_set_eq:NN \l__datatool_datum_currency_tl \l__datatool_prefix_tl
                  \tl_set:Nn \l__datatool_datum_currency_code_tl { #2 }
```

```
                }
              }
            }
          }
      }
    \tl_if_empty:NF \l__datatool_datum_currency_tl
     {
       \prop_map_break:
     }
  }
```

Handler macro for currency iterator for sequence map.

```
 \cs_new:Nn \__datatool_seq_parse_currency_do:n
 {
   \__datatool_if_starts_or_ends_with:VnTF
      \l__datatool_datum_original_value_tl { #1 }
     {
       \tl_set_eq:NN \l__datatool_datum_currency_tl \l__datatool_prefix_tl
       \seq_map_break:
     }
 }
```

Parse content that starts with `\__datatool_datum:nnnn`.

```
 \cs_new:Npn \__datatool_get_datum:w
   \__datatool_datum:nnnn #1 #2 #3 #4 #5 \q_stop
 {
   \quark_if_nil:nTF { #5 }
   {
     \@dtl@datatype = #4
     \tl_set:Nn \l__datatool_datum_original_value_tl { #1 }
     \tl_set:Nn \l__datatool_datum_value_tl { #2 }
     \tl_set:Nn \l__datatool_datum_currency_tl { #3 }
   }
   {
     \@dtl@datatype = \c_datatool_string_int
   }
 }
```

Parse content that starts with `\DTLcurrency`.

```
 \cs_new:Npn \__datatool_get_currency_datum:w
   \DTLcurrency #1 #2 \q_stop
 {
   \quark_if_nil:nTF { #2 }
   {
     \__datatool_parse_numeric:n { #1 }
     \ifnum\@dtl@datatype = \c_datatool_string_int
     \else
       \tl_set:Nn \l__datatool_datum_currency_tl { \@dtl@currency }
       \@dtl@datatype = \c_datatool_currency_int
     \fi
   }
```

```
    {
      \@dtl@datatype = \c_datatool_string_int
    }
  }
```

Parse content that starts with `\DTLfmtcurrency`.

```
\cs_new:Npn \__datatool_get_fmtcurrency_datum:w
  \DTLfmtcurrency #1 #2 #3 \q_stop
 {
   \quark_if_nil:nTF { #3 }
   {
     \__datatool_parse_numeric:n { #2 }
     \ifnum\@dtl@datatype = \c_datatool_string_int
     \else
       \@dtl@datatype = \c_datatool_currency_int
       \tl_set:Nn \l__datatool_datum_currency_tl { #1 }
     \fi
   }
   {
     \@dtl@datatype = \c_datatool_string_int
   }
 }
```

Parse content that starts with `\DTLfmtcurr`.

```
\cs_new:Npn \__datatool_get_fmtcurr_datum:w
  \DTLfmtcurr #1 #2 #3 \q_stop
 {
   \quark_if_nil:nTF { #3 }
   {
     \__datatool_parse_numeric:n { #2 }
     \ifnum\@dtl@datatype = \c_datatool_string_int
     \else
       \tl_set:Nn \l__datatool_datum_currency_code_tl { #1 }
       \@dtl@datatype = \c_datatool_currency_int
       \tl_set:Nn \l__datatool_datum_currency_tl { \DTLcurr { #1 } }
     \fi
   }
   {
     \@dtl@datatype = \c_datatool_string_int
   }
 }
```

Only parse for dates and times if enabled. Note that the boolean setting is just for general parsing commands. Temporal values can be explicitly parsed.

```
\prg_new_conditional:Npnn \__datatool_parse_datetime_if_enabled:n #1
  { T, F, TF }
{
  \bool_if:NTF \l__datatool_parse_datetime_bool
   {
     \__datatool_parse_datetime:nTF { #1 }
      { \prg_return_true: }
```

```
      { \prg_return_false: }
    }
    { \prg_return_false: }
}
```

Test if the given argument is a token list containing datum.

```
\cs_new:Nn \__datatool_if_datum_tl:nTF
 {
   \tl_if_single_token:nTF { #1 }
   {
     \exp_args:No \tl_if_head_eq_meaning:nNTF { #1 } \__datatool_datum:nnnn
     { #2 }
     { #3 }
   }
   { #3 }
 }
```

Added v3.1: Not a number warning:

```
\cs_new:Nn \datatool_warn_not_number:n
 {
   \PackageWarning { datatool-base } { #1 }
 }
```

\DTLconverttodecimal

\DTLconverttodecimal{⟨*num*⟩}{⟨*cmd*⟩}

`\DTLconverttodecimal` will convert locale dependent ⟨*num*⟩ a decimal number in a standard form that can be used in arithmetical calculations. The resulting number is stored in ⟨*cmd*⟩.

```
\NewDocumentCommand \DTLconverttodecimal { m m }
{
  \tl_if_single_token:nTF { #1 }
   {
     \exp_args:No \__datatool_parse_datum:n { #1 }
   }
   {
     \__datatool_parse_datum:n { #1 }
   }
  \datatool_if_numeric_datum_type:nF { \@dtl@datatype }
   {
     \datatool_warn_not_number:n
      { Can't ~ convert ~ non-numerical ~ ` #1 ' ~ to ~ decimal }
   }
  \tl_if_empty:NTF \l__datatool_datum_value_tl
   { \tl_set:Nn #2 { 0 } }
   { \tl_set_eq:NN #2 \l__datatool_datum_value_tl }
}
```

\DTLdecimaltolocale

\DTLdecimaltolocale{⟨*number*⟩}{⟨*cmd*⟩}

Define command to convert a decimal number into the locale dependent format. Stores result in ⟨*cmd*⟩ which must be a control sequence.

```
\newrobustcmd*{\DTLdecimaltolocale}[2]{%
 \tl_set:Nn #2 { #1 }
 \regex_replace_case_once:nN
 {
   \c_datatool_numeric_leading_zeros_regex { \1 \2 }
   \c_datatool_decimal_redundant_zeros_regex { \1 \2 }
   \c_datatool_decimal_implicit_zero_regex { \1 0 \2 }
 }
 #2
 \regex_replace_case_once:nNTF
 {
    \c_datatool_decimal_grps_regex
      { \1 \2 \u{l__datatool_decimal_tl} \3 }
    \c_datatool_decimal_grps_i_regex
       { \1 \2 \u{l__datatool_numbergroup_tl} \3 \u{l__datatool_decimal_tl} \4 }
    \c_datatool_decimal_grps_ii_regex
       { \1 \2 \u{l__datatool_numbergroup_tl} \3
            \u{l__datatool_numbergroup_tl} \4
            \u{l__datatool_decimal_tl} \5 }
    \c_datatool_decimal_grps_iii_regex
       { \1 \2 \u{l__datatool_numbergroup_tl} \3
            \u{l__datatool_numbergroup_tl} \4
            \u{l__datatool_numbergroup_tl} \5
            \u{l__datatool_decimal_tl} \6 }
 }
 #2
 {
   \tl_set:Ne #2
    {
     \exp_not:N \__datatool_datum:nnnn
      { \exp_not:V #2 }
      { #1 }
      {  }
      { \exp_not:N \c_datatool_decimal_int }
    }
  }
  {
    \regex_replace_case_once:nNTF
    {
      \c_datatool_integer_grps_i_regex
         { \1 \2 \u{l__datatool_numbergroup_tl} \3 }
      \c_datatool_integer_grps_ii_regex
         { \1 \2 \u{l__datatool_numbergroup_tl} \3
              \u{l__datatool_numbergroup_tl} \4 }
```

```
      \c_datatool_integer_grps_iii_regex
         { \1 \2 \u{l__datatool_numbergroup_tl} \3
              \u{l__datatool_numbergroup_tl} \4
              \u{l__datatool_numbergroup_tl} \5 }
```

This last one ensures that the true branch is followed:

```
      \c_datatool_integer_no_grps_regex { \1 \2 }
    }
    #2
    {
      \tl_set:Ne #2
       {
        \exp_not:N \__datatool_datum:nnnn
         { \exp_not:V #2 }
         { #1 }
         {  }
         { \exp_not:N \c_datatool_integer_int }
       }
    }
    {
```

This is only a warning rather than an error to allow for databases that contain markup (e.g. `\textemdash` or "N/A") to indicate a missing value. The numeric functions may either interpret strings as zero or skip them.

```
      \datatool_warn_not_number:n
       {
         Can't ~ convert ~ `#1' ~ to ~ decimal: ~ not ~ a ~ number
       }
      \tl_set:Nn #2
       {
         \__datatool_datum:nnnn
          { #1 }
          {  }
          {  }
          { \c_datatool_string_int }
       }
    }
  }
}
```

`\DTLdecimaltocurrency`

`\DTLdecimaltocurrency[`⟨*currency symbol*⟩`]{`⟨*number*⟩`}{`⟨*cmd*⟩`}`

This converts a decimal number into the locale dependent currency format. Stores result in ⟨*cmd*⟩ which must be a control sequence. An empty value will be treated as zero.

```
\NewDocumentCommand{\DTLdecimaltocurrency} { O{} m m }
{
```

```
   \__datatool_decimal_to_currency:nnN { #1 } { #2 } { #3 }
}
```

Round currency to this many places (set to empty for no rounding):

```
\newcommand\DTLCurrentLocaleCurrencyDP{2}
\cs_new:Nn \__datatool_decimal_to_currency:nnN
{
  \__datatool_decimal_to_currency:nnNn { #1 } { #2 } #3 { }
}
\cs_generate_variant:Nn \__datatool_decimal_to_currency:nnN { VVN , nVN }
\cs_new:Nn \__datatool_decimal_to_currency:nnNn
{
 \regex_match:NnTF \c_datatool_any_numeric_regex { #2 }
  {
   \tl_if_empty:NTF \DTLCurrentLocaleCurrencyDP
    {
      \tl_set:Nn #3 { #2 }
      \tl_if_empty:NTF #3
       {
         \tl_set:Nn #3 { 0 }
       }
       {
         \regex_replace_once:NnN
          \c_datatool_integer_leading_zeros_regex { \1 \2 } #3
       }
    }
    {
      \tl_if_empty:nTF { #2 }
       {
         \dtlround { #3 } { 0 } { \DTLCurrentLocaleCurrencyDP }
       }
       {
         \dtlround { #3 } { #2 } { \DTLCurrentLocaleCurrencyDP }
       }
      \tl_set:Ne #3
        {
          \exp_args:Ne \__datatool_decimal_trunc_pad_zeros:nn
           { #3 }
           { \DTLCurrentLocaleCurrencyDP }
        }
    }
   \regex_replace_case_once:nN
    {
      \c_datatool_decimal_grps_regex { \1 \2 \u{l__datatool_decimal_tl} \3 }
      \c_datatool_decimal_grps_i_regex
         { \1 \2 \u{l__datatool_numbergroup_tl} \3 \u{l__datatool_decimal_tl} \4 }
      \c_datatool_decimal_grps_ii_regex
         { \1 \2 \u{l__datatool_numbergroup_tl} \3
              \u{l__datatool_numbergroup_tl} \4
```

```
          \u{l__datatool_decimal_tl} \5 }
  \c_datatool_decimal_grps_iii_regex
     { \1 \2 \u{l__datatool_numbergroup_tl} \3
          \u{l__datatool_numbergroup_tl} \4
          \u{l__datatool_numbergroup_tl} \5
          \u{l__datatool_decimal_tl} \6 }
  \c_datatool_integer_grps_i_regex
     { \1 \2 \u{l__datatool_numbergroup_tl} \3 }
  \c_datatool_integer_grps_ii_regex
     { \1 \2 \u{l__datatool_numbergroup_tl} \3
          \u{l__datatool_numbergroup_tl} \4 }
  \c_datatool_integer_grps_iii_regex
     { \1 \2 \u{l__datatool_numbergroup_tl} \3
          \u{l__datatool_numbergroup_tl} \4
          \u{l__datatool_numbergroup_tl} \5 }
}
#3
\tl_set:Ne \l__datatool_datum_currency_code_tl { #4 }
\tl_if_blank:nTF { #1 }
 {
   \tl_if_empty:NTF \l__datatool_datum_currency_code_tl
    {
     \seq_if_in:NVTF
        \l_datatool_regional_currencies_seq
        \DTLCurrencyCode
      {
       \tl_set:Ne #3
        {
         \exp_not:N \__datatool_datum:nnnn
          {
            \exp_not:N \DTLfmtcurr
              { \DTLCurrencyCode }
              { \exp_not:V #3 }
          }
          { #2 }
          { \exp_not:V \@dtl@currency }
          { \exp_not:N \c_datatool_currency_int }
        }
      }
      {
       \tl_set:Ne #3
        {
         \exp_not:N \__datatool_datum:nnnn
          {
            \exp_not:N \DTLfmtcurrency
              { \exp_not:V \@dtl@currency }
              { \exp_not:V #3 }
          }
          { #2 }
          { \exp_not:V \@dtl@currency }
```

```
          { \exp_not:N \c_datatool_currency_int }
        }
      }
    }
    {
     \tl_set:Ne #3
      {
       \exp_not:N \__datatool_datum:nnnn
        {
          \exp_not:N \DTLfmtcurr
            { \l__datatool_datum_currency_code_tl }
            { \exp_not:V #3 }
        }
        { #2 }
        {
          \exp_not:N \DTLcurr
           { \l__datatool_datum_currency_code_tl }
        }
        { \exp_not:N \c_datatool_currency_int }
      }
    }
}
{
  \tl_if_empty:NT \l__datatool_datum_currency_code_tl
   {
     \datatool_get_currency_code:NV
      \l__datatool_datum_currency_code_tl
      \l__datatool_datum_currency_tl
     \tl_if_eq:NnT
      \l__datatool_datum_currency_code_tl { XXX }
      {
        \tl_clear:N \l__datatool_datum_currency_code_tl
      }
   }
  \tl_if_empty:NTF \l__datatool_datum_currency_code_tl
   {
     \tl_set:Ne #3
      {
        \exp_not:N \__datatool_datum:nnnn
        {
          \exp_not:N \DTLfmtcurrency
           { \exp_not:n { #1 } }
           { \exp_not:V #3 }
        }
        { #2 }
        { \exp_not:n { #1 } }
        { \exp_not:N \c_datatool_currency_int }
      }
   }
   {
```

```
          \tl_set:Ne #3
           {
            \exp_not:N \__datatool_datum:nnnn
             {
                \exp_not:N \DTLfmtcurr
                 { \l__datatool_datum_currency_code_tl }
                 { \exp_not:V #3 }
             }
             { #2 }
             { \exp_not:n { #1 } }
             { \exp_not:N \c_datatool_currency_int }
           }
         }
       }
   }
   {
```

This is only a warning rather than an error to allow for databases that contain markup (e.g. `\textemdash` or "N/A") to indicate a missing value. The numeric functions may either interpret strings as zero or skip them.

```
     \datatool_warn_not_number:n
      {
        Can't ~ convert ~ `#2' ~ to ~ currency: ~ not ~ a ~ number
      }
     \tl_set:Nn #3
      {
       \__datatool_datum:nnnn
        { #2 }
        {  }
        {  }
        { \c_datatool_string_int }
      }
  }
 }
 \cs_generate_variant:Nn \__datatool_decimal_to_currency:nnNn
   { VVNV , nVNV }
```

`\DTLdecimaltodatetime`

`\DTLdecimaltodatetime{`⟨*number*⟩`}{`⟨*cmd*⟩`}`

Define command to convert a decimal number into a datetime datum. Stores result in ⟨*cmd*⟩ which must be a control sequence.

```
 \NewDocumentCommand \DTLdecimaltodatetime { m m }
  {
    \datatool_decimal_to_temporal:Nnn
      #2 { \c_datatool_datetime_int } { #1 }
  }
```

\DTLdecimaltodate

```
\DTLdecimaltodate{⟨JDN⟩}{⟨cmd⟩}
```

Define command to convert a number (a Julian Day Number) into a date datum. Stores result in ⟨*cmd*⟩ which must be a control sequence.

```
\NewDocumentCommand \DTLdecimaltodate { m m }
 {
   \datatool_decimal_to_temporal:Nnn
     #2 { \c_datatool_date_int } { #1 }
 }
```

\DTLdecimaltotime

```
\DTLdecimaltotime{⟨JT⟩}{⟨cmd⟩}
```

Define command to convert a decimal number (Julian time of day fraction) into a time datum. Stores result in ⟨*cmd*⟩ which must be a control sequence.

```
\NewDocumentCommand \DTLdecimaltotime { m m }
 {
   \datatool_decimal_to_temporal:Nnn
     #2 { \c_datatool_time_int } { #1 }
 }
```

2.4.2 Dates and Times

New to version 3.0. No localisation is performed. These functions just deal with parsing ISO dates, times and timestamps. There are in total eight numeric elements of a timestamp: year, month number (1–12), day of month number (1–31), hour of day number (0–23), minute of hour number (0–59), seconds of minute number (0–59), time zone hour offset (-12–$+12$), and time zone minute offset (0–59). This can lead to commands with a lot of arguments, so a specially formatted sequence with 8 items is sometimes used instead. Each item is formatted when it is set in the sequence to make it easier to reconstruct a timestamp.

Dates, times and timestamps can be represented numerically. The Julian Day Number (JDN) is the integer number of whole solar days since noon Universal Time. A value of zero represents Monday 1st January 4713 BC, according to the proleptic Julian Calendar. So, noon 1st January 2000 has the Julian Day Number 2451545. The time of day is represented as a fractional number from noon. A negative value is a time before noon. A positive number is a time after noon. Noon has a value of 0.0. Twelve hours before noon, that is, the previous midnight has a value of -0.5. Twelve hours after noon, that is, the following midnight has a value of 0.5.

The Julian Date is a decimal that is simply the sum of the Julian Day Number and the time fraction. For example, $2451545 + 0.5 = 2451545.5$ is midnight after noon 2000-01-01 and $2451545 - 0.5 = 2451544.5$ is midnight before noon 2000-01-01. Note that while the term "Julian Date" is sometimes used to refer to the ordinal day of year in other works, from datatool's point of view, "Julian Date" is the decimal sum of the JDN and the time of day fraction.

Since JDN 0 is a Monday, the day of the week (starting with Monday=0) can be obtained from the remainder of the integer division of the Julian Day Number and 7 (that is, ⟨*JDN*⟩ modulo 7).

A temporal datum command has the data type set to one of the temporal identifiers (datetime, date or time). The string part may be the original string parsed or maybe a formatted version of the temporal value.

`\DTLtemporalvalue` It's useful to retain the original ISO specification. Since the string part of the datum variable may be used to provide a localised format, the original ISO format is hidden in the value part, which is set to:

```
\DTLtemporalvalue{⟨number⟩}{⟨ISO⟩}
```

This allows both the original ISO representation to be extracted as well as the corresponding numerical value. The default definition allows the ISO part to be discarded in numerical contexts. This means that using `\DTLdatumvalue` will still expand to a numeric value, but it requires more expansion than the other numeric data types.

This command isn't an internal command as it may occur in a dbtex-3 file, which expects the normal document category codes. The number depends on the datum type: a decimal Julian date for timestamps, an integer Julian day for date only or a decimal (fraction of a day) for time only.

```
\newcommand \DTLtemporalvalue [2] { #1 }
```

```
\datatool_decimal_to_temporal:Nnn
⟨datum-cs⟩{⟨data-type⟩}{⟨number⟩}
```

Converts a number to a datum-cs.

```
\cs_new_nopar:Nn \datatool_decimal_to_temporal:Nnn
{
  \int_case:nnF { #2 }
   {
     { \c_datatool_datetime_int }
       {
         \datatool_from_julian_date:nN { #3 }
           \l__datatool_datetime_seq
         \datatool_timestamp_to_datetime_datum:NeN
          \l__datatool_datetime_seq
          { #3 }
          #1
       }
     { \c_datatool_date_int }
       {
         \int_set:Nn \l__datatool_julian_int { \fp_to_int:n { #3 } }
         \datatool_from_julian_day:VNNN \l__datatool_julian_int
          \l__datatool_year_int
          \l__datatool_month_int
          \l__datatool_day_int
```

```
    \tl_set:Ne #1
     {
       \exp_not:N \__datatool_datum:nnnn
        {% formatted string
          \exp_not:N \DataToolDateFmt
            { \int_use:N \l__datatool_year_int }
            { \int_use:N \l__datatool_month_int }
            { \int_use:N \l__datatool_day_int }
            {
              \datatool_julian_day_to_dow:n { \l__datatool_julian_int }
            }
        }
        {% value
          \exp_not:N \DTLtemporalvalue
           { \int_use:N \l__datatool_julian_int }
           {
             \int_use:N \l__datatool_year_int
              -
             \datatool_two_digits:N \l__datatool_month_int
              -
             \datatool_two_digits:N \l__datatool_day_int
           }
        }
        { }% no currency symbol
        { \exp_not:N \c_datatool_date_int }
     }
  }
{ \c_datatool_time_int }
  {
    \datatool_from_julian_time:eNNN { #3 }
     \l__datatool_hour_int
     \l__datatool_minute_int
     \l__datatool_second_int
    \tl_set:Ne #1
     {
       \exp_not:N \__datatool_datum:nnnn
        {% formatted string
          \exp_not:N \DataToolTimeFmt
            { \int_use:N \l__datatool_hour_int }
            { \int_use:N \l__datatool_minute_int }
            { \int_use:N \l__datatool_second_int }
        }
        {% value
          \exp_not:N \DTLtemporalvalue
           { #3 }
           {
             \datatool_two_digits:n { \l__datatool_hour_int }
             \c_colon_str
             \datatool_two_digits:N \l__datatool_minute_int
             \c_colon_str
```

```
                \datatool_two_digits:N \l__datatool_second_int
              }
            }
            { }% no currency symbol
            { \exp_not:N \c_datatool_time_int }
          }
        }
    }
    {
      \PackageError { datatool-base }
       { Invalid ~ data ~ type ~ \int_eval:n { #2 }: ~ not ~ a ~ temporal ~ type }
       {
         The ~ second ~ argument ~ of
         \token_to_str:N \datatool_decimal_to_temporal:Nnn ~
         \c_space_tl ~ should ~ be ~ one ~ of: ~
         \exp_not:N \c_datatool_datetime_int , ~
         \exp_not:N \c_datatool_date_int , ~ or ~
         \exp_not:N \c_datatool_time_int
       }
    }
}
\cs_generate_variant:Nn \datatool_decimal_to_temporal:Nnn
 { NnV }
```

`\datatool_extract_timestamp:NN{`⟨*datum-cs*⟩`}`
`{`⟨*result-tl*⟩`}`

Extracts the date/time data stored in the given datum command and defines the supplied token list command to that value. If there's no date/time data in the datum command, the result token list variable will be set to empty. This may mean that the datum variable isn't a temporal variable, but may also mean that the value has lost its special markup due to expansion.

```
\cs_new:Nn \datatool_extract_timestamp:NN
 {
   \tl_clear:N #2
   \group_begin:
    \cs_set_eq:NN \DTLtemporalvalue \use_ii:nn
    \tl_set:Ne \l__datatool_tmpa_tl
      { \DTLdatumvalue { #1 } }
    \regex_match:NVTF
      \c_datatool_temporal_regex \l__datatool_tmpa_tl
     {
       \tl_set:Ne \l__datatool_tmpa_tl
         { \exp_not:N \tl_set:Nn \exp_not:N #2 { \l__datatool_tmpa_tl } }
     }
     {
       \tl_clear:N \l__datatool_tmpa_tl
     }
```

```
    \exp_after:wN
    \group_end:
       \l__datatool_tmpa_tl
 }
```

\DTLdatumtoDTM

\DTLdatumtoDTM{⟨*datum-cs*⟩}{⟨*DTM-name*⟩}

Converts the time stamp stored in the given datum command to a datetime2 date saved with \DTMsavetimestamp, \DTMsavedate, or \DTLsavetime. If the time stamp is missing, this will attempt to recalculate the relevant information based on the numeric value and data type.

```
\NewDocumentCommand \DTLdatumtoDTM { m m }
{
 \cs_if_exist:NTF \DTMsavetimestamp
  {
   \group_begin:
   \cs_set_eq:NN \DTLtemporalvalue \use_ii:nn
```

If the value contains \DTLtemporalvalue the temporary token list variable will be set to the ISO format in the second argument.

```
    \tl_set:Ne \l__datatool_tmpa_tl
      { \DTLdatumvalue { #1 } }
    \regex_match:NVTF
      \c_datatool_timestamp_regex \l__datatool_tmpa_tl
     {
```

Result matches a timestamp, so use \DTMsavetimestamp.

```
       \tl_set:Ne \l__datatool_tmpa_tl
        { \exp_not:N \DTMsavetimestamp { #2 } { \l__datatool_tmpa_tl } }
     }
     {
       \regex_match:NVTF
         \c_datatool_date_regex \l__datatool_tmpa_tl
        {
```

Result only matches a date, so use \DTMsavedate.

```
          \tl_set:Ne \l__datatool_tmpa_tl
            { \exp_not:N \DTMsavedate { #2 } { \l__datatool_tmpa_tl } }
        }
        {
          \regex_extract_once:NVNTF
            \c_datatool_time_regex
            \l__datatool_tmpa_tl
            \l_datatool_timestamp_match_seq
           {
```

Result only matches a time, so use \DTMsavetime but the seconds may be missing, which are required.

```
            \tl_if_empty:eTF
```

```
                    { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
                  {
                    \tl_set:Ne \l__datatool_tmpa_tl
                      { \exp_not:N \DTMsavetime { #2 } { \l__datatool_tmpa_tl : 00 } }
                  }
                  {
                    \tl_set:Ne \l__datatool_tmpa_tl
                      { \exp_not:N \DTMsavetime { #2 } { \l__datatool_tmpa_tl } }
                  }
              }
              {
                \tl_if_empty:NTF \l__datatool_tmpa_tl
                  {
                    \PackageError { datatool-base }
                    {
                       Can't ~ obtain ~ timestamp. ~
                       No ~ value ~ associated with ` \token_to_str:N #1 '
                    }
                    {
                      The ~ provided ~ value ~ doesn't ~ seem ~ to ~
                      be ~ numeric. ~ If ~ you ~ were ~ expecting ~ it ~
                      to ~ be ~ parsed ~ as ~ a ~ date/time ~ then ~
                      \bool_if:NTF \l__datatool_parse_datetime_bool
                       {
                         use ~ \token_to_str:N \DTLsetup { datetime=parse-
only } ~
                         or ~ \token_to_str:N \DTLsetup { datetime=reformat }
                       }
                       {
                        check ~ the ~ original ~ source ~ matched ~
                        the ~ correct ~ format
                       }
                    }
                  }
                  {
                    \int_set:Nn \@dtl@datatype { \DTLdatumtype { #1 } }
                    \int_case:nn { \@dtl@datatype }
                    {
                      { \c_datatool_integer_int }
                       {
                         \int_set_eq:NN \@dtl@datatype \c_datatool_date_int
                       }
                      { \c_datatool_decimal_int }
                       {
                         \fp_compare:nNnTF
                           { \fp_abs:n { \l__datatool_tmpa_tl } }
                             < { \c_one_fp }
                          {
                            \int_set_eq:NN \@dtl@datatype \c_datatool_time_int
                          }
```

```
      {
        \int_set_eq:NN \@dtl@datatype \c_datatool_datetime_int
      }
   }
}
\int_case:nnF { \@dtl@datatype }
{
  { \c_datatool_datetime_int }
   {
     \PackageWarning { datatool-base }
     {
       No ~ timestamp ~ could ~ be ~ found ~ in ~ the ~
       provided ~ datum ~ command. ~ Value ~ found: ~
       \l__datatool_tmpa_tl . ~ I'm ~ going ~
       to ~ assume ~ this ~ is ~ a ~ Julian ~ Date
     }
    \datatool_from_julian_date:VN
     \l__datatool_tmpa_tl
     \l__datatool_datetime_seq
    \tl_set:Ne \l__datatool_tmpa_tl
     {
       \exp_not:N \DTMsavetimestamp { #2 }
        {
          \datatool_timestamp_to_iso:N
            \l__datatool_datetime_seq
        }
      }
   }
  { \c_datatool_date_int }
   {
     \PackageWarning { datatool-base }
     {
       No ~ timestamp ~ could ~ be ~ found ~ in ~ the ~
       provided ~ datum ~ command. ~ Value ~ found: ~
       \l__datatool_tmpa_tl . ~ I'm ~ going ~
       to ~ assume ~ this ~ is ~ a ~ Julian ~ Day ~ Number
     }
    \int_set:Nn \l__datatool_julian_int
      { \l__datatool_tmpa_tl }
    \datatool_from_julian_day:VNNN
     \l__datatool_julian_int
     \l__datatool_year_int
     \l__datatool_month_int
     \l__datatool_day_int
    \tl_set:Ne \l__datatool_tmpa_tl
     {
       \exp_not:N \DTMsavenoparsedate { #2 }
        { \int_use:N \l__datatool_year_int }
        { \int_use:N \l__datatool_month_int }
        { \int_use:N \l__datatool_day_int }
```

```
            {
              \datatool_julian_day_to_dow:n
               { \l__datatool_julian_int }
            }
          }
        }
       { \c_datatool_time_int }
        {
          \PackageWarning { datatool-base }
          {
            No ~ timestamp ~ could ~ be ~ found ~ in ~ the ~
            provided ~ datum ~ command. ~ Value ~ found: ~
            \l__datatool_tmpa_tl . ~ I'm ~ going ~
            to ~ assume ~ this ~ is ~ a ~ Julian ~
            time ~ of ~ day ~ fraction
          }
         \fp_set:Nn \l__datatool_julian_fp
           { \l__datatool_tmpa_tl }
         \datatool_from_julian_time:nNNN
          { \l__datatool_julian_fp }
          \l__datatool_hour_int
          \l__datatool_minute_int
          \l__datatool_second_int
         \tl_set:Ne \l__datatool_tmpa_tl
          {
            \exp_not:N \DTMsavetime { #2 }
             {
              \int_use:N \l__datatool_year_int
              \c_colon_str
              \int_use:N \l__datatool_month_int
              \c_colon_str
              \int_use:N \l__datatool_day_int
             }
           }
        }
    }
    {

      \PackageError { datatool-base }
       {
         No ~ timestamp ~ could ~ be ~ found ~ in ~ the ~
         provided ~ datum ~ command. ~ Value ~ found: ~
         \l__datatool_tmpa_tl
       }
       {
         The ~ datum ~ value ~ supplied ~ can't ~ be ~
         interpreted ~ as ~ a ~ timestamp, ~ date ~ or ~ time.
       }
    }
}
```

```
          }
        }
      }
     \exp_after:wN
    \group_end: \l__datatool_tmpa_tl
   }
   {
    \PackageError { datatool-base }
     {
       \token_to_str:N \DTLdatumtoDTM ~ \c_space_tl ~
       can't ~ save ~ time ~ stamp. ~
       Command ~ \token_to_str:N \DTMsavetimestamp ~ \c_space_tl ~
       undefined ~ (datetime2 ~ required)
     }
     { You ~ need ~ to ~ load ~ datetime2.sty }
   }
}
```

Formatting Syntax: ⟨*datum-cs*⟩⟨*value-cs*⟩{⟨*string value*⟩}

Used to set the value part of a temporal datum and the update token list variable. If the {⟨*string value*⟩} argument is empty, the string part is obtained by formatting the temporal value.

```
\cs_new:Nn \__datatool_set_datetime_value:NNn
 {
   \int_case:nn { \@dtl@datatype }
    {
      { \c_datatool_datetime_int }
      {
        \tl_set:Ne #2
         {
           \exp_not:N \DTLtemporalvalue
            { \fp_use:N \l__datatool_julian_fp }
            {
              \__datatool_tm_iso:
            }
         }
        \tl_if_empty:nTF { #3 }
         {
           \__datatool_set_datum:NeVnn #1
               {
                 \exp_not:N \DataToolDateTimeFmt
                  {
                    { \__datatool_tm_yr: }
                    { \__datatool_tm_mn: }
                    { \__datatool_tm_dy: }
                    { \__datatool_tm_dow: }
                  }
                  {
                    { \__datatool_tm_hr: }
```

```
              { \__datatool_tm_mi: }
              { \__datatool_tm_se: }
            }
            {
              { \__datatool_tm_tzh: }
              { \__datatool_tm_tzm: }
            }
          }
          #2
          { }
          { \c_datatool_datetime_int }
    }
    {
      \__datatool_set_datum:NnVnn #1
        { #3 }
        #2
        { }
        { \c_datatool_datetime_int }
    }
}
{ \c_datatool_date_int }
{
  \tl_set:Ne #2
    {
      \exp_not:N \DTLtemporalvalue
       { \int_use:N \l__datatool_julian_int }
       {
         \__datatool_tm_iso_date:
       }
    }
  \tl_if_empty:nTF { #3 }
    {
      \__datatool_set_datum:NeVnn #1
          {
            \exp_not:N \DataToolDateFmt
             { \__datatool_tm_yr: }
             { \__datatool_tm_mn: }
             { \__datatool_tm_dy: }
             { \__datatool_tm_dow: }
          }
          #2
          { }
          { \c_datatool_date_int }
    }
    {
      \__datatool_set_datum:NnVnn #1
         { #3 }
         #2
         { }
         { \c_datatool_date_int }
```

```
        }
      }
      { \c_datatool_time_int }
      {
        \tl_set:Ne #2
        {
          \exp_not:N \DTLtemporalvalue
           { \fp_use:N \l__datatool_time_fp }
           {
             \__datatool_tm_iso_time:
           }
        }
        \tl_if_empty:nTF { #3 }
        {
          \__datatool_set_datum:NeVnn #1
              {
                \exp_not:N \DataToolTimeFmt
                 { \__datatool_tm_hr: }
                 { \__datatool_tm_mi: }
                 { \__datatool_tm_se: }
              }
              #2
              { }
              { \c_datatool_time_int }
        }
        {
          \__datatool_set_datum:NnVnn #1
              { #3 }
              #2
              { }
              { \c_datatool_time_int }
        }
      }
    }
  }
```

Constants and Scratch Variables Scratch sequence for timestamps:

```
\seq_new:N \l__datatool_datetime_seq
```

Individual integer elements of a timestamp.

```
\int_new:N \l__datatool_year_int
\int_new:N \l__datatool_month_int
\int_new:N \l__datatool_day_int
\int_new:N \l__datatool_hour_int
\int_new:N \l__datatool_minute_int
\int_new:N \l__datatool_second_int
\int_new:N \l__datatool_tzhour_int
\int_new:N \l__datatool_tzminute_int
```

Integer Julian Day Number:

```
\int_new:N \l__datatool_julian_int
\int_new:N \l__datatool_local_julian_int
```

Floating point values for the Julian date and time of day fraction:

```
\fp_new:N \l__datatool_julian_fp
\fp_new:N \l__datatool_time_fp
```

Re-initialise date/time individual scratch variables. That is, set them all back to zero.

```
\cs_new:Nn \__datatool_tm_var_init:
{
  \int_zero:N \l__datatool_year_int
  \int_zero:N \l__datatool_month_int
  \int_zero:N \l__datatool_day_int
  \int_zero:N \l__datatool_hour_int
  \int_zero:N \l__datatool_minute_int
  \int_zero:N \l__datatool_second_int
  \int_zero:N \l__datatool_tzhour_int
  \int_zero:N \l__datatool_tzminute_int
  \int_zero:N \l__datatool_julian_int
  \fp_zero:N \l__datatool_julian_fp
  \fp_zero:N \l__datatool_time_fp
}
```

Date time stamp regular expression. Time zone may be omitted. Separator between date and time may be either space or “T”.

```
\regex_const:Nn \c_datatool_timestamp_regex
 {
   \A \s*
   ( \d+ ) \x { 2D } ( \d{2} ) \x { 2D } ( \d{2} )
   (?: \x { 54 } | \s+ )
   ( \d{2} ) \x { 3A } ( \d{2} ) \x { 3A } ( \d{2} )
   ( \x { 5A } | [\+\-]? \d{2} \x { 3A } ? \d{2} ) ?
   \s* \Z
 }
```

Date only.

```
\regex_const:Nn \c_datatool_date_regex
 {
   \A \s*
   ( \d+ ) \x { 2D } ( \d{2} )\x { 2D } ( \d{2} )
   \s* \Z
 }
```

Time only. Seconds are optional.

```
\regex_const:Nn \c_datatool_time_regex
 {
   \A \s*
   ( [\+\-]? \d{2} ) \x { 3A } ( \d{2} ) (?: \x { 3A } ( \d{2} ) )?
   \s* \Z
 }
```

Maybe timestamp or just date or just time:

```
\regex_const:Nn \c_datatool_temporal_regex
 {
   (?:
      (?: \d+ ) \x { 2D } (?: \d{2} ) \x { 2D } (?: \d{2} )
      (?:
         (?: \x { 54 } | \s+ )
         (?: \d{2} ) \x { 3A } (?: \d{2} ) \x { 3A } (?: \d{2} )
         (?: \x { 5A } | [\+\-]? \d{2} \x { 3A } ? \d{2} ) ?
      ) ?
   ) |
   (?:
     (?: \d{2} ) \x { 3A } (?: \d{2} ) (?: \x { 3A } (?: \d{2} ) )?
   )
 }
```

Timestamp Sequences Time stamp sequences are just a nine-item sequence, where each item is a numeric value. All but the year (item 1) and day of week (item 9) should be correctly zero-padded. The year should include the century. For example, if the year is given as 24 then that means 0024 not 2024. The day of week item may be negative to indicate a missing value.

The following commands are simply wrappers that use normal sequence commands.

```
\cs_new:Nn \datatool_timestamp_new:N
{
  \seq_new:N #1
  \datatool_timestamp_zero:N #1
}
```

Constant representing 0000-00-00T00:00:00+00:00 (used for initialisation).

```
\seq_const_from_clist:Nn \c_datatool_timestamp_zero_seq
 { 0000 , 00 , 00 , 00 , 00 , 00 , +00 , 00, -1 }
```

Resets all elements of a timestamp sequence.

```
\cs_new:Nn \datatool_timestamp_zero:N
{
  \seq_set_eq:NN #1 \c_datatool_timestamp_zero_seq
}
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_seq_init:
{
  \datatool_timestamp_zero:N
    \l__datatool_datetime_seq
}
```

The following commands assume the provided sequence has been correctly defined with nine numeric items, as described above. Check for empty argument as regex match can have optional parts.

Extract year from a timestamp sequence. (First item.)

```
\cs_new:Nn \datatool_timestamp_get_year:N
 {
```

```
    \seq_item:Nn #1 { 1 }
  }
```
Shortcut for internal scratch variable.
```
 \cs_new:Nn \__datatool_tm_yr:
  {
     \datatool_timestamp_get_year:N
       \l__datatool_datetime_seq
  }
```
Set year (item 1). If empty, assume current year.
```
 \cs_new:Nn \datatool_timestamp_set_year:Nn
  {
    \tl_if_empty:nTF { #2 }
     {
       \exp_args:NNne \seq_set_item:Nnn #1
         { 1 } { \int_use:N \c_sys_year_int }
     }
     {
       \seq_set_item:Nnn #1 { 1 } { #2 }
     }
  }
 \cs_generate_variant:Nn \datatool_timestamp_set_year:Nn
   { Ne, NV }
```
As above but add century if between 0 and 99.
```
 \cs_new:Nn \datatool_timestamp_set_year_add_century:Nn
  {
    \tl_if_empty:nTF { #2 }
     {
       \exp_args:NNne \seq_set_item:Nnn #1
         { 1 } { \int_use:N \c_sys_year_int }
     }
     {
       \bool_lazy_or:nnTF
          { \int_compare_p:nNn { #2 } < { \c_zero_int } }
          { \int_compare_p:nNn { #2 } > { 99 } }
        {
          \seq_set_item:Nnn #1 { 1 } { #2 }
        }
        {
          \datatool_timestamp_set_year:Ne #1
           {
            \int_eval:n
             {
               ( \int_div_truncate:nn { \c_sys_year_int } { 100 } ) * 100
               + #2
             }
           }
        }
     }
```

```
  }
\cs_generate_variant:Nn \datatool_timestamp_set_year_add_century:Nn
  { Ne, NV }
```
Shortcut for internal scratch variable.
```
\cs_new:Nn \__datatool_tm_set_yr:n
 {
   \datatool_timestamp_set_year:Nn
     \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_yr:n { e }
```
Extract month from sequence. (Second item.)
```
\cs_new:Nn \datatool_timestamp_get_month:N
 {
   \seq_item:Nn #1 { 2 }
 }
```
Shortcut for internal scratch variable.
```
\cs_new:Nn \__datatool_tm_mn:
 {
   \datatool_timestamp_get_month:N
     \l__datatool_datetime_seq
 }
```
Set month (item 2):
```
\cs_new:Nn \datatool_timestamp_set_month:Nn
 {
   \tl_if_empty:nTF { #2 }
    {
      \seq_set_item:Nnn #1 { 2 } { 00 }
    }
    {
      \exp_args:NNne \seq_set_item:Nnn
       #1 { 2 } { \datatool_two_digits:n { #2 } }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_set_month:Nn
  { Ne, NV }
```
Shortcut for internal scratch variable.
```
\cs_new:Nn \__datatool_tm_set_mn:n
 {
   \datatool_timestamp_set_month:Nn
    \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_mn:n { e }
```
Extract day from sequence. (Third item.)
```
\cs_new:Nn \datatool_timestamp_get_day:N
 {
   \seq_item:Nn #1 { 3 }
 }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_dy:
 {
   \datatool_timestamp_get_day:N
     \l__datatool_datetime_seq
 }
```

Set day (item 3):

```
\cs_new:Nn \datatool_timestamp_set_day:Nn
 {
   \tl_if_empty:nTF { #2 }
    {
      \seq_set_item:Nnn #1 { 3 } { 00 }
    }
    {
      \exp_args:NNne \seq_set_item:Nnn
       #1 { 3 } { \datatool_two_digits:n { #2 } }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_set_day:Nn
  { Ne, NV }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_set_dy:n
 {
   \datatool_timestamp_set_day:Nn
    \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_dy:n { e }
```

Extract hour from sequence. (Fourth item.)

```
\cs_new:Nn \datatool_timestamp_get_hour:N
 {
   \seq_item:Nn #1 { 4 }
 }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_hr:
 {
   \datatool_timestamp_get_hour:N
     \l__datatool_datetime_seq
 }
```

Set hour (item 4):

```
\cs_new:Nn \datatool_timestamp_set_hour:Nn
 {
   \tl_if_empty:nTF { #2 }
    {
      \seq_set_item:Nnn #1 { 4 } { 00 }
    }
    {
```

```
      \exp_args:NNne \seq_set_item:Nnn
       #1 { 4 } { \datatool_two_digits:n { #2 } }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_set_hour:Nn
  { Ne, NV }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_set_hr:n
 {
   \datatool_timestamp_set_hour:Nn
     \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_hr:n { e }
```

Extract minute from sequence. (Fifth item.)

```
\cs_new:Nn \datatool_timestamp_get_minute:N
 {
   \seq_item:Nn #1 { 5 }
 }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_mi:
 {
   \datatool_timestamp_get_minute:N
     \l__datatool_datetime_seq
 }
```

Set minute (item 5):

```
\cs_new:Nn \datatool_timestamp_set_minute:Nn
 {
   \tl_if_empty:nTF { #2 }
    {
      \seq_set_item:Nnn #1 { 5 } { 00 }
    }
    {
      \exp_args:NNne \seq_set_item:Nnn
       #1 { 5 } { \datatool_two_digits:n { #2 } }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_set_minute:Nn
  { Ne, NV }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_set_mi:n
 {
   \datatool_timestamp_set_minute:Nn
     \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_mi:n { e }
```

Extract second from sequence. (Sixth item.)

```
\cs_new:Nn \datatool_timestamp_get_second:N
 {
   \seq_item:Nn #1 { 6 }
 }
```
Shortcut for internal scratch variable.
```
\cs_new:Nn \__datatool_tm_se:
 {
   \datatool_timestamp_get_second:N
     \l__datatool_datetime_seq
 }
```
Set second (item 6):
```
\cs_new:Nn \datatool_timestamp_set_second:Nn
 {
   \tl_if_empty:nTF { #2 }
    {
      \seq_set_item:Nnn #1 { 6 } { 00 }
    }
    {
      \exp_args:NNne \seq_set_item:Nnn
       #1 { 6 } { \datatool_two_digits:n { #2 } }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_set_second:Nn
  { Ne, NV }
```
Shortcut for internal scratch variable.
```
\cs_new:Nn \__datatool_tm_set_se:n
 {
   \datatool_timestamp_set_second:Nn
     \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_se:n { e }
```
Extract time zone hour from sequence. (Seventh item.)
```
\cs_new:Nn \datatool_timestamp_get_tzhour:N
 {
   \seq_item:Nn #1 { 7 }
 }
```
Shortcut for internal scratch variable.
```
\cs_new:Nn \__datatool_tm_tzh:
 {
   \datatool_timestamp_get_tzhour:N
    \l__datatool_datetime_seq
 }
```
Set time zone hour (item 7):
```
\cs_new:Nn \datatool_timestamp_set_tzhour:Nn
 {
   \tl_if_empty:nTF { #2 }
```

```
    {
      \seq_set_item:Nnn #1 { 7 } { +00 }
    }
    {
      \exp_args:NNne \seq_set_item:Nnn
       #1 { 7 }
       { \datatool_signed_two_digits:n { #2 } }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_set_tzhour:Nn
  { Ne, NV }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_set_tzh:n
 {
   \datatool_timestamp_set_tzhour:Nn
     \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_tzh:n { e }
```

Extract time zone minute from sequence. (Eighth item.)

```
\cs_new:Nn \datatool_timestamp_get_tzminute:N
 {
   \seq_item:Nn #1 { 8 }
 }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_tzm:
 {
   \datatool_timestamp_get_tzminute:N
    \l__datatool_datetime_seq
 }
```

Set time zone minute (item 8):

```
\cs_new:Nn \datatool_timestamp_set_tzminute:Nn
 {
   \tl_if_empty:nTF { #2 }
    {
      \seq_set_item:Nnn #1 { 8 } { 00 }
    }
    {
      \exp_args:NNne \seq_set_item:Nnn
       #1 { 8 } { \datatool_two_digits:n { #2 } }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_set_tzminute:Nn
  { Ne, NV }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_set_tzm:n
 {
   \datatool_timestamp_set_tzminute:Nn
```

```
    \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_tzm:n { e }
```

Extract day of week from sequence. (Ninth item.)

```
\cs_new:Nn \datatool_timestamp_get_dow:N
 {
   \seq_item:Nn #1 { 9 }
 }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_dow:
 {
   \datatool_timestamp_get_dow:N
    \l__datatool_datetime_seq
 }
```

Set day of week (item 9):

```
\cs_new:Nn \datatool_timestamp_set_dow:Nn
 {
   \tl_if_empty:nTF { #2 }
    {
      \seq_set_item:Nnn #1 { 9 } { -1 }
    }
    {
      \exp_args:NNne \seq_set_item:Nnn
       #1 { 9 } { #2 }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_set_dow:Nn
  { Ne, NV }
```

Shortcut for internal scratch variable.

```
\cs_new:Nn \__datatool_tm_set_dow:n
 {
   \datatool_timestamp_set_dow:Nn
     \l__datatool_datetime_seq { #1 }
 }
\cs_generate_variant:Nn \__datatool_tm_set_dow:n { e, V }
```

Convert timestamp sequence to ISO string. Note that the supplied argument must be a timestamp sequence. That is, a sequence with 8 numeric items that have already been correctly zero-padded.

```
\cs_new:Nn \datatool_timestamp_to_iso:N
 {
   \datatool_timestamp_get_year:N #1
   -
   \datatool_timestamp_get_month:N #1
   -
   \datatool_timestamp_get_day:N #1
   T
   \datatool_timestamp_get_hour:N #1
```

```
    \c_colon_str
    \datatool_timestamp_get_minute:N #1
    \c_colon_str
    \datatool_timestamp_get_second:N #1
    \datatool_timestamp_get_tzhour:N #1
    \c_colon_str
    \datatool_timestamp_get_tzminute:N #1
  }
```

Shortcut for internal scratch variable.

```
 \cs_new:Nn \__datatool_tm_iso:
  {
    \datatool_timestamp_to_iso:N
      \l__datatool_datetime_seq
  }
```

Just the year, month and date:

```
 \cs_new:Nn \datatool_timestamp_to_iso_date:N
  {
    \datatool_timestamp_get_year:N #1
    -
    \datatool_timestamp_get_month:N #1
    -
    \datatool_timestamp_get_day:N #1
  }
```

Shortcut for internal scratch variable.

```
 \cs_new:Nn \__datatool_tm_iso_date:
  {
    \datatool_timestamp_to_iso_date:N
      \l__datatool_datetime_seq
  }
```

Just the time (hours, minutes and seconds):

```
 \cs_new:Nn \datatool_timestamp_to_iso_time:N
  {
    \datatool_timestamp_get_hour:N #1
    \c_colon_str
    \datatool_timestamp_get_minute:N #1
    \c_colon_str
    \datatool_timestamp_get_second:N #1
  }
```

Shortcut for internal scratch variable.

```
 \cs_new:Nn \__datatool_tm_iso_time:
  {
    \datatool_timestamp_to_iso_time:N
      \l__datatool_datetime_seq
  }
```

```
\datatool_timestamp_to_datetime_datum:NnN
⟨timestamp seq-var⟩
{⟨Julian Date⟩}
⟨datum cs⟩
```

Convert a timestamp sequence variable to a datetime datum variable.

```
\cs_new:Nn \datatool_timestamp_to_datetime_datum:NnN
 {
   \tl_set:Ne #3
    {
      \exp_not:N \__datatool_datum:nnnn
       {% formatted string
         \exp_not:N \DataToolDateTimeFmt
           {
             { \datatool_timestamp_get_year:N #1 }
             { \datatool_timestamp_get_month:N #1 }
             { \datatool_timestamp_get_day:N #1 }
             { \datatool_timestamp_get_dow:N #1 }
           }
           {
             { \datatool_timestamp_get_hour:N #1 }
             { \datatool_timestamp_get_minute:N #1 }
             { \datatool_timestamp_get_second:N #1 }
           }
           {
             { \datatool_timestamp_get_tzhour:N #1 }
             { \datatool_timestamp_get_tzminute:N #1 }
           }
       }
       {% value
         \exp_not:N \DTLtemporalvalue
          { #2 }
          {
            \datatool_timestamp_to_iso:N #1
          }
       }
       { }% no currency symbol
       { \exp_not:N \c_datatool_datetime_int }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_to_datetime_datum:NnN
 { NeN, NVN }
```

```
\datatool_timestamp_to_date_datum:NnN
⟨timestamp seq-var⟩
{⟨Julian Day⟩}
⟨datum cs⟩
```

Convert a timestamp sequence variable to a date datum variable.

```
\cs_new:Nn \datatool_timestamp_to_date_datum:NnN
 {
   \tl_set:Ne #3
    {
      \exp_not:N \__datatool_datum:nnnn
       {% formatted string
         \exp_not:N \DataToolDateFmt
           { \datatool_timestamp_get_year:N #1 }
           { \datatool_timestamp_get_month:N #1 }
           { \datatool_timestamp_get_day:N #1 }
           {
             \datatool_julian_day_to_dow:n { #2 }
           }
       }
       {% value
         \exp_not:N \DTLtemporalvalue
          { #2 }
          {
            \datatool_timestamp_to_iso_date:N #1
          }
       }
       { }% no currency symbol
       { \exp_not:N \c_datatool_date_int }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_to_date_datum:NnN
 { NeN, NVN }
```

```
\datatool_timestamp_to_time_datum:NnN
⟨timestamp seq-var⟩
{⟨Julian Time⟩}
⟨datum cs⟩
```

Convert a timestamp sequence variable to a time datum variable.

```
\cs_new:Nn \datatool_timestamp_to_time_datum:NnN
 {
   \tl_set:Ne #3
    {
      \exp_not:N \__datatool_datum:nnnn
       {% formatted string
         \exp_not:N \DataToolTimeFmt
```

```
            { \datatool_timestamp_get_hour:N #1 }
            { \datatool_timestamp_get_minute:N #1 }
            { \datatool_timestamp_get_second:N #1 }
        }
        {% value
          \exp_not:N \DTLtemporalvalue
           { #2 }
           {
             \datatool_timestamp_to_iso_time:N #1
           }
        }
        { }% no currency symbol
        { \exp_not:N \c_datatool_time_int }
    }
 }
\cs_generate_variant:Nn \datatool_timestamp_to_time_datum:NnN
 { NeN, NVN }
```

Calculations Convert Unix time to Julian date.

```
\cs_new:Nn \datatool_unix_to_julian:n
 {
   \fp_eval:n { #1 / 86400 + 2440587.5 }
 }
```

Convert a Julian date to Unix time. Note that the Unix time may be too large to be a TeX integer. The result will be a decimal.

```
\cs_new:Nn \datatool_julian_to_unix:n
 {
   \fp_eval:n { (#1 - 2440587.5 ) * 86400 }
 }
```

\datatool_julian_day_to_dow:n
{⟨*JDN*⟩}

Obtains the day of week from the integer Julian Day Number.

```
\cs_new:Nn \datatool_julian_day_to_dow:n
 {
   \int_mod:nn { #1 } { 7 }
 }
```

\datatool_calc_julian_date:NN
⟨*result fp-var*⟩
⟨*timestamp seq-var*⟩

Calculate the Julian Date for the timestamp data contained in the timestamp sequence variable ⟨*timestamp seq-var*⟩ and store the resulting value in the floating-point variable ⟨*result fp-var*⟩.

```
\cs_new:Nn \datatool_calc_julian_date:NN
 {
   \int_compare:nNnTF { \seq_count:N #2 } = { 8 }
    {
      \seq_set_eq:NN \l__datatool_datetime_seq #2
      \_datatool_calc_julian_date:
      \fp_set_eq:NN #1 \l__datatool_julian_fp
    }
    {
      \PackageError { datatool-base }
        {
           Not ~ a ~ timestamp ~ sequence: ~
           \token_to_str:N #2
        }
        {
           The ~ second ~ argument ~ of ~
           \token_to_str:N \datatool_calc_julian_date:NN ~
           \c_string_tl ~ must ~ be ~ a ~ sequence ~ containing ~
           8 ~ integer ~ elements
        }
    }
 }
```

\datatool_calc_julian_day:Nnnn ⟨*result fp-var*⟩ {⟨*year*⟩} {⟨*month*⟩} {⟨*day*⟩}

Calculate the Julian Day value from the given year, month and day. NB very sensitive to rounding! Don't rearrange.

```
\cs_new:Nn \datatool_calc_julian_day:Nnnn
{
  \int_set:Nn #1
   {
     #4 - 32075
      +
         \int_div_truncate:nn
          {
             1461
             *
              ( #2 + 4800 + \int_div_truncate:nn { #3 - 14 } { 12 } )
          }
          { 4 }
      +
         \int_div_truncate:nn
          {
            367
             *
             ( #3 - 2
               - \int_div_truncate:nn { #3 - 14 } { 12 } * 12 )
          }
```

```
          { 12 }
      -
         \int_div_truncate:nn
          {
            3
            *
            (
              \int_div_truncate:nn
              {
                #2 + 4900
                + \int_div_truncate:nn { #3 - 14 } { 12 }
              }
              { 100 }
            )
          }
          { 4 }
   }
}
\cs_generate_variant:Nn \datatool_calc_julian_day:Nnnn
  { NVVV , Neee }
```

\datatool_from_julian_time:nNNN
{⟨*JT*⟩} ⟨*hour int var*⟩ ⟨*minute int var*⟩ ⟨*second int var*⟩

Calculate the time of day from the Julian time decimal fraction.

```
\cs_new:Nn \datatool_from_julian_time:nNNN
{
  \int_set:Nn #4 { 43200 + \fp_to_int:n { #1 * 86400 } }
  \int_set:Nn #3
    {
      \int_mod:nn { \int_div_truncate:nn { #4 } { 60 } } { 60 }
    }
  \int_set:Nn #2
    {
      \int_div_truncate:nn { #4 } { 3600 }
    }
  \int_set:Nn #4 { \int_mod:nn { #4 } { 60 } }
}
\cs_generate_variant:Nn \datatool_from_julian_time:nNNN
  { VNNN, eNNN }
```

\datatool_from_julian_day:nNNN
{⟨*JDN*⟩} ⟨*year int var*⟩ ⟨*month int var*⟩ ⟨*day int var*⟩

Calculate the year, month and day from the Julian Day Number.

```
\cs_new:Nn \datatool_from_julian_day:nNNN
{
  \int_set:Nn \l__datatool_tmpa_int { #1 + 68569 }
```

```
  \int_set:Nn \l__datatool_tmpb_int
    { \int_div_truncate:nn { 4 * \l__datatool_tmpa_int } { 146097 } }
  \int_set:Nn \l__datatool_tmpa_int
    {
      \l__datatool_tmpa_int
       - \int_div_truncate:nn { 146097 * \l__datatool_tmpb_int + 3 } { 4 }
    }
  \int_set:Nn \l__datatool_tmpc_int
    {
      \int_div_truncate:nn { 4000 * \l__datatool_tmpa_int + 1 } { 1461001 }
    }
  \int_set:Nn \l__datatool_tmpa_int
    {
      \l__datatool_tmpa_int
      - \int_div_truncate:nn { 1461 * \l__datatool_tmpc_int } { 4 }
      + 31
    }
  \int_set:Nn \l__datatool_tmpd_int
    {
      \int_div_truncate:nn { 80 * \l__datatool_tmpa_int } { 2447 }
    }
  \int_set:Nn #4
    {
      \l__datatool_tmpa_int
      - \int_div_truncate:nn { 2447 * \l__datatool_tmpd_int } { 80 }
    }
  \int_set:Nn \l__datatool_tmpa_int
    {
      \int_div_truncate:nn { \l__datatool_tmpd_int } { 11 }
    }
  \int_set:Nn \l__datatool_tmpd_int
    {
      \l__datatool_tmpd_int + 2 - 12 * \l__datatool_tmpa_int
    }
  \int_set:Nn \l__datatool_tmpc_int
    {
      100 * ( \l__datatool_tmpb_int - 49 )
      + \l__datatool_tmpc_int + \l__datatool_tmpa_int
    }
  \int_set_eq:NN #2 \l__datatool_tmpc_int
  \int_set_eq:NN #3 \l__datatool_tmpd_int
}
\cs_generate_variant:Nn \datatool_from_julian_day:nNNN
  { VNNN , eNNN }
```

{⟨*JD*⟩} ⟨*year int var*⟩ ⟨*month int var*⟩ ⟨*day int var*⟩ ⟨*hour int var*⟩ ⟨*minute int var*⟩ ⟨*second int var*⟩

Calculate the timestamp (UTC+0) from the Julian Date.

```
\cs_new:Nn \datatool_from_julian_date:nNNNNNN
 {
   \int_set:Nn \l__datatool_julian_int
      { \fp_to_int:n { round ( #1 ) } }
   \datatool_from_julian_day:nNNN
      { \l__datatool_julian_int }
      #2 #3 #4
   \exp_args:Ne \datatool_from_julian_time:nNNN
      { \fp_eval:n { #1 - \l__datatool_julian_int } }
      #5 #6 #7
 }
\cs_generate_variant:Nn \datatool_from_julian_date:nNNNNNN
 { VNNNNNN, eNNNNNN }
```

`\datatool_from_julian_date:nN {⟨JD⟩} ⟨timestamp seq-var⟩`

As above but the result should be stored in a timestamp sequence variable.

```
\cs_new:Nn \datatool_from_julian_date:nN
 {
   \datatool_from_julian_date:nNNNNNN
    { #1 }
    \l__datatool_year_int
    \l__datatool_month_int
    \l__datatool_day_int
    \l__datatool_hour_int
    \l__datatool_minute_int
    \l__datatool_second_int
   \datatool_timestamp_zero:N #2
   \datatool_timestamp_set_year:NV #2 \l__datatool_year_int
   \datatool_timestamp_set_month:NV #2 \l__datatool_month_int
   \datatool_timestamp_set_day:NV #2 \l__datatool_day_int
   \datatool_timestamp_set_hour:NV #2 \l__datatool_hour_int
   \datatool_timestamp_set_minute:NV #2 \l__datatool_minute_int
   \datatool_timestamp_set_second:NV #2 \l__datatool_second_int
   \datatool_timestamp_set_dow:Ne #2
    {
      \datatool_julian_day_to_dow:n
       { \fp_eval:n { round ( #1 ) } }
    }
 }
\cs_generate_variant:Nn \datatool_from_julian_date:nN { VN , eN }
```

Internal Calculation Functions Calculate the Julian day from values provided by the scratch date and time variables (which needs to be set first). Time zone adjustment needed:

```
\cs_new:Nn \__datatool_calc_julian_day_tmz:
```

```
{
  \__datatool_calc_julian_day:
```

Keep a record of the unadjusted Julian day:

```
  \int_set_eq:NN \l__datatool_local_julian_int \l__datatool_julian_int
```

Implement time zone shift:

```
  \int_if_zero:nF { \l__datatool_tzminute_int }
    {
      \int_sub:Nn \l__datatool_minute_int { \l__datatool_tzminute_int }
      \int_compare:nNnTF { \l__datatool_minute_int } < { \c_zero_int }
       {
         \int_add:Nn \l__datatool_minute_int { 60 }
         \int_decr:N \l__datatool_hour_int
       }
       {
         \int_compare:nNnT { \l__datatool_minute_int } > { 59 }
          {
            \int_sub:Nn \l__datatool_minute_int { 60 }
            \int_incr:N \l__datatool_hour_int
          }
       }
    }
  \int_if_zero:nF { \l__datatool_tzhour_int }
    {
      \int_sub:Nn \l__datatool_hour_int { \l__datatool_tzhour_int }
    }
  \int_compare:nNnTF { \l__datatool_hour_int } < { \c_zero_int }
    {
      \int_add:Nn \l__datatool_hour_int { 23 }
      \int_decr:N \l__datatool_julian_int
    }
    {
      \int_compare:nNnT { \l__datatool_hour_int } > { 23 }
        {
          \int_decr:Nn \l__datatool_hour_int { 24 }
          \int_incr:N \l__datatool_julian_int
        }
    }
}
```

Calculate the Julian day number from just the year, month and day scratch variables (noon UTC+0).

```
\cs_new:Nn \__datatool_calc_julian_day:
{
  \datatool_calc_julian_day:Nnnn
    \l__datatool_julian_int
    { \l__datatool_year_int }
    { \l__datatool_month_int }
    { \l__datatool_day_int }
}
```

Calculate the Julian time, which is a fraction of the day starting from 12noon. (Note that this isn't the same as TeX's `\time` value, which is an integer number of minutes since midnight.) For example, 6am is $(6-12)/24+0/1440+0/86400=-0.25$ and 6pm is $(18-12)/24+0/1440+0/86400=+0.25$.

```
\cs_new:Nn \__datatool_calc_julian_time:
{
  \fp_set:Nn \l__datatool_time_fp
   {
      ( \l__datatool_hour_int - 12) / 24
    + \l__datatool_minute_int / 1440
    + \l__datatool_second_int / 86400
   }
}
```

The Julian date can then be obtained by adding the Julian day and Julian time.

Perform the reverse:

```
\cs_new:Nn \__datatool_from_julian_time:
{
  \__datatool_from_julian_time:n { \l__datatool_time_fp }
}
\cs_new:Nn \__datatool_from_julian_time:n
{
  \datatool_from_julian_time:nNNN { #1 }
    \l__datatool_hour_int
    \l__datatool_minute_int
    \l__datatool_second_int
}
```

Convert Julian Day number (integer) to Gregorian year, month and day.

```
\cs_new:Nn \__datatool_from_julian_day:
{
  \__datatool_from_julian_day:n { \l__datatool_julian_int }
}
\cs_new:Nn \__datatool_from_julian_day:n
{
   \datatool_from_julian_day:nNNN { #1 }
    \l__datatool_year_int
    \l__datatool_month_int
    \l__datatool_day_int
}
```

Convert from decimal Julian date to UTC+0.

```
\cs_new:Nn \__datatool_from_julian_date:
{
  \__datatool_from_julian_date:n { \l__datatool_julian_int }
}
\cs_new:Nn \__datatool_from_julian_date:n
{
   \datatool_from_julian_date:nNNNNNN { #1 }
    \l__datatool_year_int
```

```
    \l__datatool_month_int
    \l__datatool_day_int
    \l__datatool_hour_int
    \l__datatool_minute_int
    \l__datatool_second_int
}
```

Parsing Timestamps, dates and times must be in ISO format in order to be parsed correctly unless support is provided by localisation files.

```
\datatool_parse_timestamp:NnTF
⟨timestamp seq var⟩
{⟨ISO⟩}
{⟨true⟩} {⟨false⟩}
```

Parse ⟨*ISO*⟩ timestamp and save the numeric values in the provided timestamp sequence variable. Does ⟨*true*⟩ if ⟨*ISO*⟩ successfully parsed. Otherwise does ⟨*false*⟩. The timestamp format should be in the form ⟨*YYYY*⟩-⟨*MM*⟩-⟨*DD*⟩T⟨*hh*⟩:⟨*mm*⟩:⟨*ss*⟩⟨*TZhr*⟩:⟨*TZmin*⟩ The time zone parts may be omitted or may simply be the letter "Z", in which case the time zone is UTC+0. The "T" separator may be a space instead. Leading and trailing space is ignored.

```
\prg_new_conditional:Npnn \datatool_parse_timestamp:Nn #1 #2
  { T, F, TF }
{
  \__datatool_parse_timestamp:nTF { #2 }
   {
     \seq_set_eq:NN #1 \l__datatool_datetime_seq
     \prg_return_true:
   }
   { \prg_return_false: }
}
```

```
\datatool_parse_date:NnTF
⟨timestamp seq var⟩
{⟨date⟩}
{⟨true⟩} {⟨false⟩}
```

Parses the ⟨*date*⟩ argument, which must be in the form ⟨*YYYY*⟩-⟨*MM*⟩-⟨*DD*⟩ and saves the year, month and date in the provided timestamp sequence variable. Note that the other elements aren't changed unless the sequence was originally empty. Does ⟨*true*⟩ if ⟨*date*⟩ successfully parsed. Otherwise does ⟨*false*⟩.

```
\prg_new_conditional:Npnn \datatool_parse_date:Nn #1 #2
  { T, F, TF }
{
  \__datatool_parse_date:nTF { #2 }
   {
```

```
      \seq_if_empty:NTF #1
       {
         \seq_set_eq:NN #1 \l__datatool_datetime_seq
       }
       {
          \datatool_timestamp_set_year:Ne #1
            { \__datatool_tm_yr: }
          \datatool_timestamp_set_month:Ne #1
            { \__datatool_tm_mn: }
          \datatool_timestamp_set_day:Ne #1
            { \__datatool_tm_day: }
       }
      \prg_return_true:
   }
   { \prg_return_false: }
}
```

\datatool_parse_time:NnTF
⟨*timestamp seq var*⟩
{⟨*time*⟩}
{⟨*true*⟩} {⟨*false*⟩}

Parses the ⟨*time*⟩ argument, which must be in the form ⟨*hh*⟩:⟨*mm*⟩:⟨*ss*⟩ or ⟨*hh*⟩:⟨*mm*⟩ and saves the hour, minute and second (0, if omitted) in the provided timestamp sequence variable. Note that the other elements aren't changed unless the sequence was originally empty. Does ⟨*true*⟩ if ⟨*time*⟩ successfully parsed. Otherwise does ⟨*false*⟩.

```
\prg_new_conditional:Npnn \datatool_parse_time:Nn #1 #2
  { T, F, TF }
{
  \__datatool_parse_time:nTF { #2 }
   {
     \seq_if_empty:NTF #1
      {
        \seq_set_eq:NN #1 \l__datatool_datetime_seq
      }
      {
         \datatool_timestamp_set_hour:Ne #1
           { \__datatool_tm_hr: }
         \datatool_timestamp_set_minute:Ne #1
           { \__datatool_tm_mi: }
         \datatool_timestamp_set_second:Ne #1
           { \__datatool_tm_se: }
      }
     \prg_return_true:
   }
   { \prg_return_false: }
}
```

Parse given timestamp and save values in the datetime scratch sequence variable.

```
\prg_new_conditional:Npnn \__datatool_parse_timestamp:n #1
  { T, F, TF }
{
 \bool_if:NTF \l__datatool_parse_datetime_iso_bool
  {
   \regex_extract_once:NnNTF
     \c_datatool_timestamp_regex { #1 }
     \l_datatool_timestamp_match_seq
    {
```

Initialise all elements to zero.

```
      \__datatool_tm_seq_init:
```

Year:

```
       \__datatool_tm_set_yr:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Month:

```
       \__datatool_tm_set_mn:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Day:

```
       \__datatool_tm_set_dy:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Hour:

```
       \__datatool_tm_set_hr:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
```

Minute:

```
       \__datatool_tm_set_mi:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 6 } }
```

Second:

```
       \__datatool_tm_set_se:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 7 } }
```

Check time zone present.

```
       \tl_if_empty:eF
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 8 } }
        {
          \exp_args:NNe \regex_extract_once:NnNT
             \c_datatool_time_regex
             { \seq_item:Nn \l_datatool_timestamp_match_seq { 8 } }
             \l_datatool_timestamp_match_seq
           {
             \__datatool_tm_set_tzh:e
              { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
             \__datatool_tm_set_tzm:e
              { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
           }
        }
       \prg_return_true:
```

```
    }
    {
     \bool_if:NTF \l__datatool_parse_datetime_regional_bool
      {
        \DTLCurrentLocaleParseTimeStamp
          \l__datatool_datetime_seq
          { #1 }
          { \prg_return_true: }
          { \prg_return_false: }
      }
      { \prg_return_false: }
   }
  }
  {
    \bool_if:NTF \l__datatool_parse_datetime_regional_bool
     {
       \DTLCurrentLocaleParseTimeStamp
         \l__datatool_datetime_seq
         { #1 }
         { \prg_return_true: }
         { \prg_return_false: }
     }
     { \prg_return_false: }
  }
 }
```

rentLocaleParseTimeStamp

\DTLCurrentLocaleParseTimeStamp{⟨*seq*⟩}{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩}

Allow for regional support. If implemented, this should set the given timestamp sequence variable and do ⟨*true*⟩ otherwise it should do ⟨*false*⟩.

```
\newcommand{\DTLCurrentLocaleParseTimeStamp}[4]{#4}
```

Parse a date (no time):

```
\prg_new_conditional:Npnn \__datatool_parse_date:n #1
  { T, F, TF }
{
 \bool_if:NTF \l__datatool_parse_datetime_iso_bool
  {
   \regex_extract_once:NnNTF
     \c_datatool_date_regex { #1 }
     \l_datatool_timestamp_match_seq
    {
```

Initialise all elements to zero:

```
      \__datatool_tm_seq_init:
```

Year:

```
      \__datatool_tm_set_yr:e
```

```
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```
Month:
```
      \__datatool_tm_set_mn:e
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```
Day:
```
      \__datatool_tm_set_dy:e
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
      \prg_return_true:
    }
    {
      \bool_if:NTF \l__datatool_parse_datetime_regional_bool
       {
         \DTLCurrentLocaleParseDate
          \l__datatool_datetime_seq
          { #1 }
          { \prg_return_true: }
          { \prg_return_false: }
       }
       { \prg_return_false: }
    }
  }
  {
    \bool_if:NTF \l__datatool_parse_datetime_regional_bool
     {
       \DTLCurrentLocaleParseDate
        \l__datatool_datetime_seq
        { #1 }
        { \prg_return_true: }
        { \prg_return_false: }
     }
     { \prg_return_false: }
  }
 }
```

TLCurrentLocaleParseDate

\DTLCurrentLocaleParseDate{⟨*seq*⟩}{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩}

Allow for regional support. If implemented, this should set the given date sequence variable and do ⟨*true*⟩ otherwise it should do ⟨*false*⟩.

```
\newcommand{\DTLCurrentLocaleParseDate}[4]{#4}
```

Parse a time (no date):

```
\prg_new_conditional:Npnn \__datatool_parse_time:n #1
  { T, F, TF }
{
 \bool_if:NTF \l__datatool_parse_datetime_iso_bool
  {
```

```
    \regex_extract_once:NnNTF
      \c_datatool_time_regex { #1 }
      \l_datatool_timestamp_match_seq
     {
       \__datatool_tm_seq_init:
```
Hour:
```
       \__datatool_tm_set_hr:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```
Minute:
```
       \__datatool_tm_set_mi:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```
Second:
```
       \__datatool_tm_set_se:e
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
       \prg_return_true:
     }
     {
       \bool_if:NTF \l__datatool_parse_datetime_regional_bool
        {
          \DTLCurrentLocaleParseTime
            \l__datatool_datetime_seq
            { #1 }
            { \prg_return_true: }
            { \prg_return_false: }
        }
        { \prg_return_false: }
     }
    }
    {
     \bool_if:NTF \l__datatool_parse_datetime_regional_bool
      {
        \DTLCurrentLocaleParseTime
          \l__datatool_datetime_seq
          { #1 }
          { \prg_return_true: }
          { \prg_return_false: }
      }
      { \prg_return_false: }
    }
 }
```

`\DTLCurrentLocaleParseTime{`⟨*seq*⟩`}{`⟨*value*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

`\DTLCurrentLocaleParseTime`

Allow for regional support. If implemented, this should set the given date sequence variable and do ⟨*true*⟩ otherwise it should do ⟨*false*⟩.

```
\newcommand{\DTLCurrentLocaleParseTime}[4]{#4}
```

Parse argument to determine if it's a timestamp, date only, time only or none of those. In the last case, false part is done, otherwise `\@dtl@datatype` will be set to the applicable data type, and the Julian Day number, Julian Date decimal and time fraction will be calculated, and then true part is done.

```
\prg_new_conditional:Npnn \__datatool_parse_datetime:n #1
  { T, F, TF }
{
  \seq_clear:N \l__datatool_datetime_seq
  \fp_zero:N \l__datatool_julian_fp
  \fp_zero:N \l__datatool_time_fp
  \int_zero:N \l__datatool_local_julian_int
  \int_zero:N \l__datatool_julian_int
  \__datatool_parse_timestamp:nTF { #1 }
   {
     \@dtl@datatype = \c_datatool_datetime_int
   }
   {
     \__datatool_parse_date:nTF { #1 }
      {
        \@dtl@datatype = \c_datatool_date_int
      }
      {
        \__datatool_parse_time:nT { #1 }
         {
           \@dtl@datatype = \c_datatool_time_int
         }
      }
   }
  \seq_if_empty:NTF \l__datatool_datetime_seq
   { \prg_return_false: }
   {
     \_datatool_calc_julian_date:
     \prg_return_true:
   }
}
```

Calculate the Julian Date from the scratch timestamp sequence.

```
\cs_new:Nn \_datatool_calc_julian_date:
 {
     \int_set:Nn \l__datatool_year_int { \__datatool_tm_yr: }
     \int_set:Nn \l__datatool_month_int { \__datatool_tm_mn: }
     \int_set:Nn \l__datatool_day_int { \__datatool_tm_dy: }
     \int_set:Nn \l__datatool_hour_int { \__datatool_tm_hr: }
     \int_set:Nn \l__datatool_minute_int { \__datatool_tm_mi: }
     \int_set:Nn \l__datatool_second_int { \__datatool_tm_se: }
     \int_set:Nn \l__datatool_tzhour_int { \__datatool_tm_tzh: }
     \int_set:Nn \l__datatool_tzminute_int { \__datatool_tm_tzm: }
```

Calculate the Julian day with time zone adjustment.

```
\__datatool_calc_julian_day_tmz:
```

Update the day of week item in the timestamp sequence.

```
\__datatool_tm_set_dow:e
 {
   \datatool_julian_day_to_dow:n
    { \l__datatool_local_julian_int }
 }
```

Calculate the Julian date. First the time part:

```
\__datatool_calc_julian_time:
```

Then add the day and time values:

```
\fp_set:Nn \l__datatool_julian_fp
 {
   \int_use:N \l__datatool_julian_int
   + \l__datatool_time_fp
 }
}
```

Set the update token variable for a temporal datum

```
\cs_new:Nn \__datatool_set_datetime_value:
 {
   \__datatool_set_datetime_value:n { }
 }
\cs_new:Nn \__datatool_set_datetime_value:n
 {
   \__datatool_set_datetime_value:NNn
       \l__datatool_datum_update_value_tl
       \l__datatool_datum_value_tl
       { #1 }
 }
\cs_new:Nn \__datatool_set_datetime_value:Nn
 {
   \__datatool_set_datetime_value:NNn
       #1 \l__datatool_datum_value_tl { #2 }
 }
```

Some common formats for use with regions. First a property variable to store time-zone mappings.

```
\prop_new:N \l_datatool_timezone_map_prop
\tl_new:N \l_datatool_timezone_map_value_tl
```

The regions can store applicable time zones in the format ⟨*region-tag*⟩ `/` ⟨*zone-id*⟩ (for example, `GB / GMT` and `GB / BST`). The value should be in the form ⟨*sign*⟩⟨*hh*⟩:⟨*mm*⟩.

```
\cs_new:Nn \datatool_region_set_timezone_map:nn
 {
   \prop_put:Nnn \l_datatool_timezone_map_prop { #1 } { #2 }
 }
```

Get the value and store in the scratch token list.

```
\cs_new:Nn \datatool_region_get_timezone_map:n
 {
   \prop_get:NnN \l_datatool_timezone_map_prop
     { #1 }
     \l_datatool_timezone_map_value_tl
 }
```

Add regionless identifiers UTC and Z:

```
\datatool_region_set_timezone_map:nn { UTC } { +00:00 }
\datatool_region_set_timezone_map:nn { Z } { +00:00 }
```

```
\newcommand{\DTLCurrentLocaleGetTimeZoneMap}[1]{
  \tl_set_eq:NN \l_datatool_timezone_map_value_tl \q_no_value
}
```

Similarly for month name to number mappings:

```
\prop_new:N \l_datatool_monthname_map_prop
\tl_new:N \l_datatool_monthname_map_value_tl
```

Set month name mapping:

```
\cs_new:Nn \datatool_region_set_monthname_map:nn
 {
   \prop_put:Nnn \l_datatool_monthname_map_prop { #1 } { #2 }
 }
```

Get the value and store in the scratch token list.

```
\cs_new:Nn \datatool_region_get_monthname_map:n
 {
   \prop_get:NnN \l_datatool_monthname_map_prop
     { #1 }
     \l_datatool_monthname_map_value_tl
 }
```

```
\newcommand{\DTLCurrentLocaleGetMonthNameMap}[1]{
  \tl_set_eq:NN \l_datatool_monthname_map_value_tl \q_no_value
}
```

Day of month regular expression (1-31):

```
\regex_const:Nn \c_datatool_day_of_month_regex
 {
   ( 0?[1-9] | [12][0-9] | 30 | 31 )
 }
```

One or two digit month number (1-12):

```
\regex_const:Nn \c_datatool_month_number_regex
 {
  ( 0?[1-9] | 10 | 11 | 12 )
 }
```

One or two digit hour (0-24):

```
\regex_const:Nn \c_datatool_hour_regex
 {
```

```
   ( 0?[0-9] | 1[0-9] | 2[0-4] )
  }
```

Two digit minute or second (0-59):

```
 \regex_const:Nn \c_datatool_minsec_regex
  {
    [0-5][0-9]
  }
```

Time zone offset, optionally prefixed with GMT or UTC or UT:

```
 \regex_const:Nn \c_datatool_timezone_regex
  {
    (?:GMT|UTC?)?
    ( [\+\-]? \ur{c_datatool_hour_regex} )
    (?: \: ? ( \ur{c_datatool_minsec_regex} ) ) ?
  }
```

Time zone (either uppercase identifier, which will require a mapping, or offset):

```
 \regex_const:Nn \c_datatool_timezone_id_regex
  {
    [A-Z]+ | \ur{c_datatool_timezone_regex}
  }
```

\DTLCurrentLocaleIfpmTF

\DTLCurrentLocaleIfpmTF{⟨*id*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the identifier is recognised as indicating the afternoon, do true, otherwise do false. Localisation support should redefine this as appropriate.

```
 \newcommand \DTLCurrentLocaleIfpmTF [ 3 ]
  {
    \tl_if_eq:nnTF { #1 } { pm } { #2 } { #3 }
  }
```

The following regular expression constants are for common formats. Localisation support may use the applicable expression or provide custom patterns. Note that it's important that the captured groups match the applicable parsing function.

Non-anchored expressions allow them to be used as a sub-expression for timestamp parsing.

Common dd mm yyyy formats.

```
 \regex_const:Nn \c_datatool_slash_ddmmyyyy_date_regex
  {
    ( \ur{c_datatool_day_of_month_regex} )
     \/
    ( \ur{c_datatool_month_number_regex}  )
     \/ ( \d+ )
  }
```

```
 \regex_const:Nn \c_datatool_slash_anchored_ddmmyyyy_date_regex
  {
    \A \s*
```

```
   ( \ur{c_datatool_day_of_month_regex} )
    \/
   ( \ur{c_datatool_month_number_regex}  )
    \/ ( \d+ )
   \s* \Z
 }
\regex_const:Nn \c_datatool_slash_ddmmyy_date_regex
 {
   ( \ur{c_datatool_day_of_month_regex} )
    \/
   ( \ur{c_datatool_month_number_regex}  )
    \/ ( \d{2} )
 }
\regex_const:Nn \c_datatool_slash_anchored_ddmmyy_date_regex
 {
   \A \s*
   ( \ur{c_datatool_day_of_month_regex} )
    \/
   ( \ur{c_datatool_month_number_regex}  )
    \/ ( \d{2} )
   \s* \Z
 }
\regex_const:Nn \c_datatool_hyphen_ddmmyyyy_date_regex
 {
   ( \ur{c_datatool_day_of_month_regex} )
    \-
   ( \ur{c_datatool_month_number_regex}  )
    \- ( \d+ )
 }
\regex_const:Nn \c_datatool_hyphen_anchored_ddmmyyyy_date_regex
 {
   \A \s*
   ( \ur{c_datatool_day_of_month_regex} )
    \-
   ( \ur{c_datatool_month_number_regex}  )
    \- ( \d+ )
   \s* \Z
 }
\regex_const:Nn \c_datatool_hyphen_ddmmyy_date_regex
 {
   ( \ur{c_datatool_day_of_month_regex} )
    \-
   ( \ur{c_datatool_month_number_regex} )
    \- ( \d{2} )
 }
\regex_const:Nn \c_datatool_hyphen_anchored_ddmmyy_date_regex
 {
```

```
    \A \s*
    ( \ur{c_datatool_day_of_month_regex} )
     \-
    ( \ur{c_datatool_month_number_regex} )
     \- ( \d{2} )
    \s* \Z
  }
\regex_const:Nn \c_datatool_dot_ddmmyyyy_date_regex
  {
    ( \ur{c_datatool_day_of_month_regex} )
     \.
    ( \ur{c_datatool_month_number_regex} )
     \. ( \d+ )
  }
\regex_const:Nn \c_datatool_dot_anchored_ddmmyyyy_date_regex
  {
    \A \s*
    ( \ur{c_datatool_day_of_month_regex} )
     \.
    ( \ur{c_datatool_month_number_regex} )
     \. ( \d+ )
    \s* \Z
  }
\regex_const:Nn \c_datatool_dot_ddmmyy_date_regex
  {
    ( \ur{c_datatool_day_of_month_regex} )
     \.
    ( \ur{c_datatool_month_number_regex} )
     \. ( \d{2} )
  }
\regex_const:Nn \c_datatool_dot_anchored_ddmmyy_date_regex
  {
    \A \s*
    ( \ur{c_datatool_day_of_month_regex} )
     \.
    ( \ur{c_datatool_month_number_regex} )
     \. ( \d{2} )
    \s* \Z
  }
```

Common mm dd yyyy formats.

```
\regex_const:Nn \c_datatool_slash_mmddyyyy_date_regex
  {
    ( \ur{c_datatool_month_number_regex} )
     \/
    ( \ur{c_datatool_day_of_month_regex} )
     \/ ( \d+ )
  }
```

```
\regex_const:Nn \c_datatool_slash_anchored_mmddyyyy_date_regex
 {
   \A \s*
   ( \ur{c_datatool_month_number_regex} )
    \/
   ( \ur{c_datatool_day_of_month_regex} )
    \/ ( \d+ )
   \s* \Z
 }
\regex_const:Nn \c_datatool_slash_mmddyy_date_regex
 {
   ( \ur{c_datatool_month_number_regex} )
    \/
   ( \ur{c_datatool_day_of_month_regex} )
    \/ ( \d{2} )
 }
\regex_const:Nn \c_datatool_slash_anchored_mmddyy_date_regex
 {
   \A \s*
   ( \ur{c_datatool_month_number_regex} )
    \/
   ( \ur{c_datatool_day_of_month_regex} )
    \/ ( \d{2} )
   \s* \Z
 }
\regex_const:Nn \c_datatool_hyphen_mmddyyyy_date_regex
 {
   ( \ur{c_datatool_month_number_regex} )
    \-
   ( \ur{c_datatool_day_of_month_regex} )
    \- ( \d+ )
 }
\regex_const:Nn \c_datatool_hyphen_anchored_mmddyyyy_date_regex
 {
   \A \s*
   ( \ur{c_datatool_month_number_regex} )
    \-
   ( \ur{c_datatool_day_of_month_regex} )
    \- ( \d+ )
   \s* \Z
 }
\regex_const:Nn \c_datatool_hyphen_mmddyy_date_regex
 {
   ( \ur{c_datatool_month_number_regex} )
    \-
   ( \ur{c_datatool_day_of_month_regex} )
    \- ( \d{2} )
 }
```

```
\regex_const:Nn \c_datatool_hyphen_anchored_mmddyy_date_regex
 {
   \A \s*
   ( \ur{c_datatool_month_number_regex} )
    \-
   ( \ur{c_datatool_day_of_month_regex} )
    \- ( \d{2} )
   \s* \Z
 }
\regex_const:Nn \c_datatool_dot_mmddyyyy_date_regex
 {
   ( \ur{c_datatool_month_number_regex} )
    \.
   ( \ur{c_datatool_day_of_month_regex} )
    \. ( \d+ )
 }
\regex_const:Nn \c_datatool_dot_anchored_mmddyyyy_date_regex
 {
   \A \s*
   ( \ur{c_datatool_month_number_regex} )
    \.
   ( \ur{c_datatool_day_of_month_regex} )
    \. ( \d+ )
   \s* \Z
 }
\regex_const:Nn \c_datatool_dot_mmddyy_date_regex
 {
   ( \ur{c_datatool_month_number_regex} )
    \.
   ( \ur{c_datatool_day_of_month_regex} )
    \. ( \d{2} )
 }
\regex_const:Nn \c_datatool_dot_anchored_mmddyy_date_regex
 {
   \A \s*
   ( \ur{c_datatool_month_number_regex} )
    \.
   ( \ur{c_datatool_day_of_month_regex} )
    \. ( \d{2} )
   \s* \Z
 }
```

Common yyyy mm dd formats.

```
\regex_const:Nn \c_datatool_slash_yyyymmdd_date_regex
 {
   ( \d+ ) \/
   ( \ur{c_datatool_month_number_regex} )
    \/
```

```
  ( \ur{c_datatool_day_of_month_regex} )
 }
\regex_const:Nn \c_datatool_slash_anchored_yyyymmdd_date_regex
 {
  \A \s*
  ( \d+ ) \/
  ( \ur{c_datatool_month_number_regex} )
   \/
  ( \ur{c_datatool_day_of_month_regex} )
  \s* \Z
 }
\regex_const:Nn \c_datatool_slash_yymmdd_date_regex
 {
  ( \d{2} ) \/
  ( \ur{c_datatool_month_number_regex} )
   \/
  ( \ur{c_datatool_day_of_month_regex} )
 }
\regex_const:Nn \c_datatool_slash_anchored_yymmdd_date_regex
 {
  \A \s*
  ( \d{2} ) \/
  ( \ur{c_datatool_month_number_regex} )
   \/
  ( \ur{c_datatool_day_of_month_regex} )
  \s* \Z
 }
\regex_const:Nn \c_datatool_hyphen_yyyymmdd_date_regex
 {
  ( \d+ ) \-
  ( \ur{c_datatool_month_number_regex} )
   \-
  ( \ur{c_datatool_day_of_month_regex} )
 }
\regex_const:Nn \c_datatool_hyphen_anchored_yyyymmdd_date_regex
 {
  \A \s*
  ( \d+ ) \-
  ( \ur{c_datatool_month_number_regex} )
   \-
  ( \ur{c_datatool_day_of_month_regex} )
  \s* \Z
 }
\regex_const:Nn \c_datatool_hyphen_yymmdd_date_regex
 {
  ( \d{2} ) \-
  ( \ur{c_datatool_month_number_regex} )
```

```
    \-
   ( \ur{c_datatool_day_of_month_regex} )
 }
\regex_const:Nn \c_datatool_hyphen_anchored_yymmdd_date_regex
 {
   \A \s*
   ( \d{2} ) \-
   ( \ur{c_datatool_month_number_regex} )
    \-
   ( \ur{c_datatool_day_of_month_regex} )
   \s* \Z
 }
\regex_const:Nn \c_datatool_dot_yyyymmdd_date_regex
 {
   ( \d+ ) \.
   ( \ur{c_datatool_month_number_regex} )
    \.
   ( \ur{c_datatool_day_of_month_regex} )
 }
\regex_const:Nn \c_datatool_dot_anchored_yyyymmdd_date_regex
 {
   \A \s*
   ( \d+ ) \.
   ( \ur{c_datatool_month_number_regex} )
    \.
   ( \ur{c_datatool_day_of_month_regex} )
   \s* \Z
 }
\regex_const:Nn \c_datatool_dot_yymmdd_date_regex
 {
   ( \d{2} ) \.
   ( \ur{c_datatool_month_number_regex} )
    \.
   ( \ur{c_datatool_day_of_month_regex} )
 }
\regex_const:Nn \c_datatool_dot_anchored_yymmdd_date_regex
 {
   \A \s*
   ( \d{2} ) \.
   ( \ur{c_datatool_month_number_regex} )
    \.
   ( \ur{c_datatool_day_of_month_regex} )
   \s* \Z
 }
```

Time formats. If a regional format needs to support other am/pm identifiers, they'll need to provide their own regular expressions.

```
\regex_const:Nn \c_datatool_colon_hhmmss_time_regex
```

```
  {
    ( \ur{c_datatool_hour_regex} )
    \: ( \ur{c_datatool_minsec_regex} )
    (?: \: ( \ur{c_datatool_minsec_regex} ) ) ?
    (?: \s* ( am | pm ) ) ?
  }
\regex_const:Nn \c_datatool_colon_anchored_hhmmss_time_regex
  {
    \A \s*
    ( \ur{c_datatool_hour_regex} )
    \: ( \ur{c_datatool_minsec_regex} )
    (?: \: ( \ur{c_datatool_minsec_regex} ) ) ?
    (?: \s* ( am | pm ) ) ?
    \s* \Z
  }
\regex_const:Nn \c_datatool_dot_hhmmss_time_regex
  {
    ( \ur{c_datatool_hour_regex} )
    \. ( \ur{c_datatool_minsec_regex} )
    (?: \. ( \ur{c_datatool_minsec_regex} ) ) ?
    (?: \s* ( am | pm ) ) ?
  }
\regex_const:Nn \c_datatool_dot_anchored_hhmmss_time_regex
  {
    \A \s*
    ( \ur{c_datatool_hour_regex} )
    \. ( \ur{c_datatool_minsec_regex} )
    (?: \. ( \ur{c_datatool_minsec_regex} ) ) ?
    (?: \s* ( am | pm ) ) ?
    \s* \Z
  }
```

The timestamp parsing functions may either have a single regular expression that captures both the date and time, or may have two separate regular expressions for the date and time. In either case, the time zone part is optional and will be parsed with `\datatool_parse_regional_timezone:NnTF`.

Once the date, time and timestamp have been extracted, they can then be matched separately. This would be easier if named captured groups were supported, but they aren't at the time of writing, so multiple functions are needed to reference the groups by index.

Syntax: ⟨*date-regex*⟩ ⟨*time-regex*⟩ ⟨*timestamp-var*⟩ {⟨*date*⟩} {⟨*time*⟩} {⟨*timezone*⟩} {⟨*return true*⟩}{⟨*return false*⟩}

```
\cs_new:Nn \datatool_ddmmyyyy_hhmmss_tz_parse_datetime:NNNnnnTF
  {
```

First extract the date:

```
    \regex_extract_once:NnNTF #1 { #4 }
      \l_datatool_timestamp_match_seq
     {
```

Set the month first, to skip rest on failure:

```
\datatool_parse_regional_month:NeTF #3
 { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
 {
```

Successfully set month. Set the day of month:

```
 \datatool_timestamp_set_day:Ne #3
  { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the year:

```
 \datatool_timestamp_set_year:Ne #3
  { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Now extract the time.

```
 \regex_extract_once:NnNTF #2 { #5 }
   \l_datatool_timestamp_match_seq
  {
```

Set the hour:

```
    \exp_args:Ne \DTLCurrentLocaleIfpmTF
     { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
     {
      \datatool_timestamp_set_hour:Ne #3
        {
          \int_eval:n
           {
             12 +
             \seq_item:Nn \l_datatool_timestamp_match_seq { 2 }
           }
        }
     }
     {
      \datatool_timestamp_set_hour:Ne #3
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
     }
```

Set the minute:

```
    \datatool_timestamp_set_minute:Ne #3
      { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the second:

```
    \datatool_timestamp_set_second:Ne #3
      { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
  }
  { #8 }
```

Parse timezone if present:

```
\datatool_parse_regional_timezone:NnTF #3
 { #6 } { #7 } { #8 }
}
{
```

No match on month:

```
          #8
        }
    }
    { #8 }
}
\cs_generate_variant:Nn
 \datatool_ddmmyyyy_hhmmss_tz_parse_datetime:NNNnnnTF
 { NNNeeeTF }
```

As above but century needs to be added.

```
\cs_new:Nn \datatool_ddmmyy_hhmmss_tz_parse_datetime:NNNnnnTF
 {
```

First extract the date:

```
    \regex_extract_once:NnNTF #1 { #4 }
      \l_datatool_timestamp_match_seq
     {
```

Set the month first, to skip rest on failure:

```
       \datatool_parse_regional_month:NeTF #3
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
        {
```

Successfully set month. Set the day of month:

```
         \datatool_timestamp_set_day:Ne #3
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the year:

```
         \datatool_timestamp_set_year_add_century:Ne #3
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Now extract the time.

```
         \regex_extract_once:NnNTF #2 { #5 }
           \l_datatool_timestamp_match_seq
          {
```

Set the hour:

```
            \exp_args:Ne \DTLCurrentLocaleIfpmTF
             { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
             {
              \datatool_timestamp_set_hour:Ne #3
                {
                  \int_eval:n
                   {
                     12 +
                     \seq_item:Nn \l_datatool_timestamp_match_seq { 2 }
                   }
                }
             }
             {
              \datatool_timestamp_set_hour:Ne #3
```

```
                { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
            }
```
Set the minute:
```
            \datatool_timestamp_set_minute:Ne #3
              { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```
Set the second:
```
            \datatool_timestamp_set_second:Ne #3
              { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
          }
          { #8 }
```
Parse timezone if present:
```
        \datatool_parse_regional_timezone:NnTF #3
         { #6 } { #7 } { #8 }
        }
        {
```
No match on month:
```
          #8
        }
    }
    { #8 }
 }
\cs_generate_variant:Nn
 \datatool_ddmmyy_hhmmss_tz_parse_datetime:NNNnnnTF
 { NNNeeeTF }
```
Same again but the month is in the first captured group and the day is in the second.
Syntax: ⟨*date-regex*⟩ ⟨*time-regex*⟩ ⟨*timestamp-var*⟩ {⟨*date*⟩} {⟨*time*⟩} {⟨*timezone*⟩} {⟨*return true*⟩}{⟨*return false*⟩}
```
\cs_new:Nn \datatool_mmddyyyy_hhmmss_tz_parse_datetime:NNNnnnTF
 {
```
First extract the date:
```
   \regex_extract_once:NnNTF #1 { #4 }
     \l_datatool_timestamp_match_seq
    {
```
Set the month first, to skip rest on failure:
```
      \datatool_parse_regional_month:NeTF #3
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       {
```
Successfully set month. Set the day of month:
```
        \datatool_timestamp_set_day:Ne #3
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```
Set the year:
```
        \datatool_timestamp_set_year:Ne #3
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Now extract the time.

```
          \regex_extract_once:NnNTF #2 { #5 }
            \l_datatool_timestamp_match_seq
           {
```

Set the hour:

```
              \exp_args:Ne \DTLCurrentLocaleIfpmTF
               { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
               {
                \datatool_timestamp_set_hour:Ne #3
                  {
                    \int_eval:n
                     {
                       12 +
                       \seq_item:Nn \l_datatool_timestamp_match_seq { 2 }
                     }
                  }
               }
               {
                \datatool_timestamp_set_hour:Ne #3
                  { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
               }
```

Set the minute:

```
              \datatool_timestamp_set_minute:Ne #3
                { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the second:

```
              \datatool_timestamp_set_second:Ne #3
                { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
            }
            { #8 }
```

Parse timezone if present:

```
          \datatool_parse_regional_timezone:NnTF #3
           { #6 } { #7 } { #8 }
          }
          {
```

No match on month:

```
            #8
          }
      }
      { #8 }
  }
 \cs_generate_variant:Nn
  \datatool_mmddyyyy_hhmmss_tz_parse_datetime:NNNnnnTF
  { NNNeeeTF }
```

As above but century needs to be added:

```
 \cs_new:Nn \datatool_mmddyy_hhmmss_tz_parse_datetime:NNNnnnTF
  {
```

First extract the date:

```
\regex_extract_once:NnNTF #1 { #4 }
  \l_datatool_timestamp_match_seq
 {
```

Set the month first, to skip rest on failure:

```
\datatool_parse_regional_month:NeTF #3
 { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
 {
```

Successfully set month. Set the day of month:

```
\datatool_timestamp_set_day:Ne #3
 { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the year:

```
\datatool_timestamp_set_year_add_century:Ne #3
 { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Now extract the time.

```
\regex_extract_once:NnNTF #2 { #5 }
  \l_datatool_timestamp_match_seq
 {
```

Set the hour:

```
  \exp_args:Ne \DTLCurrentLocaleIfpmTF
   { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
   {
    \datatool_timestamp_set_hour:Ne #3
      {
        \int_eval:n
         {
           12 +
           \seq_item:Nn \l_datatool_timestamp_match_seq { 2 }
         }
      }
   }
   {
    \datatool_timestamp_set_hour:Ne #3
      { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
   }
```

Set the minute:

```
  \datatool_timestamp_set_minute:Ne #3
    { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the second:

```
  \datatool_timestamp_set_second:Ne #3
    { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
 }
 { #8 }
```

Parse timezone if present:

```
        \datatool_parse_regional_timezone:NnTF #3
         { #6 } { #7 } { #8 }
        }
        {
```

No match on month:

```
          #8
        }
    }
    { #8 }
}
\cs_generate_variant:Nn
 \datatool_mmddyy_hhmmss_tz_parse_datetime:NNNnnnTF
 { NNNeeeTF }
```

Same again but the year is in the first captured group, the month is in the second captured group and the day is in the third.

Syntax: ⟨*date-regex*⟩ ⟨*time-regex*⟩ ⟨*timestamp-var*⟩ {⟨*date*⟩} {⟨*time*⟩} {⟨*timezone*⟩} {⟨*return true*⟩}{⟨*return false*⟩}

```
\cs_new:Nn \datatool_yyyymmdd_hhmmss_tz_parse_datetime:NNNnnnTF
 {
```

First extract the date:

```
    \regex_extract_once:NnTF #1 { #4 }
      \l_datatool_timestamp_match_seq
     {
```

Set the month first, to skip rest on failure:

```
       \datatool_parse_regional_month:NeTF #3
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
        {
```

Successfully set month. Set the day of month:

```
         \datatool_timestamp_set_day:Ne #3
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Set the year:

```
         \datatool_timestamp_set_year:Ne #3
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Now extract the time.

```
         \regex_extract_once:NnTF #2 { #5 }
           \l_datatool_timestamp_match_seq
          {
```

Set the hour:

```
            \exp_args:Ne \DTLCurrentLocaleIfpmTF
             { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
             {
              \datatool_timestamp_set_hour:Ne #3
                {
```

```
                \int_eval:n
                 {
                   12 +
                   \seq_item:Nn \l_datatool_timestamp_match_seq { 2 }
                 }
               }
             }
             {
              \datatool_timestamp_set_hour:Ne #3
                { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
             }
```

Set the minute:

```
            \datatool_timestamp_set_minute:Ne #3
              { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the second:

```
            \datatool_timestamp_set_second:Ne #3
              { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
          }
          { #8 }
```

Parse timezone if present:

```
        \datatool_parse_regional_timezone:NnTF #3
         { #6 } { #7 } { #8 }
        }
        {
```

No match on month:

```
          #8
        }
     }
     { #8 }
  }
 \cs_generate_variant:Nn
  \datatool_yyyymmdd_hhmmss_tz_parse_datetime:NNNnnnTF
  { NNNeeeTF }
```

As above, but the century needs to be added. (Unlikely with year first format, but provided for completeness.)

```
 \cs_new:Nn \datatool_yymmdd_hhmmss_tz_parse_datetime:NNNnnnTF
  {
```

First extract the date:

```
    \regex_extract_once:NnNTF #1 { #4 }
      \l_datatool_timestamp_match_seq
     {
```

Set the month first, to skip rest on failure:

```
       \datatool_parse_regional_month:NeTF #3
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
        {
```

Successfully set month. Set the day of month:

```
          \datatool_timestamp_set_day:Ne #3
           { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Set the year:

```
          \datatool_timestamp_set_year_add_century:Ne #3
           { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Now extract the time.

```
          \regex_extract_once:NnNTF #2 { #5 }
            \l_datatool_timestamp_match_seq
           {
```

Set the hour:

```
              \exp_args:Ne \DTLCurrentLocaleIfpmTF
               { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
               {
                \datatool_timestamp_set_hour:Ne #3
                  {
                    \int_eval:n
                     {
                       12 +
                       \seq_item:Nn \l_datatool_timestamp_match_seq { 2 }
                     }
                  }
               }
               {
                \datatool_timestamp_set_hour:Ne #3
                  { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
               }
```

Set the minute:

```
              \datatool_timestamp_set_minute:Ne #3
                { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the second:

```
              \datatool_timestamp_set_second:Ne #3
                { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
            }
            { #8 }
```

Parse timezone if present:

```
          \datatool_parse_regional_timezone:NnTF #3
           { #6 } { #7 } { #8 }
          }
          {
```

No match on month:

```
            #8
          }
      }
      { #8 }
```

```
 }
\cs_generate_variant:Nn
 \datatool_yymmdd_hhmmss_tz_parse_datetime:NNNnnnTF
 { NNNeeeTF }
```

The timestamp separator used when date and time patterns provided separately:

```
\regex_new:N \l_datatool_regional_timestamp_sep_regex
\regex_set:Nn \l_datatool_regional_timestamp_sep_regex { \,? \s+ }
```

Syntax: `{`⟨*date-regex-varname*⟩`}` `{`⟨*time-regex-varname*⟩`}` ⟨*timestamp seq-var*⟩ `{`⟨*text*⟩`}` `{`⟨*return true*⟩`}` `{`⟨*return false*⟩`}`

The date regular expression should have the following groups (in order): day, month, year. The time regular expression should have the following groups: hour, minute, second (may be empty), am/pm (may be empty). The am/pm part can be anything that `\DTLCurrentLocaleIfpmTF` can detect as afternoon (that is, add 12 to the hour). This assumes that the date comes first and is separated from the time by `\l_datatool_regional_timestamp_sep_regex`.

```
\cs_new:Nn \datatool_ddmmyyyy_hhmmss_tz_parse_timestamp:NNNnTF
 {
   \tl_set_eq:NN \l__datatool_tmpa_regex #1
   \tl_set_eq:NN \l__datatool_tmpb_regex #2
   \regex_extract_once:nnNTF
    {
      \A \s*
      ( \ur{l__datatool_tmpa_regex} )
      \ur{l_datatool_regional_timestamp_sep_regex}
      ( \ur{l__datatool_tmpb_regex} )
      \s*
      ( \ur{c_datatool_timezone_id_regex} ) ?
      \s*
      \Z
    }
    { #4 }
     \l_datatool_timestamp_match_seq
    {
      \datatool_timestamp_zero:N #3
      \datatool_ddmmyyyy_hhmmss_tz_parse_datetime:NNNeeeTF
       #1 #2 #3
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
       { #5 } { #6 }
    }
    { #6 }
 }
\cs_generate_variant:Nn
 \datatool_ddmmyyyy_hhmmss_tz_parse_timestamp:NNNnTF
 { ccNnTF }
```

As above, but century needs to be added.

```
\cs_new:Nn \datatool_ddmmyy_hhmmss_tz_parse_timestamp:NNNnTF
 {
   \tl_set_eq:NN \l__datatool_tmpa_regex #1
   \tl_set_eq:NN \l__datatool_tmpb_regex #2
   \regex_extract_once:nnNTF
    {
      \A \s*
      ( \ur{l__datatool_tmpa_regex} )
      \ur{l_datatool_regional_timestamp_sep_regex}
      ( \ur{l__datatool_tmpb_regex} )
      \s*
      ( \ur{c_datatool_timezone_id_regex} ) ?
      \s*
      \Z
    }
    { #4 }
     \l_datatool_timestamp_match_seq
    {
      \datatool_timestamp_zero:N #3
      \datatool_ddmmyy_hhmmss_tz_parse_datetime:NNNeeeTF
       #1 #2 #3
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
       { #5 } { #6 }
    }
    { #6 }
 }
\cs_generate_variant:Nn
 \datatool_ddmmyy_hhmmss_tz_parse_timestamp:NNNnTF
 { ccNnTF }
```

Same again, but month in first captured group.

```
\cs_new:Nn \datatool_mmddyyyy_hhmmss_tz_parse_timestamp:NNNnTF
 {
   \tl_set_eq:NN \l__datatool_tmpa_regex #1
   \tl_set_eq:NN \l__datatool_tmpb_regex #2
   \regex_extract_once:nnNTF
    {
      \A \s*
      ( \ur{l__datatool_tmpa_regex} )
      \ur{l_datatool_regional_timestamp_sep_regex}
      ( \ur{l__datatool_tmpb_regex} )
      \s*
      ( \ur{c_datatool_timezone_id_regex} ) ?
      \s*
      \Z
    }
    { #4 }
     \l_datatool_timestamp_match_seq
```

```
    {
      \datatool_timestamp_zero:N #3
      \datatool_mmddyyyy_hhmmss_tz_parse_datetime:NNNeeeTF
       #1 #2 #3
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
       { #5 } { #6 }
    }
    { #6 }
 }
\cs_generate_variant:Nn
 \datatool_mmddyyyy_hhmmss_tz_parse_timestamp:NNNnTF
 { ccNnTF }
```

As above, but century needs to be added.

```
\cs_new:Nn \datatool_mmddyy_hhmmss_tz_parse_timestamp:NNNnTF
 {
   \tl_set_eq:NN \l__datatool_tmpa_regex #1
   \tl_set_eq:NN \l__datatool_tmpb_regex #2
   \regex_extract_once:nnNTF
    {
      \A \s*
      ( \ur{l__datatool_tmpa_regex} )
      \ur{l_datatool_regional_timestamp_sep_regex}
      ( \ur{l__datatool_tmpb_regex} )
      \s*
      ( \ur{c_datatool_timezone_id_regex} ) ?
      \s*
      \Z
    }
    { #4 }
     \l_datatool_timestamp_match_seq
    {
      \datatool_timestamp_zero:N #3
      \datatool_mmddyy_hhmmss_tz_parse_datetime:NNNeeeTF
       #1 #2 #3
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
       { #5 } { #6 }
    }
    { #6 }
 }
\cs_generate_variant:Nn
 \datatool_mmddyy_hhmmss_tz_parse_timestamp:NNNnTF
 { ccNnTF }
```

Same again, but year in first captured group.

```
\cs_new:Nn \datatool_yyyymmdd_hhmmss_tz_parse_timestamp:NNNnTF
 {
```

```
    \tl_set_eq:NN \l__datatool_tmpa_regex #1
    \tl_set_eq:NN \l__datatool_tmpb_regex #2
    \regex_extract_once:nnNTF
     {
       \A \s*
       ( \ur{l__datatool_tmpa_regex} )
       \ur{l_datatool_regional_timestamp_sep_regex}
       ( \ur{l__datatool_tmpb_regex} )
       \s*
       ( \ur{c_datatool_timezone_id_regex} ) ?
       \s*
       \Z
     }
     { #4 }
      \l_datatool_timestamp_match_seq
     {
       \datatool_timestamp_zero:N #3
       \datatool_yyyymmdd_hhmmss_tz_parse_datetime:NNNeeeTF
        #1 #2 #3
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
        { #5 } { #6 }
     }
     { #6 }
  }
 \cs_generate_variant:Nn
  \datatool_yyyymmdd_hhmmss_tz_parse_timestamp:NNNnTF
  { ccNnTF }
```

As above, but century needs to be added.

```
 \cs_new:Nn \datatool_yymmdd_hhmmss_tz_parse_timestamp:NNNnTF
  {
    \tl_set_eq:NN \l__datatool_tmpa_regex #1
    \tl_set_eq:NN \l__datatool_tmpb_regex #2
    \regex_extract_once:nnNTF
     {
       \A \s*
       ( \ur{l__datatool_tmpa_regex} )
       \ur{l_datatool_regional_timestamp_sep_regex}
       ( \ur{l__datatool_tmpb_regex} )
       \s*
       ( \ur{c_datatool_timezone_id_regex} ) ?
       \s*
       \Z
     }
     { #4 }
      \l_datatool_timestamp_match_seq
     {
       \datatool_timestamp_zero:N #3
```

```
      \datatool_yymmdd_hhmmss_tz_parse_datetime:NNNeeeTF
       #1 #2 #3
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
       { #5 } { #6 }
    }
    { #6 }
 }
\cs_generate_variant:Nn
 \datatool_yymmdd_hhmmss_tz_parse_timestamp:NNNnTF
 { ccNnTF }
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

The regular expression should have the following groups (in order): day, month, year, hour, minute, second (may be empty), am/pm (may be empty), time zone (may be empty). The am/pm part can be anything that `\DTLCurrentLocaleIfpmTF` can detect as afternoon (that is, add 12 to the hour). The constants provided with this base package only match "am" and "pm". Localisation files may provide their own regular expressions with appropriate patterns.

```
\cs_new:Nn \datatool_ddmmyyyy_hhmmss_tz_parse_timestamp:NNnTF
 {
   \regex_extract_once:NnNTF #1 { #3 }
     \l_datatool_timestamp_match_seq
    {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       {
```

Successfully set month. Set the day of month:

```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the year:

```
        \datatool_timestamp_set_year:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Set the hour:

```
        \exp_args:Ne \DTLCurrentLocaleIfpmTF
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 8 } }
         {
          \datatool_timestamp_set_hour:Ne #2
            {
              \int_eval:n
               {
                 12 +
                 \seq_item:Nn \l_datatool_timestamp_match_seq { 5 }
```

```
            }
          }
        }
        {
         \datatool_timestamp_set_hour:Ne #2
           { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
        }
```

Set the minute:

```
        \datatool_timestamp_set_minute:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 6 } }
```

Set the second:

```
        \datatool_timestamp_set_second:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 7 } }
```

Parse timezone if present:

```
       \datatool_parse_regional_timezone:NeTF #2
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 9 } }
        { #4 } { #5 }
      }
      {
```

No match on month:

```
        #5
      }
    }
    { #5 } % No match
 }
```

As above but century missing. Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_ddmmyy_hhmmss_tz_parse_timestamp:NNnTF
 {
   \regex_extract_once:NnNTF #1 { #3 }
     \l_datatool_timestamp_match_seq
    {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       {
```

Successfully set month. Set the day of month:

```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the year:

```
        \datatool_timestamp_set_year_add_century:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Set the hour:

```
\exp_args:Ne \DTLCurrentLocaleIfpmTF
 { \seq_item:Nn \l_datatool_timestamp_match_seq { 8 } }
 {
  \datatool_timestamp_set_hour:Ne #2
    {
      \int_eval:n
       {
         12 +
         \seq_item:Nn \l_datatool_timestamp_match_seq { 5 }
       }
    }
 }
 {
   \datatool_timestamp_set_hour:Ne #2
     { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
 }
```

Set the minute:

```
\datatool_timestamp_set_minute:Ne #2
  { \seq_item:Nn \l_datatool_timestamp_match_seq { 6 } }
```

Set the second:

```
\datatool_timestamp_set_second:Ne #2
  { \seq_item:Nn \l_datatool_timestamp_match_seq { 7 } }
```

Parse timezone if present:

```
\datatool_parse_regional_timezone:NeTF #2
 { \seq_item:Nn \l_datatool_timestamp_match_seq { 9 } }
 { #4 } { #5 }
}
{
```

No match on month:

```
    #5
  }
}
{ #5 } % No match
}
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

The regular expression should have the following groups (in order): month, day, year, hour, minute, second (may be empty), am/pm (may be empty), time zone (may be empty).

```
\cs_new:Nn \datatool_mmddyyyy_hhmmss_tz_parse_timestamp:NNnTF
{
  \regex_extract_once:NnNTF #1 { #3 }
    \l_datatool_timestamp_match_seq
  {
```

Initialise:

```
\datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
        \datatool_parse_regional_month:NeTF #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
         {
```

Successfully set month. Set the day of month:

```
          \datatool_timestamp_set_day:Ne #2
           { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the year:

```
          \datatool_timestamp_set_year:Ne #2
           { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Set the hour:

```
          \exp_args:Ne \DTLCurrentLocaleIfpmTF
           { \seq_item:Nn \l_datatool_timestamp_match_seq { 8 } }
           {
            \datatool_timestamp_set_hour:Ne #2
              {
                \int_eval:n
                 {
                   12 +
                   \seq_item:Nn \l_datatool_timestamp_match_seq { 5 }
                 }
              }
           }
           {
             \datatool_timestamp_set_hour:Ne #2
               { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
           }
```

Set the minute:

```
          \datatool_timestamp_set_minute:Ne #2
            { \seq_item:Nn \l_datatool_timestamp_match_seq { 6 } }
```

Set the second:

```
          \datatool_timestamp_set_second:Ne #2
            { \seq_item:Nn \l_datatool_timestamp_match_seq { 7 } }
```

Parse timezone if present:

```
         \datatool_parse_regional_timezone:NeTF #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 9 } }
          { #4 } { #5 }
        }
        {
```

No match on month:

```
          #5
        }
      }
      { #5 } % No match
  }
```

As above but missing century. Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_mmddyy_hhmmss_tz_parse_timestamp:NNnTF
 {
   \regex_extract_once:NnNTF #1 { #3 }
     \l_datatool_timestamp_match_seq
    {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       {
```

Successfully set month. Set the day of month:

```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the year:

```
        \datatool_timestamp_set_year_add_century:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Set the hour:

```
        \exp_args:Ne \DTLCurrentLocaleIfpmTF
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 8 } }
         {
          \datatool_timestamp_set_hour:Ne #2
            {
              \int_eval:n
               {
                 12 +
                 \seq_item:Nn \l_datatool_timestamp_match_seq { 5 }
               }
            }
         }
         {
           \datatool_timestamp_set_hour:Ne #2
             { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
         }
```

Set the minute:

```
        \datatool_timestamp_set_minute:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 6 } }
```

Set the second:

```
        \datatool_timestamp_set_second:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 7 } }
```

Parse timezone if present:

```
       \datatool_parse_regional_timezone:NeTF #2
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 9 } }
```

```
        { #4 } { #5 }
      }
      {
```

No match on month:

```
        #5
      }
    }
    { #5 } % No match
  }
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

The regular expression should have the following groups (in order): year, month, day, hour, minute, second (may be empty), am/pm (may be empty), time zone (may be empty).

```
\cs_new:Nn \datatool_yyyymmdd_hhmmss_tz_parse_timestamp:NNnTF
 {
   \regex_extract_once:NnNTF #1 { #3 }
    \l_datatool_timestamp_match_seq
   {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       {
```

Successfully set month. Set the year:

```
        \datatool_timestamp_set_year:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the day of month:

```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Set the hour:

```
        \exp_args:Ne \DTLCurrentLocaleIfpmTF
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 8 } }
         {
          \datatool_timestamp_set_hour:Ne #2
            {
              \int_eval:n
               {
                 12 +
                 \seq_item:Nn \l_datatool_timestamp_match_seq { 5 }
               }
            }
         }
         {
           \datatool_timestamp_set_hour:Ne #2
```

```
            { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
          }
```
Set the minute:
```
        \datatool_timestamp_set_minute:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 6 } }
```
Set the second:
```
        \datatool_timestamp_set_second:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 7 } }
```
Parse timezone if present:
```
       \datatool_parse_regional_timezone:NeTF #2
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 9 } }
        { #4 } { #5 }
      }
      {
```
No match on month:
```
        #5
      }
    }
    { #5 } % No match
  }
```
As above but missing century. Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}
```
\cs_new:Nn \datatool_yymmdd_hhmmss_tz_parse_timestamp:NNnTF
 {
   \regex_extract_once:NnNTF #1 { #3 }
     \l_datatool_timestamp_match_seq
    {
```
Initialise:
```
      \datatool_timestamp_zero:N #2
```
Set the month first, to skip rest on failure:
```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       {
```
Successfully set month. Set the year:
```
        \datatool_timestamp_set_year_add_century:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```
Set the day of month:
```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```
Set the hour:
```
        \exp_args:Ne \DTLCurrentLocaleIfpmTF
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 8 } }
         {
```

```
          \datatool_timestamp_set_hour:Ne #2
            {
              \int_eval:n
               {
                 12 +
                 \seq_item:Nn \l_datatool_timestamp_match_seq { 5 }
               }
            }
          }
          {
            \datatool_timestamp_set_hour:Ne #2
              { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
          }
```

Set the minute:

```
        \datatool_timestamp_set_minute:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 6 } }
```

Set the second:

```
        \datatool_timestamp_set_second:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 7 } }
```

Parse timezone if present:

```
       \datatool_parse_regional_timezone:NeTF #2
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 9 } }
        { #4 } { #5 }
      }
      {
```

No match on month:

```
        #5
      }
    }
    { #5 } % No match
  }
```

Syntax: ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_parse_regional_timezone:NnTF
 {
   \tl_if_empty:nTF { #2 }
    {
```

Missing time zone valid so return true.

```
      #3
    }
    {
```

Check for known timezone identifier. First try regionless mapping.

```
      \prop_get:NnNF \l_datatool_timezone_map_prop
        { #2 }
       \l_datatool_timezone_map_value_tl
       {
```

No regionless mapping so now try locale mapping.

```
        \DTLCurrentLocaleGetTimeZoneMap { #2 }
        \quark_if_no_value:NT
          \l_datatool_timezone_map_value_tl
         {
```

No mapping so may be numeric.

```
            \tl_set:Nn \l_datatool_timezone_map_value_tl { #2 }
          }
        }
       \exp_args:NNV \regex_extract_once:NnNTF
          \c_datatool_timezone_regex
          \l_datatool_timezone_map_value_tl
          \l_datatool_regex_match_seq
         {
          \datatool_timestamp_set_tzhour:Ne #1
            { \seq_item:Nn \l_datatool_regex_match_seq { 2 } }
          \tl_if_empty:eF
            { \seq_item:Nn \l_datatool_regex_match_seq { 3 } }
           {
            \datatool_timestamp_set_tzminute:Ne #1
              { \seq_item:Nn \l_datatool_regex_match_seq { 3 } }
           }
          #3
         }
         { #4 }
    }
 }
\cs_generate_variant:Nn \datatool_parse_regional_timezone:NnTF
  { NeTF }
```

Syntax: ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_parse_regional_month:NnTF
 {
```

Check for month name mapping.

```
    \DTLCurrentLocaleGetMonthNameMap { #2 }
    \quark_if_no_value:NT
      \l_datatool_monthname_map_value_tl
     {
```

No mapping so check if numeric.

```
       \tl_set:Nn \l_datatool_monthname_map_value_tl { #2 }
     }
    \exp_args:NNV \regex_extract_once:NnNTF
       \c_datatool_month_number_regex
       \l_datatool_monthname_map_value_tl
       \l_datatool_regex_match_seq
      {
        \datatool_timestamp_set_month:Ne #1
          { \seq_item:Nn \l_datatool_regex_match_seq { 2 } }
```

```
        #3
      }
      { #4 }
  }
 \cs_generate_variant:Nn \datatool_parse_regional_month:NnTF
  { NeTF }
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

The regular expression should have the following groups (in order): day, month, year.

```
 \cs_new:Nn \datatool_ddmmyyyy_parse_date:NNnTF
  {
    \regex_extract_once:NnNTF #1 { #3 }
      \l_datatool_timestamp_match_seq
     {
```

Initialise:

```
       \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
       \datatool_parse_regional_month:NeTF #2
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
        {
```

Successfully set month. Set the day of month:

```
         \datatool_timestamp_set_day:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the year:

```
         \datatool_timestamp_set_year:Ne #2
          { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Success

```
         #4
       }
       {
```

No match on month:

```
         #5
       }
     }
     { #5 } % No match
  }
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

The regular expression should have the following groups (in order): month, day, year.

```
 \cs_new:Nn \datatool_mmddyyyy_parse_date:NNnTF
  {
    \regex_extract_once:NnNTF #1 { #3 }
      \l_datatool_timestamp_match_seq
     {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       {
```

Successfully set month. Set the day of month:

```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the year:

```
        \datatool_timestamp_set_year:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Success

```
        #4
      }
      {
```

No match on month:

```
        #5
      }
    }
    { #5 } % No match
  }
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

The regular expression should have the following groups (in order): year, month, day.

```
 \cs_new:Nn \datatool_yyyymmdd_parse_date:NNnTF
  {
    \regex_extract_once:NnNTF #1 { #3 }
     \l_datatool_timestamp_match_seq
    {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       {
```

Successfully set month. Set the year:

```
        \datatool_timestamp_set_year:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the day of month:

```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Success

```
          #4
        }
        {
```

No match on month:

```
          #5
        }
      }
      { #5 } % No match
  }
```

As above but century missing: Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
 \cs_new:Nn \datatool_ddmmyy_parse_date:NNnTF
  {
    \regex_extract_once:NnNTF #1 { #3 }
      \l_datatool_timestamp_match_seq
     {
```

Initialise:

```
       \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
       \datatool_parse_regional_month:NeTF #2
        { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
        {
```

Successfully set month. Set the day of month:

```
          \datatool_timestamp_set_day:Ne #2
           { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the year:

```
          \datatool_timestamp_set_year_add_century:Ne #2
           { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Success

```
          #4
        }
        {
```

No match on month:

```
          #5
        }
      }
      { #5 } % No match
  }
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
 \cs_new:Nn \datatool_mmddyy_parse_date:NNnTF
  {
    \regex_extract_once:NnNTF #1 { #3 }
      \l_datatool_timestamp_match_seq
     {
```

```
Initialise:
      \datatool_timestamp_zero:N #2
Set the month first, to skip rest on failure:
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       {
Successfully set month. Set the day of month:
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
Set the year:
        \datatool_timestamp_set_year_add_century:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
Success
        #4
      }
      {
No match on month:
        #5
      }
    }
    { #5 } % No match
}
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_yymmdd_parse_date:NNnTF
{
  \regex_extract_once:NnNTF #1 { #3 }
    \l_datatool_timestamp_match_seq
   {
Initialise:
      \datatool_timestamp_zero:N #2
Set the month first, to skip rest on failure:
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       {
Successfully set month. Set the year:
        \datatool_timestamp_set_year_add_century:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
Set the day of month:
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
Success
        #4
      }
      {
```

No match on month:

```
        #5
      }
    }
    { #5 } % No match
}
```

As above, but year missing. Assume current year: Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_ddmm_parse_date:NNnTF
{
  \regex_extract_once:NnNTF #1 { #3 }
    \l_datatool_timestamp_match_seq
  {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
       {
```

Successfully set month. Set the day of month:

```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
```

Set the year:

```
        \datatool_timestamp_set_year:Nn #2 { }
```

Success

```
        #4
      }
      {
```

No match on month:

```
        #5
      }
    }
    { #5 } % No match
}
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_mmdd_parse_date:NNnTF
{
  \regex_extract_once:NnNTF #1 { #3 }
    \l_datatool_timestamp_match_seq
  {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the month first, to skip rest on failure:

```
      \datatool_parse_regional_month:NeTF #2
       { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
       {
```

Successfully set month. Set the day of month:

```
        \datatool_timestamp_set_day:Ne #2
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the year:

```
        \datatool_timestamp_set_year:Nn #2 { }
```

Success

```
        #4
      }
      {
```

No match on month:

```
        #5
      }
    }
    { #5 } % No match
}
```

Syntax: ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_hhmmss_parse_time:NnTF
{
  \datatool_hhmmss_parse_time:NNnTF
    \c_datatool_colon_anchored_hhmmss_time_regex #1 { #2 } { #3 } { #4 }
}
```

Syntax: ⟨*regex-var*⟩ ⟨*timestamp seq-var*⟩ {⟨*text*⟩} {⟨*return true*⟩} {⟨*return false*⟩}

```
\cs_new:Nn \datatool_hhmmss_parse_time:NNnTF
{
  \regex_extract_once:NnNTF
    #1 { #3 }
    \l_datatool_timestamp_match_seq
   {
```

Initialise:

```
      \datatool_timestamp_zero:N #2
```

Set the hour:

```
        \exp_args:Ne \DTLCurrentLocaleIfpmTF
         { \seq_item:Nn \l_datatool_timestamp_match_seq { 5 } }
         {
          \datatool_timestamp_set_hour:Ne #2
            {
              \int_eval:n
               {
                 12 +
                 \seq_item:Nn \l_datatool_timestamp_match_seq { 2 }
               }
```

```
            }
          }
          {
            \datatool_timestamp_set_hour:Ne #2
              { \seq_item:Nn \l_datatool_timestamp_match_seq { 2 } }
          }
```

Set the minute:

```
\datatool_timestamp_set_minute:Ne #2
  { \seq_item:Nn \l_datatool_timestamp_match_seq { 3 } }
```

Set the second:

```
\datatool_timestamp_set_second:Ne #2
  { \seq_item:Nn \l_datatool_timestamp_match_seq { 4 } }
```

Success

```
        #4
      }
      { #5 } % No match
  }
 \cs_generate_variant:Nn \datatool_hhmmss_parse_time:NNnTF
  { cNnTF }
```

2.4.3 Currencies

`\@dtl@currencies` Renamed `\@dtl@currencies` since the internal structure is changed. This will trigger an unknown control sequence error if it's being used by anything. The new name is `\l__datatool_known_currencies_seq` but it needs to be defined before `\datatool_set_currencysign:nn` is used.

An internal list that stores all known currencies.

```
\seq_put_right:Nn \l__datatool_known_currencies_seq { \$ }
\seq_put_right:Nn \l__datatool_known_currencies_seq {\pounds }
\seq_put_right:Nn \l__datatool_known_currencies_seq {\textdollar }
\seq_put_right:Nn \l__datatool_known_currencies_seq {\textstirling }
\seq_put_right:Nn \l__datatool_known_currencies_seq {\texteuro }
\seq_put_right:Nn \l__datatool_known_currencies_seq {\textyen }
\seq_put_right:Nn \l__datatool_known_currencies_seq {\textwon }
\seq_put_right:Nn \l__datatool_known_currencies_seq {\textcurrency }
```

`\@dtl@currency` `\@dtl@currency` is set by `\DTLconverttodecimal` and `\@dtl@checknumerical`. It is used by `\DTLcurrency`. Set to `\$` by default.

```
\newcommand*{\@dtl@currency}{\$}
```

`\DTLCurrencySymbol`

```
\newcommand*{\DTLCurrencySymbol}{\@dtl@currency}
```

`\DTLCurrencyCode`

```
\newcommand*{\DTLCurrencyCode}{XXX}
```

\DTLsetdefaultcurrency `\DTLsetdefaultcurrency{`⟨*symbol*⟩`}` sets the default currency.

```
\NewDocumentCommand \DTLsetdefaultcurrency { m }
{
```

Detokenize in test in case argument isn't a currency code to allow for backward-compatibility.

```
  \tl_if_exist:cTF { dtl@curr@ \tl_to_str:n { #1 } @sym }
  {
```

Defined currency code.

```
    \tl_set:Ne \@dtl@currency { \exp_not:c { DTLcurr #1 } }
    \tl_set:Ne \DTLCurrencyCode { #1 }
    \tl_set:Ne \DTLfmtcurrency
      { \exp_not:c { dtl@curr@ #1 @fmt } }
  }
  {
```

Unknown currency.

```
    \tl_set:Nn \DTLCurrencyCode { XXX }
    \tl_set:Nn \@dtl@currency { #1 }
  }
}
```

For use with the region files.

```
\cs_new:Nn \datatool_register_regional_currency_code:nn
 {
   \seq_put_right:Nn \l_datatool_regional_currencies_seq { #2 }
   \prop_put:Nnn \l_datatool_regional_currencies_prop { #1 } { #2 }
 }
\cs_new:Nn \datatool_currency_symbol_region_prefix:n
 {
   \DTLcurrCodeOrSymOrChar
    { }
    { \datatoolcurrencysymbolprefixfmt { #1 } }
    { \datatoolcurrencysymbolprefixfmt { #1 } }
 }
```

\DTLcurrSym

```
\newcommand{\DTLcurrSym}[1]{\csuse{dtl@curr@ #1 @sym}}
```

\DTLcurrChar

```
\newcommand{\DTLcurrChar}[1]{\csuse{dtl@curr@ #1 @tl}}
```

\DTLcurrStr

```
\newcommand{\DTLcurrStr}[1]{\csuse{dtl@curr@ #1 @str}}
```

\DTLdefcurrency Define currency.

`\DTLdefcurrency[`⟨*fmt*⟩`]{`⟨*iso-code*⟩`}{`⟨*symbol*⟩`}{`⟨*string*⟩`}`

NB `\DTLdefcurrency` doesn't pick up the fourth ⟨*string*⟩ argument. Category code of $ needs to be changed first.

```
\NewDocumentCommand{\DTLdefcurrency} { O{\dtlcurrdefaultfmt} m m }
{
  \group_begin:
  \char_set_catcode_other:N \$
  \__datatool_def_currency:nnnn { #1 } { #2 } { #3 }
}
\cs_new:Nn \__datatool_def_currency:nnnn
{
  \group_end:
  \datatool_def_currency:nnne { #1 } { #2 } { #3 } { #4 }
}
```

Lower-level command. Three arguments, use default format:

```
\cs_new:Nn \datatool_def_currency:nnn
 {
   \datatool_def_currency:nnnn
     { \dtlcurrdefaultfmt } { #1 } { #2 } { #3 }
 }
\cs_generate_variant:Nn \datatool_def_currency:nnn
  { nnV , nne , nVV }
```

Four arguments, supply formatting command in first argument:

```
\cs_new:Nn \datatool_def_currency:nnnn
 {
  \seq_put_right:Nn \l_datatool_currencies_seq { #2 }

  \csedef { dtl@curr@ #2 @str } { \detokenize { #4 } }
  \csdef { dtl@curr@ #2 @tl } { #4 }
  \csdef { dtl@curr@ #2 @sym } { #3 }
  \csedef { DTLcurr #2 }
   {
    \exp_not:N \dtltexorsort
     {
       \exp_not:N \DTLcurrCodeOrSymOrChar
        { #2 }
        { \exp_not:N \DTLcurrSym { #2 } }
        { \exp_not:N \DTLcurrChar { #2 } }
     }
     {
       \exp_not:N \DTLcurrStr { #2 }
     }
   }
  \DTLnewcurrencysymbol { #3 }
  \DTLnewcurrencysymbol { #4 }
  \exp_args:Nc \DTLnewcurrencysymbol { DTLcurr #2 }
  \csdef { dtl@curr@ #2 @fmt } { #1 }
 }
\cs_generate_variant:Nn \datatool_def_currency:nnnn
```

```
{ nnnV , nnne , nnVV }
```

Find first declared regional currency code that matches the given symbol. Returns empty if no match. First tests if a match on current currency.

```
\cs_new:Nn \datatool_get_currency_code:Nn
 {
   \__datatool_if_current_currency:nTF { #2 }
    {
      \tl_set_eq:NN #1 \DTLCurrencyCode
    }
    {
      \tl_clear:N #1
      \seq_map_inline:Nn \l_datatool_regional_currencies_seq
       {
         \tl_if_eq:nnTF { ##1 } { #2 }
          {
            \tl_set:Nn #1 { ##1 }
            \seq_map_break:
          }
          {
            \tl_if_eq:nnTF { \DTLcurr { ##1 } } { #2 }
             {
               \tl_set:Nn #1 { ##1 }
               \seq_map_break:
             }
             {
               \exp_args:Ne \tl_if_eq:nnTF { \exp_not:c { DTLcurr ##1 } } { #2 }
                {
                  \tl_set:Nn #1 { ##1 }
                  \seq_map_break:
                }
                {
                  \tl_if_eq:cnTF { dtl@curr@ ##1 @sym } { #2 }
                   {
                     \tl_set:Nn #1 { ##1 }
                     \seq_map_break:
                   }
                   {
                     \str_if_eq:vnT { dtl@curr@ ##1 @str } { #2 }
                      {
                        \tl_set:Nn #1 { ##1 }
                        \seq_map_break:
                      }
                   }
                }
             }
          }
       }
    }
 }
```

```
\cs_generate_variant:Nn  \datatool_get_currency_code:Nn { NV }
```

Test if the supplied token list matches the current symbol or string.

```
\prg_new_conditional:Npnn \__datatool_if_current_currency:n #1
   { T, F, TF }
 {
  \tl_if_eq:NnTF \@dtl@currency { #1 }
   {
     \prg_return_true:
   }
   {
     \tl_if_exist:cTF
      { DTLcurr \DTLCurrencyCode }
      {
        \tl_if_eq:cnTF
          { DTLcurr \DTLCurrencyCode }
          { #1 }
         {
           \prg_return_true:
         }
         {
           \exp_args:Nv \tl_if_eq:nnTF
             { dtl@curr@ \DTLCurrencyCode @sym } { #1 }
           {
             \prg_return_true:
           }
           {
             \str_if_eq:vnTF
               { dtl@curr@ \DTLCurrencyCode @str }
               { #1 }
             {
               \prg_return_true:
             }
             {
               \exp_args:Ne \tl_if_eq:nnTF
                { \exp_not:N \DTLcurr { \DTLCurrencyCode } }
                { #1 }
                {
                  \prg_return_true:
                }
                {
                  \prg_return_false:
                }
             }
           }
         }
      }
      {
       \prg_return_false:
      }
```

```
    }
  }
```

Allow the command used for the currency to be changed. Note that this will add the new symbol to the list of known currency symbols but won't remove the old one.

```
\cs_new:Nn \datatool_set_currency_symbol:nn
 {
   \tl_if_exist:cTF { dtl@curr@ #1 @sym }
    {
      \csdef { dtl@curr@ #1 @sym } { #2 }
      \DTLnewcurrencysymbol { #2 }
    }
    {
      \PackageError { datatool-base }
       {
         Can't ~ set ~ currency ~ symbol ~ to ~ \tl_to_str:n { #2 } ~ : ~
         Currency ~ ` #1 ' ~ not ~ defined
       }
       {
         Check ~ that ~ you ~ have ~ spelt ~ the ~
         currency ~ code ~ correctly
       }
    }
 }
\cs_generate_variant:Nn \datatool_set_currency_symbol:nn
 { nV , ne }
```

`\DTLcurrency` Format currency using current symbol.

```
\newcommand{\DTLcurrency}[1]{\DTLfmtcurrency{\@dtl@currency}{#1}}
```

`\DTLfmtcurr` Format currency according to the given currency code or use the default format if not defined. May fully expand (unless currency symbol is fragile, which is less likely with newer LaTeX kernel).

```
\newcommand{\DTLfmtcurr}[2]{%
  \cs_if_exist:cTF { dtl@curr@ #1 @fmt }
   {
     \use:c { dtl@curr@ #1 @fmt } { \DTLcurr { #1 } } { #2 }
   }
   { \DTLcurrency { #2 } }
}
```

`\DTLfmtcurrency` Format currency using given symbol.

```
\newcommand{\DTLfmtcurrency}{\dtlcurrdefaultfmt}
```

`\dtlcurrdefaultfmt`

```
\newcommand{\dtlcurrdefaultfmt}{\dtlcurrprefixfmt}
```

`\dtlcurrprefixfmt`

```
\newcommand{\dtlcurrprefixfmt}[2]{
```

```
 \datatool_prefix_adjust_sign:nnn { #1 } { \dtlcurrfmtsep } { #2 }
}

\cs_new:Nn \datatool_prefix_adjust_sign:nnn
 {
   \bool_lazy_or:nnTF
      { \tl_if_head_eq_charcode_p:nN { #3 } + }
      { \tl_if_head_eq_charcode_p:nN { #3 } - }
    {
      \exp_args:Ne \datatool_adjust_sign_fmt:n { \tl_head:n { #3 } }
      #1 #2
      \tl_tail:n { #3 }
    }
    { #1 #2 #3 }
 }
```

Allow the sign to be formatted if outside of math mode.

```
\cs_new:Nn \datatool_adjust_sign_fmt:n
 {
   \ifmmode
     #1
   \else
    \tl_if_head_eq_charcode:nNTF { #1 } -
     {
       \textminus
     }
     { #1 }
  \fi
 }
```

`\dtlcurrsuffixfmt`

```
\newcommand{\dtlcurrsuffixfmt}[2]{
 \datatool_suffix_adjust_sign:nnn { #1 } { \dtlcurrfmtsep } { #2 }
}

\cs_new:Nn \datatool_suffix_adjust_sign:nnn
 {
   \bool_lazy_or:nnTF
      { \tl_if_head_eq_charcode_p:nN { #3 } + }
      { \tl_if_head_eq_charcode_p:nN { #3 } - }
    {
      \exp_args:Ne \datatool_adjust_sign_fmt:n { \tl_head:n { #3 } }
      \tl_tail:n { #3 }
       #2 #1
    }
    { #3 #2 #1 }
 }
```

`\dtlcurrfmtsymsep`

```
\newcommand{\dtlcurrfmtsymsep}{}
```

```
\ExplSyntaxOff
```

\dtlcurrfmtsep

```
\newcommand{\dtlcurrfmtsep}{\DTLcurrCodeOrSymOrChar{~}{\dtlcurrfmtsymsep}{\dtlcurrfmtsym
```

```
\ExplSyntaxOn
```

\DTLcurr

```
\newcommand{\DTLcurr}[1]{\ifcsdef{DTLcurr#1}{\csuse{DTLcurr#1}}{#1}}
```

\DTLdefaultEURcurrencyfmt

```
\newcommand{\DTLdefaultEURcurrencyfmt}{\dtlcurrdefaultfmt}
```

\DTLcurrXXX

```
\datatool_def_currency:nnV
 { XXX }
 { \textcurrency }
 \l_datatool_currency_str
```

\DTLcurrXBT

```
\cs_if_exist:NT \faBtc
 {
   \datatool_def_currency:nnV
     { XBT }
     { \faBtc }
     \l_datatool_bitcoin_str
 }
```

Try to guess the command used for the Euro currency symbol. Mainly provided to retain backward compatibility with pre v3.0. If incorrect, this may be changed with `\datatool_set_currency_symbol:nn` Other currencies can be defined in the applicable region file.

```
\tl_new:N \l__datatool_eurocs_tl
\cs_if_exist:NTF \euro
 {
   \tl_set:Nn \l__datatool_eurocs_tl { \euro }
 }
 {
  \cs_if_exist:NTF \Euro
    {
      \tl_set:Nn \l__datatool_eurocs_tl { \Euro }
    }
    {
     \cs_if_exist:NTF \EUR
     {
       \tl_set:Nn \l__datatool_eurocs_tl { \EUR }
     }
     {
```

```
                \cs_if_exist:NTF \faEur
                {
                  \tl_set:Nn \l__datatool_eurocs_tl { \faEur }
                }
                {
                  \cs_if_exist:NTF \wasyeuro
                  {
                    \tl_set:Nn \l__datatool_eurocs_tl { \wasyeuro }
                  }
                  {
                    \cs_if_exist:NTF \texteuro
                    {
                      \tl_set:Nn \l__datatool_eurocs_tl { \texteuro }
                    }
                    {
                      \tl_set:Nn \l__datatool_eurocs_tl { \l_datatool_euro_str }
                    }
                  }
                }
              }
            }
          }

\DTLcurrEUR
          \datatool_def_currency:nnVV
           { \DTLdefaultEURcurrencyfmt }
           { EUR }
           \l__datatool_eurocs_tl
           \l_datatool_euro_str

\datatoolSetCurrencySort
          \newcommand \datatoolSetCurrencySort
           {
             \let \textdollar \c_dollar_str
             \let \textdollaroldstyle \c_dollar_str
             \let \textcentoldstyle \l_datatool_cent_str
             \let \textcent \l_datatool_cent_str
             \let \textsterling \l_datatool_pound_str
             \let \pounds \l_datatool_pound_str
             \let \textcurrency \l_datatool_currency_str
             \let \textyen \l_datatool_yen_str
             \let \textflorin \l_datatool_florin_str
             \let \texteuro \l_datatool_euro_str
             \let \textcolonmonetary \l_datatool_colonsign_str
             \let \textwon \l_datatool_won_str
             \let \textnaira \l_datatool_naira_str
             \let \textguarani \l_datatool_guarani_str
             \let \textpeso \l_datatool_peso_str
             \let \textlira \l_datatool_lira_str
             \let \textdong \l_datatool_dong_str
```

```
    \let \textbaht \l_datatool_baht_str
  }
```

2.5 Floating Point Arithmetic

The commands defined in this section are designed for localised numeric values. They all have to first convert the formatted value to a numeric value acceptable to the underlying arithmetic function and then convert back again. If the original supplied values had different data types, the data type of the result depends on which type is dominant.

First provide some common functions to update the datum structure after operating on two values with potentially different types.

Determine dominant data type: decimal overrides integer, currency overrides integer and decimal, temporal values override other values, but time should be converted to datetime if added to a date or datetime or other numeric value, and date should be converted to datetime if added to anything other than an integer.

NB if two currencies are added, their symbols are assumed to represent the same currency unit. No exchange rate information is available.

```
\cs_new:Nn \__datatool_update_datatype:
 {
   \datatool_update_datatype:NNNN
     \@dtl@datatype
     \l__datatool_tmp_datatype_int
     \l__datatool_datum_currency_tl
     \l__datatool_tmp_currency_tl
 }
```

Syntax: ⟨*type1 int-var*⟩ ⟨*type2 int-var*⟩ ⟨*curr-sym1 tl-var*⟩ ⟨*curr-sym2 tl-var*⟩ This will update ⟨*type1 int-var*⟩ to the dominant type and (if the dominant type is currency) ⟨*curr-sym1 tl-var*⟩ to the currency symbol.

```
\cs_new:Nn \datatool_update_datatype:NNNN
 {
  \bool_lazy_or:nnF
     { \int_compare_p:nNn { #1 } = { #2 } }
     { \int_compare_p:nNn { #2 } = { \c_datatool_unknown_int } }
   {
    \int_compare:nNnTF { #1 } = { \c_datatool_unknown_int }
     {
       \int_set_eq:NN #1 #2
       \tl_set_eq:NN #3 #4
     }
     {
       \bool_lazy_or:nnTF
         { \datatool_if_temporal_datum_type_p:n { #1 } }
         { \datatool_if_temporal_datum_type_p:n { #2 } }
        {
```

If either is a temporal value (and already checked they are not the same type), the result will be a timestamp unless an integer has been added to a date.

```
          \bool_lazy_or:nnTF
```

```
            {
              \bool_lazy_and_p:nn
               { \int_compare_p:nNn { #1 } = { \c_datatool_date_int } }
               { \int_compare_p:nNn { #2 } = { \c_datatool_integer_int } }
            }
            {
              \bool_lazy_and_p:nn
               { \int_compare_p:nNn { #2 } = { \c_datatool_date_int } }
               { \int_compare_p:nNn { #1 } = { \c_datatool_integer_int } }
            }
            {
              \int_set_eq:NN #1 \c_datatool_date_int
            }
            {
              \int_set_eq:NN #1 \c_datatool_datetime_int
            }
```

Doesn't make sense to add a currency to a date/time value.

```
          \tl_clear:N #3
        }
        {
```

Neither is a temporal type, so order of precedence can simply be determined by the type's numeric ID.

```
         \int_compare:nNnT { #1 } < { #2 }
          {
           \int_set_eq:NN #1 #2
           \tl_set_eq:NN #3 #4
          }
        }
      }
    }
  }
```

Convert the numeric value to localised format. Note that the `\l__datatool_result_tl` should be correctly expanded before this function.

```
\cs_new:Nn \__datatool_assign_result:N
{
  \datatool_assign_result:NNNN #1
   \@dtl@datatype
   \l__datatool_result_tl
   \l__datatool_datum_currency_tl
}
```

Syntax: ⟨*tl-var*⟩ ⟨*type int-var*⟩ ⟨*value tl-var*⟩ ⟨*curr-sym tl-var*⟩

```
\cs_new:Nn \datatool_assign_result:NNNN
{
 \datatool_if_temporal_datum_type:nTF { #2 }
  {
    \datatool_decimal_to_temporal:Nnn
      #1 { #2 } { #3 }
```

```
  }
  {
    \tl_if_empty:NTF #4
     {
       \exp_args:NV \DTLdecimaltolocale #3 #1
     }
     {
       \exp_args:NVV \__datatool_decimal_to_currency:nnN
          #4 #3 #1
     }
  }
}
```

\datatool_numeric_fn:NnnNN
⟨*result tl-var*⟩ {⟨*num1*⟩} {⟨*num2*⟩}
⟨*decimal-operator-cs*⟩ ⟨*int-operator-cs*⟩

Perform a numerical operation on two values that need parsing or that are already in datum format. If they are both integers, use the ⟨*int-operator-cs*⟩ function, which should take two arguments and expand to the result, otherwise use the `\decimal-operator-cs` function, which should take three arguments where the first is the result token list variable and the others are the numeric values.

```
\cs_new:Nn \datatool_numeric_fn:NnnNN
 {
  \DTLconverttodecimal { #2 } \l__datatool_resulta_tl
  \int_set_eq:NN
    \l__datatool_tmp_datatype_int
    \@dtl@datatype
  \tl_set_eq:NN
    \l__datatool_tmp_currency_tl
    \l__datatool_datum_currency_tl
  \DTLconverttodecimal { #3 } \l__datatool_resultb_tl
```

If the data types are different, need to determine the dominant one.

```
  \__datatool_update_datatype:
```

Determine which function should be used.

```
  \datatool_if_any_int_datum_type:nTF { \@dtl@datatype }
   {
```

Version 3.1: changed `\tl_set:Nx` to `\tl_set:Ne` (not sure if this makes difference).

```
     \tl_set:Ne \l__datatool_result_tl
      { #5 { \l__datatool_resulta_tl } { \l__datatool_resultb_tl } }
   }
   {
     #4
       { \l__datatool_result_tl }
       { \l__datatool_resulta_tl }
```

```
      { \l__datatool_resultb_tl }
   }
  \__datatool_assign_result:N #1
 }
```

\datatool_numeric_fn:NnNN
⟨*result tl-var*⟩ {⟨*num*⟩}
⟨*decimal-operator-cs*⟩ ⟨*int-operator-cs*⟩

Perform a numerical operation on one value that needs parsing or that is already in datum format. If the value is an integer, use the ⟨*int-operator-cs*⟩ function, which should take one argument and expand to the result, otherwise use the `\decimal-operator-cs` function, which should take two arguments where the first is the result token list variable and the second is the numeric value.

```
\cs_new:Nn \datatool_numeric_fn:NnNN
 {
  \DTLconverttodecimal { #2 } \l__datatool_resulta_tl
```

Determine which function should be used.

```
  \datatool_if_any_int_datum_type:nTF { \@dtl@datatype }
   {
```

Version 3.1: changed `\tl_set:Nx` to `\tl_set:Ne` (not sure if this makes difference).

```
     \tl_set:Ne \l__datatool_result_tl
      { #4 { \l__datatool_resulta_tl }  }
   }
   {
     #3
       { \l__datatool_result_tl }
       { \l__datatool_resulta_tl }
   }
  \__datatool_assign_result:N #1
 }
```

\datatool_numeric_fn:NnN
⟨*result tl-var*⟩ {⟨*num*⟩}
⟨*decimal-operator-cs*⟩

Perform a numerical operation on one value that needs parsing or that is already in datum format. The calculation is performed by `\decimal-operator-cs` function, which should take two arguments where the first is the result token list variable and the second is the numeric value. The result is not expected to be an integer.

```
\cs_new:Nn \datatool_numeric_fn:NnN
 {
  \DTLconverttodecimal { #2 } \l__datatool_resulta_tl
   #3
     { \l__datatool_result_tl }
```

```
    { \l__datatool_resulta_tl }
  \datatool_if_any_int_datum_type:nTF { \@dtl@datatype }
   {
      \int_set_eq:NN \@dtl@datatype \c_datatool_decimal_int
   }
  \__datatool_assign_result:N #1
 }
```

\datatool_numeric_list_fn:NnNN
⟨*result tl-var*⟩ {⟨*num list*⟩}
⟨*decimal-operator-cs*⟩ ⟨*int-operator-cs*⟩

Perform a numerical operation on a list of values that need parsing or that are already in datum format. As \datatool_numeric_fn:NnnNN but iterates through the list performing the applicable function sequentially. The result will be set to \DTLnumbernull if the list is empty (and a warning will occur).

```
\cs_new:Nn \datatool_numeric_list_fn:NnNN
 {
```

Convert the list to the scratch sequence:

```
    \@dtl@assigntmpseq { #2 }
```

Keep a note of the number of items in the sequence if required.

```
    \int_set:Nn \l__datatool_count_int
      { \seq_count:N \l__datatool_tmp_seq }
```

If sequence is empty, set the result to number null:

```
    \int_if_zero:nTF { \l__datatool_count_int }
     {
       \PackageWarning { datatool-base }
          { empty ~ list ~ ` #2 ' ~ found ~ in ~ aggregate ~ function }
       \tl_set_eq:NN #1 \DTLnumbernull
     }
     {
```

Pop the first item from the sequence:

```
       \seq_pop_left:NN \l__datatool_tmp_seq \l__datatool_resulta_tl
       \exp_args:NV \DTLconverttodecimal
         \l__datatool_resulta_tl
         \l__datatool_result_tl
       \seq_map_inline:Nn \l__datatool_tmp_seq
        {
```

Save previous:

```
          \int_set_eq:NN
            \l__datatool_tmp_datatype_int
            \@dtl@datatype
          \tl_set_eq:NN
            \l__datatool_tmp_currency_tl
            \l__datatool_datum_currency_tl
```

```
          \tl_set_eq:NN
            \l__datatool_resulta_tl
            \l__datatool_result_tl
```

Parse this value.

```
          \DTLconverttodecimal { ##1 } \l__datatool_resultb_tl
```

If the data types are different, need to determine the dominant one.

```
          \__datatool_update_datatype:
```

Determine which function should be used.

```
          \datatool_if_any_int_datum_type:nTF { \@dtl@datatype }
           {
```

Version 3.1: changed `\tl_set:Nx` to `\tl_set:Ne` (not sure if this makes difference).

```
              \tl_set:Ne \l__datatool_result_tl
               { #4 { \l__datatool_resulta_tl } { \l__datatool_resultb_tl } }
           }
           {
             #3
               { \l__datatool_result_tl }
               { \l__datatool_resulta_tl }
               { \l__datatool_resultb_tl }
           }
       }
      \__datatool_assign_result:N #1
    }
}
```

`\datatool_decimal_list_fn:NnN`
⟨*result fp-var*⟩ {⟨*num list*⟩}
⟨*fp-operator-cs*⟩

Similar to the above but the result is expected to be a floating point variable and the function should by a l2fp update function, such as `\fp_add:Nn`.

```
\cs_new:Nn \datatool_decimal_list_fn:NnN
 {
```

Convert the list to the scratch sequence:

```
    \@dtl@assigntmpseq { #2 }
```

Keep a note of the number of items in the sequence if required.

```
    \int_set:Nn \l__datatool_count_int
      { \seq_count:N \l__datatool_tmp_seq }
```

Trigger an error if the list is empty:

```
    \int_if_zero:nTF { \l__datatool_count_int }
     {
       \PackageError { datatool-base }
         { empty ~ list ~ ` #2 ' ~ found ~ in ~ aggregate ~ function }
```

```
          { one ~ or ~ more ~ numeric ~ items ~ are ~ expected }
      }
      {
```

Pop the first item from the sequence:

```
      \seq_pop_left:NN \l__datatool_tmp_seq \l__datatool_resulta_tl
      \datatool_set_fp:NV #1 \l__datatool_resulta_tl
      \seq_map_inline:Nn \l__datatool_tmp_seq
       {
```

Save previous datum information:

```
        \int_set_eq:NN
          \l__datatool_tmp_datatype_int
          \@dtl@datatype
        \tl_set_eq:NN
          \l__datatool_tmp_currency_tl
          \l__datatool_datum_currency_tl
```

Get this value as fp variable.

```
        \datatool_set_fp:Nn \l__datatool_tmpa_fp { ##1 }
```

If the data types are different, need to determine the dominant one.

```
        \__datatool_update_datatype:
```

Perform the update.

```
            #3 #1 { \l__datatool_tmpa_fp }
        }
      }
    }
```

`\DTLadd` `\DTLadd{`⟨*cmd*⟩`}{`⟨*num1*⟩`}{`⟨*num2*⟩`}`

Sets ⟨*cmd*⟩ = ⟨*num1*⟩ + ⟨*num2*⟩

```
\NewDocumentCommand \DTLadd { m m m }
{
  \datatool_numeric_fn:NnnNN
    #1 { #2 } { #3 } \dtladd \datatool_int_add:nn
}
```

For use in the above and related commands:

```
\cs_new:Nn \datatool_int_add:nn { \int_eval:n { #1 + #2 } }
```

`\DTLgadd` Global version

```
\NewDocumentCommand \DTLgadd { m m m }
{
  \DTLadd { \l__datatool_resulta_tl } { #2 } { #3 }
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

\DTLaddall

\DTLaddall{⟨*cmd*⟩}{⟨*num list*⟩}

Sums all the values in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\NewDocumentCommand \DTLaddall { m m }
{
  \datatool_numeric_list_fn:NnNN
    #1 { #2 } \dtladd \datatool_int_add:nn
}
```

\DTLgaddall

\DTLgaddall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\NewDocumentCommand \DTLgaddall { m m }
{
  \DTLaddall { \l__datatool_resultb_tl } { #2 }
  \tl_gset_eq:NN #1 \l__datatool_resultb_tl
}
```

\DTLsub

\DTLsub{⟨*cmd*⟩}{⟨*num1*⟩}{⟨*num2*⟩}

Sets ⟨*cmd*⟩ = ⟨*num1*⟩ - ⟨*num2*⟩

```
\NewDocumentCommand \DTLsub { m m m }
{
  \datatool_numeric_fn:NnnNN
    #1 { #2 } { #3 } \dtlsub \datatool_int_sub:nn
}
```

For use in the above:

```
\cs_new:Nn \datatool_int_sub:nn { \int_eval:n { #1 - ( #2 ) } }
```

\DTLgsub Global version

```
\NewDocumentCommand \DTLgsub { m m m }
{
  \DTLsub { \l__datatool_resulta_tl } {#2} {#3}
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

\DTLmul

\DTLmul{⟨*cmd*⟩}{⟨*num1*⟩}{⟨*num2*⟩}

Sets ⟨*cmd*⟩ = ⟨*num1*⟩ × ⟨*num2*⟩

```
\NewDocumentCommand \DTLmul { m m m }
{
  \datatool_numeric_fn:NnnNN
```

```
    #1 { #2 } { #3 } \dtlmul \datatool_int_mul:nn
}
```

For use in the above:

```
\cs_new:Nn \datatool_int_mul:nn { \int_eval:n { ( #1 ) * ( #2 ) } }
```

`\DTLgmul` Global version

```
\newcommand*{\DTLgmul}[3]{%
  \DTLmul{ \l__datatool_resulta_tl } {#2} {#3}
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

`\DTLdiv`

```
\DTLdiv{⟨cmd⟩}{⟨num1⟩}{⟨num2⟩}
```

Sets ⟨*cmd*⟩ = ⟨*num1*⟩ / ⟨*num2*⟩

```
\NewDocumentCommand \DTLdiv { m m m }
{
  \datatool_numeric_fn:NnnNN
    #1 { #2 } { #3 } \dtldiv \int_div_round:nn
}
```

`\DTLgdiv` Global version

```
\NewDocumentCommand \DTLgdiv { m m m }
{
  \DTLdiv{ \l__datatool_resulta_tl } {#2} {#3}
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

`\DTLabs`

```
\DTLabs{⟨cmd⟩}{⟨num⟩}
```

Sets ⟨*cmd*⟩ = abs(⟨*num*⟩)

```
\NewDocumentCommand \DTLabs { m m }
{
  \datatool_numeric_fn:NnNN
    #1 { #2 } \dtlabs \int_abs:n
}
```

`\DTLgabs` Global version

```
\NewDocumentCommand \DTLgabs { m m }
{
  \DTLabs{ \l__datatool_resulta_tl } {#2}
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

`\DTLneg`

```
\DTLneg{⟨cmd⟩}{⟨num⟩}
```

Sets ⟨*cmd*⟩ = -⟨*num*⟩

```
\NewDocumentCommand \DTLneg { m m }
{
  \datatool_numeric_fn:NnNN
    #1 { #2 } \dtlneg \datatool_int_neg:n
}
```

Integer negation function for use in the above.

```
\cs_new:Nn \datatool_int_neg:n
  { \int_eval:n { - ( #1 ) } }
```

`\DTLgneg` Global version

```
\NewDocumentCommand \DTLgneg { m m }
{
  \DTLneg{ \l__datatool_resulta_tl } {#2}
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

`\DTLsqrt` `\DTLsqrt{`⟨*cmd*⟩`}{`⟨*num*⟩`}`

Sets ⟨*cmd*⟩ = sqrt(⟨*num*⟩)

```
\NewDocumentCommand \DTLsqrt { m m }
{
  \datatool_numeric_fn:NnN #1 { #2 } \dtlroot
}
```

`\DTLgsqrt` Global version

```
\NewDocumentCommand \DTLgsqrt { m m }
{
  \DTLsqrt{ \l__datatool_resulta_tl } {#2}
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

`\DTLmin` `\DTLmin{`⟨*cmd*⟩`}{`⟨*num1*⟩`}{`⟨*num2*⟩`}`

Sets ⟨*cmd*⟩ = min(⟨*num1*⟩, ⟨*num2*⟩)

```
\NewDocumentCommand \DTLmin { m m m }
{
  \datatool_numeric_fn:NnnNN
    #1 { #2 } { #3 } \dtlmin \int_min:nn
}
```

`\DTLgmin` Global version

```
\NewDocumentCommand \DTLgmin { m m m }
{
  \DTLmin{ \l__datatool_resulta_tl } {#2} {#3}
```

```
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

\DTLminall

\DTLminall{⟨*cmd*⟩}{⟨*num list*⟩}

Finds the minimum value in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\NewDocumentCommand \DTLminall { m m }
{
  \datatool_numeric_list_fn:NnNN
    #1 { #2 } \dtlmin \int_min:nn
}
```

\DTLgminall

\DTLgminall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\NewDocumentCommand \DTLgminall { m m }
{
  \DTLminall{ \l__datatool_resultb_tl } { #2 }
  \tl_gset_eq:NN #1 \l__datatool_resultb_tl
}
```

\DTLmax

\DTLmax{⟨*cmd*⟩}{⟨*num1*⟩}{⟨*num2*⟩}

Sets ⟨*cmd*⟩ = max(⟨*num1*⟩, ⟨*num2*⟩)

```
\NewDocumentCommand \DTLmax { m m m }
{
  \datatool_numeric_fn:NnnNN
    #1 { #2 } { #3 } \dtlmax \int_max:nn
}
```

\DTLgmax

Global version

```
\NewDocumentCommand \DTLgmax { m m m }
{
  \DTLmax { \l__datatool_resultb_tl } { #2 } { #3 }
  \tl_gset_eq:NN #1 \l__datatool_resultb_tl
}
```

\DTLmaxall

\DTLmaxall{⟨*cmd*⟩}{⟨*num list*⟩}

Finds the maximum value in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\NewDocumentCommand \DTLmaxall { m m }
{
  \datatool_numeric_list_fn:NnNN
    #1 { #2 } \dtlmax \int_max:nn
}
```

\DTLgmaxall

\DTLgmaxall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\NewDocumentCommand \DTLgmaxall { m m }
{
  \DTLmaxall{ \l__datatool_resultb_tl } { #2 }
  \tl_gset_eq:NN #1 \l__datatool_resultb_tl
}
```

\DTLmeanforall

\DTLmeanforall{⟨*cmd*⟩}{⟨*num list*⟩}

Computes the arithmetic mean of all the values in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\NewDocumentCommand \DTLmeanforall { m m }
{
  \datatool_decimal_list_fn:NnN
    \l__datatool_total_fp { #2 } \fp_add:Nn
  \int_if_zero:nTF { \l__datatool_count_int }
   {
     \tl_set_eq:NN #1 \DTLnumbernull
   }
   {
     \fp_set:Nn
       \l__datatool_mean_fp
       { \l__datatool_total_fp / \l__datatool_count_int }

     \tl_set:Ne
       \l__datatool_result_tl
       { \fp_use:N \l__datatool_mean_fp }
     \__datatool_assign_result:N #1
   }
}
```

\DTLgmeanforall

\DTLgmeanforall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\NewDocumentCommand \DTLgmeanforall { m m }
{
  \DTLmeanforall{ \l__datatool_resultb_tl } { #2 }
```

```
  \tl_gset_eq:NN #1 \l__datatool_resultb_tl
}
```

\DTLvarianceforall

\DTLvarianceforall{⟨*cmd*⟩}{⟨*num list*⟩}

Computes the variance of all the values in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence. This is more complicated than the previous aggregate functions.

```
\NewDocumentCommand \DTLvarianceforall { m m }
{
  \@dtl@assigntmpseq{#2}
  \int_zero:N \l__datatool_count_int
  \fp_zero:N \l__datatool_total_fp
  \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int
  \tl_clear:N \l__datatool_datum_currency_tl
  \seq_clear:N \l__datatool_tmpb_seq
  \seq_map_inline:Nn \l__datatool_tmp_seq
   {
     \int_incr:N \l__datatool_count_int
```

Save previous datum information:

```
     \int_set_eq:NN
       \l__datatool_tmp_datatype_int
       \@dtl@datatype
     \tl_set_eq:NN
       \l__datatool_tmp_currency_tl
       \l__datatool_datum_currency_tl
```

Convert:

```
     \DTLconverttodecimal { ##1 } \l__datatool_resulta_tl
     \__datatool_update_datatype:
     \seq_put_right:No \l__datatool_tmpb_seq { \l__datatool_resulta_tl }
     \fp_add:Nn
       \l__datatool_total_fp
       { \l__datatool_resulta_tl }
   }
  \int_if_zero:nTF { \l__datatool_count_int }
   {
     \PackageWarning { datatool-base }
        { empty ~ list ~ ` #2 ' ~ found ~ in ~ aggregate ~ function }
     \tl_set_eq:NN #1 \DTLnumbernull
   }
   {
    \fp_set:Nn
      \l__datatool_mean_fp
      {  \l__datatool_total_fp  /  \l__datatool_count_int  }
    \fp_zero:N \l__datatool_tmpa_fp
    \seq_map_inline:Nn \l__datatool_tmpb_seq
     {
```

```
        \fp_set:Nn \l__datatool_tmpb_fp
         {
           ##1 - \l__datatool_mean_fp
         }
        \fp_add:Nn \l__datatool_tmpa_fp
         {
            \l__datatool_tmpb_fp *  \l__datatool_tmpb_fp
         }
      }
     \fp_set:Nn \l__datatool_tmpa_fp { \l__datatool_tmpa_fp / \l__datatool_count_int }

     \tl_set:Ne
       \l__datatool_result_tl
       { \fp_use:N \l__datatool_tmpa_fp }
     \__datatool_assign_result:N #1
    }
}
```

\DTLgvarianceforall

\DTLgvarianceforall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\NewDocumentCommand \DTLgvarianceforall { m m }
{
  \DTLvarianceforall { \l__datatool_resultb_tl } { #2 }
  \tl_gset_eq:NN #1 \l__datatool_resultb_tl
}
```

\DTLsdforall

\DTLsdforall{⟨*cmd*⟩}{⟨*num list*⟩}

Computes the standard deviation of all the values in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\NewDocumentCommand \DTLsdforall { m m }
{
  \@dtl@assigntmpseq{#2}
  \int_zero:N \l__datatool_count_int
  \fp_zero:N \l__datatool_total_fp
  \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int
  \tl_clear:N \l__datatool_datum_currency_tl
  \seq_clear:N \l__datatool_tmpb_seq
  \seq_map_inline:Nn \l__datatool_tmp_seq
   {
     \int_incr:N \l__datatool_count_int
```

Save previous datum information:

```
      \int_set_eq:NN
        \l__datatool_tmp_datatype_int
        \@dtl@datatype
```

```
      \tl_set_eq:NN
        \l__datatool_tmp_currency_tl
        \l__datatool_datum_currency_tl
```

Convert:

```
      \DTLconverttodecimal { ##1 } \l__datatool_resulta_tl
      \__datatool_update_datatype:
      \seq_put_right:No \l__datatool_tmpb_seq { \l__datatool_resulta_tl }
      \fp_add:Nn
        \l__datatool_total_fp
        { \l__datatool_resulta_tl }
    }
   \int_if_zero:nTF { \l__datatool_count_int }
    {
      \PackageWarning { datatool-base }
         { empty ~ list ~ ` #2 ' ~ found ~ in ~ aggregate ~ function }
      \tl_set_eq:NN #1 \DTLnumbernull
    }
    {
      \fp_set:Nn
        \l__datatool_mean_fp
        {  \l__datatool_total_fp  /  \l__datatool_count_int  }
      \fp_zero:N \l__datatool_tmpa_fp
      \seq_map_inline:Nn \l__datatool_tmpb_seq
       {
         \fp_set:Nn \l__datatool_tmpb_fp
          {
            ##1 - \l__datatool_mean_fp
          }
         \fp_add:Nn \l__datatool_tmpa_fp
          {
             \l__datatool_tmpb_fp *  \l__datatool_tmpb_fp
          }
       }
      \fp_set:Nn \l__datatool_tmpa_fp
         { sqrt ( \l__datatool_tmpa_fp / \l__datatool_count_int ) }

      \tl_set:Ne
        \l__datatool_result_tl
        { \fp_use:N \l__datatool_tmpa_fp }
      \__datatool_assign_result:N #1
    }
}
```

\DTLgsdforall

\DTLgsdforall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\NewDocumentCommand \DTLgsdforall { m m }
{
```

```
  \DTLsdforall { \l__datatool_resultb_tl } { #2 }
  \tl_gset_eq:NN #1 \l__datatool_resultb_tl
}
```

\DTLround

```
\DTLround{⟨cmd⟩}{⟨num⟩}{⟨num digits⟩}
```

Sets ⟨*cmd*⟩ to ⟨*num*⟩ rounded to ⟨*num digits*⟩ digits after the decimal character.

```
\NewDocumentCommand \DTLround { m m m }
{
  \DTLconverttodecimal{#2} \l__datatool_result_tl
  \dtlround
    { \l__datatool_result_tl }
    { \l__datatool_result_tl }
    { #3 }
  \__datatool_assign_result:N #1
}
```

\DTLground Global version

```
\NewDocumentCommand \DTLground { m m m }
{
  \DTLround { \l__datatool_resulta_tl } { #2 } { #3 }
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

\DTLtrunc

```
\DTLtrunc{⟨cmd⟩}{⟨num⟩}{⟨num digits⟩}
```

Sets ⟨*cmd*⟩ to ⟨*num*⟩ truncated to ⟨*num digits*⟩ digits after the decimal character.

```
\NewDocumentCommand \DTLtrunc { m m m }
{
  \DTLconverttodecimal { #2 } \l__datatool_result_tl
  \dtltrunc
    { \l__datatool_result_tl }
    { \l__datatool_result_tl }
    { #3 }
  \__datatool_assign_result:N #1
}
```

\DTLgtrunc Global version

```
\NewDocumentCommand \DTLgtrunc { m m m }
{
  \DTLtrunc{ \l__datatool_resulta_tl } {#2} {#3}
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

\DTLclip

```
\DTLclip{⟨cmd⟩}{⟨num⟩}
```

Sets ⟨*cmd*⟩ to ⟨*num*⟩ with all unnecessary 0's removed.

```
\NewDocumentCommand \DTLclip { m m }
{
  \DTLconverttodecimal{#2} \l__datatool_result_tl
  \dtlclip
    { \l__datatool_result_tl }
    { \l__datatool_result_tl }
  \__datatool_assign_result:N #1
}
```

`\DTLgclip` Global version

```
\NewDocumentCommand \DTLgclip { m m }
{
  \DTLclip{ \l__datatool_resulta_tl } {#2}
  \tl_gset_eq:NN #1 \l__datatool_resulta_tl
}
```

2.6 String Macros

`ntLocaleGetInitialLetter`

```
\newcommand{\DTLCurrentLocaleGetInitialLetter}[2]{%
  \datatool_get_first_letter:nN { #1 } #2
}
```

```
\regex_new:N \l__datatool_initial_cs_regex
\regex_set:Nn \l__datatool_initial_cs_regex { \A \c{.+} \cB(.) }
```

`\DTLGetInitialLetter` Designed to obtain the first letter for initials or letter groups. This can't simply grab the first token as it may be a multi-byte character or may start with a command.

```
\NewDocumentCommand{\DTLGetInitialLetter} { m m }
 {
   \bool_if:NTF \l__datatool_initial_purify_early_bool
   {
     \tl_if_head_is_group:nTF { #1 }
      {
        \tl_set:Nx #2 { \text_purify:n { \tl_head:n { #1 } } }
      }
      {
        \exp_args:Nx \__datatool_get_initial_letter:nN
          { \text_purify:n { #1 } } #2
      }
   }
   {
     \__datatool_get_initial_letter:nN { #1 } #2
   }
 }
\cs_new:Nn \__datatool_get_initial_letter:nN
 {
   \tl_if_blank:nTF { #1 }
```

```
    {
      \tl_clear:N #2
    }
    {
      \tl_if_head_is_group:nTF { #1 }
       {
         \tl_set:Nx #2 { \tl_head:n { #1 } }
         \tl_set:Nx #2 { \text_purify:n { #2 } }
       }
       {
        \regex_match:NnTF \l__datatool_initial_cs_regex { #1 }
         {
           \tl_set:Nx #2 { \tl_tail:n { #1 } }
           \exp_args:NNx \tl_set:Nx #2 { \exp_args:No \tl_head:n { #2 } }
           \exp_args:No \DTLCurrentLocaleGetInitialLetter { #2 } { #2 }
           \tl_set:Nx #2 { \tl_head:n { #1 } { \exp_not:o { #2 } } }
         }
         {
           \DTLCurrentLocaleGetInitialLetter { #1 } { #2 }
         }
       }
    }
}
```

Get the first grapheme (which may be letter or punctuation):

```
\cs_new:Nn \datatool_get_first_grapheme:nN
 {
   \tl_clear:N #2
   \exp_args:Nx \text_map_inline:nn { \text_purify:n { #1 } }
   {
     \tl_set:Nn #2 { ##1 }
     \text_map_break:
   }
 }
```

Getting an initial letter but skipping leading punctuation is awkward because `[:alpha:]` only matches ASCII letters. Need to also allow for the fact that UTF-8 characters are actually an active character with one or more arguments with inputenc and it's also necessary to allow for control characters that are used to adjust sorting which have been given a letter character code.

So the following first tests if the category code is a letter (which will be the case for UTF-8 letters with XeLaTeX and LuaLaTeX but not with pdfLaTeX). If not a letter category code, then the next part is a bit of a hack and is based on the assumption that all letters have a different upper and lower representation. The first argument should be a single grapheme (which may be a multi-byte character). It won't work if the first argument consists of a mixture of letters and non-letters.

```
\prg_new_conditional:Npnn \datatool_if_letter:n #1
  { T, F, TF }
 {
```

```
    \tl_if_head_eq_catcode:nNTF { #1 } \c_catcode_letter_token
     { \prg_return_true: }
     {
       \exp_args:Nee \tl_if_eq:nnTF
        { \text_uppercase:n { #1 } } { \text_lowercase:n { #1 } }
        { \prg_return_false: }
        { \prg_return_true: }
     }
  }
```

Get the first letter (skipping any preceding non-letters).

```
 \cs_new:Nn \datatool_get_first_letter:nN
  {
    \tl_clear:N #2
    \exp_args:Nx \text_map_inline:nn { \text_purify:n { #1 } }
    {
      \datatool_if_letter:nT { ##1 }
       {
         \tl_set:Nn #2 { ##1 }
         \text_map_break:
       }
    }
  }
```

Convert a string into a sequence of words. NB `[:alpha:]` only matches ASCII letters (see above). Multi-byte characters match `[:punct:]`.

```
 \regex_new:N \l_datatool_word_head_regex
 \regex_set:Nn \l_datatool_word_head_regex
  {
     ([[:punct:][:digit:]]*)
     ([^[:punct:][:digit:][:space:]\~]+?)
     ([[:punct:][:digit:]]*)
     ((?:[[:space:]\~]|\c{protect}?\c{(?:nobreak)?space\ ?})+)
  }
 \regex_new:N \l_datatool_word_hyphen_regex
 \regex_set:Nn \l_datatool_word_hyphen_regex
  {
     ([[:punct:][:digit:]]*)
     ([^[:punct:][:digit:][:space:]\~]+?)
     ([[:punct:][:digit:]]*)
     -{1}
  }
 \regex_new:N \l_datatool_word_apos_regex
 \regex_set:Nn \l_datatool_word_apos_regex
  {
     ([[:punct:][:digit:]]*)
     ([^[:punct:][:digit:][:space:]\~]+)
     \ur{l_datatool_apos_regex}
     (
       [^[:punct:][:digit:][:space:]\~]+
```

```
      (?:\ur{l_datatool_apos_regex}[^[:punct:][:digit:][:space:]\~]+)*
    )
    ([[:punct:][:digit:]]*)
    ((?:[[:space:]\~]|\c{protect}?\c{(?:nobreak)?space\ ?})+)
 }
\regex_new:N \l_datatool_word_apos_hyphen_regex
\regex_set:Nn \l_datatool_word_apos_hyphen_regex
 {
    ([[:punct:][:digit:]]*)
    ([^[:punct:][:digit:][:space:]\~]+)
    \ur{l_datatool_apos_regex}
    (
      [^[:punct:][:digit:][:space:]\~]+
      (?:\ur{l_datatool_apos_regex}[^[:punct:][:digit:][:space:]\~]+)*
    )
    ([[:punct:][:digit:]]*)
    -{1}
 }
\regex_new:N \l_datatool_word_apos_tail_regex
\regex_set:Nn \l_datatool_word_apos_tail_regex
 {
    ([[:punct:][:digit:]]*)
    ([^[:punct:][:digit:][:space:]\~]+)
    \ur{l_datatool_apos_regex}
    (
      [^[:punct:][:digit:][:space:]\~]+
      (?:\ur{l_datatool_apos_regex}[^[:punct:][:digit:][:space:]\~]+)*
    )
    ([[:punct:][:digit:]]*)
    \Z
 }
\regex_new:N \l_datatool_word_tail_regex
\regex_set:Nn \l_datatool_word_tail_regex
 {
    ([[:punct:][:digit:]]*)
    ([^[:punct:][:digit:][:space:]\~]+)
    ([[:punct:][:digit:]]*)
    \Z
 }
\regex_new:N \l_datatool_symbols_regex
\regex_set:Nn \l_datatool_symbols_regex
 {
    ((?:[!-@\[\]\^\_`\{\|\}]|\c{[[:alpha:][:punct:]]+})+?)
    ((?:[[:space:]\-\~]|\c{(?:nobreak)?space})+)
 }
\regex_new:N \l_datatool_other_regex
\regex_set:Nn \l_datatool_other_regex
 {
    (.+?)
    ((?:[[:space:]\-\~]|\c{(?:nobreak)?space})+)
```

```
 }
\cs_new:Nn \__datatool_leading_punc:n { \exp_not:n { #1 } }
\cs_new:Nn \__datatool_trailing_punc:n { \exp_not:n { #1 } }
\cs_new:Nn \__datatool_word:n { \exp_not:n { #1 } }
\cs_new:Nn \__datatool_word_hyphen:n { \exp_not:n { #1 } }
\cs_new:Nn \__datatool_word_apos:nn { \exp_not:n { #1 } ' \exp_not:n { #2 } }
\cs_new:Nn \__datatool_word_apos_hyphen:nn { \exp_not:n { #1 } ' \exp_not:n { #2 } }
\cs_new:Nn \__datatool_last_word_apos:nn { \exp_not:n { #1 } ' \exp_not:n { #2 } }
\cs_new:Nn \__datatool_last_word:n { \exp_not:n { #1 } }
\cs_new:Nn \__datatool_word_sep:n { \exp_not:n { #1 } }
\cs_new:Nn \__datatool_hyphen_sep: { - }
\cs_new:Nn \__datatool_symbol:n { \exp_not:n { #1 } }
\cs_new:Nn \datatool_parse_words:N
 {
   \exp_args:NNo \datatool_parse_words:Nn #1 { #1 }
 }
\cs_new:Nn \datatool_parse_words:Nn
 {
   \tl_set:Nx #1 { \tl_trim_spaces:n { #2 } }
   \regex_replace_case_all:nN
    {
      \l_datatool_word_head_regex
        {
          \c{__datatool_leading_punc:n}{\1}
          \c{__datatool_word:n}{\2}
          \c{__datatool_trailing_punc:n}{\3}
          \c{__datatool_word_sep:n}{\4}
        }
      \l_datatool_word_apos_hyphen_regex
        {
          \c{__datatool_leading_punc:n}{\1}
          \c{__datatool_word_apos_hyphen:nn}{\2}{\3}
          \c{__datatool_trailing_punc:n}{\4}
          \c{__datatool_hyphen_sep:n}
        }
      \l_datatool_word_hyphen_regex
        {
          \c{__datatool_leading_punc:n}{\1}
          \c{__datatool_word_hyphen:n}{\2}
          \c{__datatool_trailing_punc:n}{\3}
          \c{__datatool_hyphen_sep:n}
        }
      \l_datatool_word_apos_regex
        {
          \c{__datatool_leading_punc:n}{\1}
          \c{__datatool_word_apos:nn}{\2}{\3}
          \c{__datatool_trailing_punc:n}{\4}
          \c{__datatool_word_sep:n}{\5}
        }
      \l_datatool_word_apos_tail_regex
```

```
        {
          \c{__datatool_leading_punc:n}{\1}
          \c{__datatool_last_word_apos:nn}{\2}{\3}
          \c{__datatool_trailing_punc:n}{\4}
        }
      \l_datatool_word_tail_regex
        {
          \c{__datatool_leading_punc:n}{\1}
          \c{__datatool_last_word:n}{\2}
          \c{__datatool_trailing_punc:n}{\3}
        }
      \l_datatool_symbols_regex
        {
          \c{__datatool_symbol:n}{\1}
          \c{__datatool_word_sep:n}{\4}
        }
      \l_datatool_other_regex
        {
          \c{__datatool_symbol:n}{\1}
          \c{__datatool_word_sep:n}{\4}
        }
    }
    #1
}
```

\DTLinitials

\DTLinitials{⟨*string*⟩}

Convert a string into initials. (Any ~ character found is first converted into a space.) As from v3.0 this just uses `\DTLstoreinitials` with a temporary command. The extra grouping probably isn't necessary.

```
\NewDocumentCommand \DTLinitials { m }
{
  \group_begin:
   \DTLstoreinitials { #1 } { \l__datatool_tmpb_tl }
   \l__datatool_tmpb_tl
  \group_end:
}
```

\xDTLinitials

\xDTLinitials{⟨*cmd*⟩}

```
\NewDocumentCommand \xDTLinitials { m }
{
  \exp_args:No \DTLinitials { #1 }
}
```

\DTLstoreinitials

\DTLstoreinitials{⟨*string*⟩}{⟨*cmd*⟩}

Convert a string into initials and store in ⟨*cmd*⟩. (Any ~ character found is first converted into a space.)

```
\NewDocumentCommand \DTLstoreinitials { m m }
{
  \tl_clear:N #2
  \tl_if_empty:nF { #1 }
  {
    \datatool_parse_words:Nn \l__datatool_tmpa_tl { #1 }
    \tl_put_right:Nn \l__datatool_tmpa_tl { \q_recursion_tail }
    \__datatool_store_initials:N #2
    \q_recursion_stop
  }
}
\cs_new:Nn \__datatool_store_initials:N
 {
   \tl_set:Nx \l__datatool_tmp_initial_tl { \tl_head:N \l__datatool_tmpa_tl }
   \tl_set:Nx \l__datatool_tmpa_tl { \tl_tail:N \l__datatool_tmpa_tl }
   \tl_if_eq:NnTF
    \l__datatool_tmp_initial_tl { \__datatool_word:n }
    {
      \tl_set:Nx \l__datatool_tmp_initial_tl { \tl_head:N \l__datatool_tmpa_tl }
      \tl_set:Nx \l__datatool_tmpa_tl { \tl_tail:N \l__datatool_tmpa_tl }
      \exp_args:No \DTLStoreInitialGetLetter
         { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
      \tl_if_empty:NF \l__datatool_tmp_initial_tl
       {
         \tl_put_right:Nn #1 { \DTLinitialpunc }
         \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
         \tl_put_right:Nn #1 { { \DTLbetweeninitials } }
       }
    }
    {
      \tl_if_eq:NnTF
       \l__datatool_tmp_initial_tl { \__datatool_last_word:n }
       {
         \tl_set:Nx \l__datatool_tmp_initial_tl { \tl_head:N \l__datatool_tmpa_tl }
         \tl_set:Nx \l__datatool_tmpa_tl { \tl_tail:N \l__datatool_tmpa_tl }
         \exp_args:No \DTLStoreInitialGetLetter
            { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
         \tl_if_empty:NF \l__datatool_tmp_initial_tl
          {
            \tl_put_right:Nn #1 { \DTLinitialpunc }
            \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
            \tl_put_right:Nn #1 { { \DTLafterinitials } }
          }
       }
```

```
{
  \tl_if_eq:NnTF
   \l__datatool_tmp_initial_tl { \__datatool_word_hyphen:n }
   {
     \tl_set:Nx \l__datatool_tmp_initial_tl { \tl_head:N \l__datatool_tmpa_tl }
     \tl_set:Nx \l__datatool_tmpa_tl { \tl_tail:N \l__datatool_tmpa_tl }
     \exp_args:No \DTLStoreInitialGetLetter
       { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
     \tl_if_empty:NF \l__datatool_tmp_initial_tl
      {
        \tl_put_right:Nn #1 { \DTLinitialpunc }
        \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
        \tl_put_right:Nn #1 { { \DTLafterinitialbeforehyphen } }
        \tl_put_right:Nn #1 { \DTLinitialhyphen }
      }
   }
   {
     \tl_if_eq:NnTF
      \l__datatool_tmp_initial_tl { \__datatool_word_apos:nn }
      {
        \tl_set:Nx \l__datatool_tmp_initial_tl { \tl_head:N \l__datatool_tmpa_tl
        \tl_set:Nx \l__datatool_tmpa_tl { \tl_tail:N \l__datatool_tmpa_tl }
        \exp_args:No \DTLStoreInitialGetLetter
          { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
        \tl_if_empty:NTF \l__datatool_tmp_initial_tl
         {
           \tl_set:Nx \l__datatool_tmp_initial_tl { \tl_head:N \l__datatool_tmpa_
           \tl_set:Nx \l__datatool_tmpa_tl { \tl_tail:N \l__datatool_tmpa_tl }
           \exp_args:No \DTLStoreInitialGetLetter
             { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
           \tl_if_empty:NF \l__datatool_tmp_initial_tl
            {
              \tl_put_right:Nn #1 { \DTLinitialpunc }
              \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
              \tl_put_right:Nn #1 { { \DTLbetweeninitials } }
            }
         }
         {
           \tl_put_right:Nn #1 { \DTLaposinitialpunc }
           \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
           \tl_set:Nx \l__datatool_tmp_initial_tl { \tl_head:N \l__datatool_tmpa_
           \tl_set:Nx \l__datatool_tmpa_tl { \tl_tail:N \l__datatool_tmpa_tl }
           \exp_args:No \DTLStoreInitialGetLetter
             { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
           \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
           \tl_put_right:Nn #1 { { \DTLbetweeninitials } }
         }
      }
      {
        \tl_if_eq:NnTF
```

```
\l__datatool_tmp_initial_tl { \__datatool_word_apos_hyphen:nn }
{
  \tl_set:Nx \l__datatool_tmp_initial_tl { \tl_head:N \l__datatool_tmpa_
  \tl_set:Nx \l__datatool_tmpa_tl { \tl_tail:N \l__datatool_tmpa_tl }
  \exp_args:No \DTLStoreInitialGetLetter
    { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
  \tl_if_empty:NTF \l__datatool_tmp_initial_tl
   {
     \tl_set:Nx \l__datatool_tmp_initial_tl
       { \tl_head:N \l__datatool_tmpa_tl }
     \tl_set:Nx \l__datatool_tmpa_tl
       { \tl_tail:N \l__datatool_tmpa_tl }
     \exp_args:No \DTLStoreInitialGetLetter
       { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
     \tl_if_empty:NF \l__datatool_tmp_initial_tl
      {
        \tl_put_right:Nn #1 { \DTLinitialpunc }
        \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
        \tl_put_right:Nn #1 { { \DTLafterinitialbeforehyphen } }
        \tl_put_right:Nn #1 { \DTLinitialhyphen }
      }
   }
   {
     \tl_put_right:Nn #1 { \DTLaposinitialpunc }
     \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
     \tl_set:Nx \l__datatool_tmp_initial_tl
       { \tl_head:N \l__datatool_tmpa_tl }
     \tl_set:Nx \l__datatool_tmpa_tl
       { \tl_tail:N \l__datatool_tmpa_tl }
     \exp_args:No \DTLStoreInitialGetLetter
       { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
     \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
     \tl_put_right:Nn #1 { { \DTLafterinitialbeforehyphen } }
     \tl_put_right:Nn #1 { \DTLinitialhyphen }
   }
}
{
  \tl_if_eq:NnT
   \l__datatool_tmp_initial_tl { \__datatool_last_word_apos:nn }
   {
     \tl_set:Nx \l__datatool_tmp_initial_tl
        { \tl_head:N \l__datatool_tmpa_tl }
     \tl_set:Nx \l__datatool_tmpa_tl
        { \tl_tail:N \l__datatool_tmpa_tl }
     \exp_args:No \DTLStoreInitialGetLetter
       { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
     \tl_if_empty:NTF \l__datatool_tmp_initial_tl
      {
        \tl_set:Nx \l__datatool_tmp_initial_tl
          { \tl_head:N \l__datatool_tmpa_tl }
```

```
                                    \tl_set:Nx \l__datatool_tmpa_tl
                                      { \tl_tail:N \l__datatool_tmpa_tl }
                                    \exp_args:No \DTLStoreInitialGetLetter
                                      { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
                                    \tl_if_empty:NF \l__datatool_tmp_initial_tl
                                     {
                                       \tl_put_right:Nn #1 { \DTLinitialpunc }
                                       \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
                                       \tl_put_right:Nn #1 { { \DTLafterinitials } }
                                     }
                                  }
                                  {
                                    \tl_put_right:Nn #1 { \DTLaposinitialpunc }
                                    \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
                                    \tl_set:Nx \l__datatool_tmp_initial_tl
                                      { \tl_head:N \l__datatool_tmpa_tl }
                                    \tl_set:Nx \l__datatool_tmpa_tl
                                      { \tl_tail:N \l__datatool_tmpa_tl }
                                    \exp_args:No \DTLStoreInitialGetLetter
                                      { \l__datatool_tmp_initial_tl } \l__datatool_tmp_initial_tl
                                    \tl_put_right:Nx #1 { { \l__datatool_tmp_initial_tl } }
                                    \tl_put_right:Nn #1 { { \DTLafterinitials } }
                                  }
                               }
                            }
                         }
                      }
                   }
                 }
               \quark_if_recursion_tail_stop:N \l__datatool_tmpa_tl
               \__datatool_store_initials:N #1
             }
```

`DTLStoreInitialGetLetter`

```
\newcommand{\DTLStoreInitialGetLetter}[2]{\DTLGetInitialLetter{#1}{#2}}
```

`\DTLafterinitials` Defines what to do after the final initial.

```
\newcommand*{\DTLafterinitials}{.}
```

`\DTLbetweeninitials` Defines what to do between initials.

```
\newcommand*{\DTLbetweeninitials}{.}
```

`afterinitialbeforehyphen` Defines what to do before a hyphen.

```
\newcommand*{\DTLafterinitialbeforehyphen}{.}
```

`\DTLinitialhyphen` Defines what to do at the hyphen

```
\newcommand*{\DTLinitialhyphen}{-}
```

`\DTLinitialpunc`

```
\newcommand{\DTLinitialpunc}[2]{#1#2}
```

\DTLaposinitialpunc

```
\newcommand{\DTLaposinitialpunc}[3]{#1#3}
```

\DTLifAllUpperCase

\DTLifAllUpperCase{⟨*string*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

If ⟨*string*⟩ only contains uppercase characters do ⟨*true part*⟩, otherwise do ⟨*false part*⟩. This needs to take UTF-8 characters into account. NB regular expression `lower` and `upper` character classes doesn't match UTF-8 characters (regardless of the engine).

```
\newrobustcmd*{\DTLifAllUpperCase}[3]{%
  \tl_set:Nx \l__datatool_tmpa_tl { \text_purify:n { #1 } }
  \tl_set:Nx \l__datatool_tmpb_tl
    { \text_uppercase:n { \l__datatool_tmpa_tl } }
  \tl_if_eq:NNTF
   \l__datatool_tmpa_tl  \l__datatool_tmpb_tl
  { #2 } { #3 }
}
```

\DTLifAllLowerCase

\DTLifAllLowerCase{⟨*string*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

If ⟨*string*⟩ only contains lowercase characters do ⟨*true part*⟩, otherwise do ⟨*false part*⟩.

```
\newrobustcmd*{\DTLifAllLowerCase}[3]{%
  \tl_set:Nx \l__datatool_tmpa_tl { \text_purify:n { #1 } }
  \tl_set:Nx \l__datatool_tmpb_tl
    { \text_lowercase:n { \l__datatool_tmpa_tl } }
  \tl_if_eq:NNTF
   \l__datatool_tmpa_tl \l__datatool_tmpb_tl
  { #2 } { #3 }
}
```

\DTLsubstitute

\DTLsubstitute{⟨*cmd*⟩}{⟨*original*⟩}{⟨*replacement*⟩}

Substitutes first occurrence of ⟨*original*⟩ with {⟨*replacement*⟩} within the string given by ⟨*cmd*⟩

```
\newrobustcmd{\DTLsubstitute}[3]{%
 \tl_replace_once:Nnn #1 { #2 } { #3 }
}
```

\DTLsplitstring

\DTLsplitstring{⟨*string*⟩}{⟨*split text*⟩}{⟨*before cmd*⟩}{⟨*after cmd*⟩}

Splits string at ⟨*split text*⟩ stores the pre split text in ⟨*before cmd*⟩ and the post split text in ⟨*after cmd*⟩.

```
\newrobustcmd*{\DTLsplitstring}[4]{%
  \def\dtl@splitstr##1#2##2\@nil{%
    \def#3{##1}%
    \def#4{##2}%
    \ifdefempty{#4}%
    {%
      \let\@dtl@replaced=\@empty
    }%
    {%
      \def\@dtl@replaced{#2}%
      \dtl@split@str##2\@nil
    }%
  }%
  \def\dtl@split@str##1#2\@nil{\def#4{##1}}%
  \dtl@splitstr#1#2\@nil
}
```

\DTLxsplitstring

```
\DTLxsplitstring{⟨string⟩}{⟨split text⟩}{⟨before cmd⟩}{⟨after cmd⟩}
```

As above but expands the first two arguments

```
\newrobustcmd*{\DTLxsplitstring}[2]{%
  \exp_args:Noo \DTLsplitstring { #1 } { #2 }
}
```

\DTLsubstituteall

```
\DTLsubstituteall{⟨cmd⟩}{⟨original⟩}{⟨replacement⟩}
```

Substitutes all occurrences of ⟨*original*⟩ with {⟨*replacement*⟩} within the string given by ⟨*cmd*⟩

```
\newrobustcmd{\DTLsubstituteall}[3]{%
  \tl_replace_all:Nnn #1 { #2 } { #3 }
}
```

2.7 Conditionals

\if@dtl@condition

```
\newif\if@dtl@condition
```

2.7.1 Testing for a prefix

The following are used when splitting content.

```
\tl_new:N \l__datatool_prefix_tl
\tl_new:N \l__datatool_suffix_tl
```

Use to store the length:

```
\int_new:N \l__datatool_prefix_int
\int_new:N \l__datatool_suffix_int
```

Used to test for currency symbol at either end, but consider the symbol the ‘prefix’ and the rest the ‘suffix’ regardless of which way round they are. The first argument is the token list the second is the possible prefix or suffix.

```
\prg_new_conditional:Npnn \__datatool_if_starts_or_ends_with:nn #1 #2
 { T, F, TF}
 {
   \tl_clear:N \l__datatool_prefix_tl
   \tl_clear:N \l__datatool_suffix_tl
   \tl_if_eq:nnTF { #1 } { #2 }
    {
      \tl_set:Nn \l__datatool_prefix_tl { #2 }
      \prg_return_true:
    }
    {
     \int_set:Nn \l__datatool_prefix_int { \tl_count:n { #2 } }
     \int_set:Nn \l__datatool_suffix_int { \tl_count:n { #1 } }
     \int_compare:nNnTF { \l__datatool_suffix_int } > { \l__datatool_prefix_int }
      {
        \tl_if_eq:enTF
          { \tl_range:nnn { #1 } { \c_one_int } { \l__datatool_prefix_int } }
          { #2 }
         {
           \tl_set:Nn \l__datatool_prefix_tl { #2 }
           \tl_set:Ne \l__datatool_suffix_tl
             {
               \tl_range:nnn { #1 }
                { \l__datatool_prefix_int + \c_one_int }
                { \l__datatool_suffix_int }
             }
           \prg_return_true:
         }
         {
           \int_sub:Nn \l__datatool_suffix_int { \l__datatool_prefix_int }
           \tl_if_eq:enTF
             {
               \tl_range:nnn { #1 }
                 { \l__datatool_suffix_int + \c_one_int }
                 { - \c_one_int }
             }
             { #2 }
            {
              \tl_set:Nn \l__datatool_prefix_tl { #2 }
              \tl_set:Ne \l__datatool_suffix_tl
                {
                  \tl_range:nnn { #1 }
                   { \c_one_int }
```

```
                { \l__datatool_suffix_int }
              }
            \prg_return_true:
          }
          {
            \prg_return_false:
          }
       }
     }
     {
       \prg_return_false:
     }
   }
 }
\cs_generate_variant:Nn \__datatool_if_starts_or_ends_with:nnTF
 { VvTF , VeTF, VnTF }
```

The following tests if the token list in the first argument starts with the tokens in the second argument (without taking the category code into account). If true, the prefix token list will contain the tokens in the second argument (the prefix) and the suffix token list will contain the remaining tokens. If false the prefix token list will be empty and the suffix will contain all the tokens in the first argument.

```
\cs_new:Nn \__datatool_if_starts_with:NnTF
 {
   \exp_args:NV \__datatool_if_starts_with:nnTF #1 { #2 } { #3 } { #4 }
 }
\cs_new:Nn \__datatool_if_starts_with:nnTF
 {
```

Initialise the prefix to an empty list and the suffix to the token list under examination.

```
   \tl_clear:N \l__datatool_prefix_tl
   \tl_set:Nn \l__datatool_suffix_tl { #1 }
   \tl_set:Nn \l__datatool_tmpa_tl { #1 \q_recursion_tail }
   \tl_set:Nn \l__datatool_tmpb_tl { #2 \q_recursion_tail }
   \__datatool_if_starts_with:
   \q_recursion_stop
   \__datatool_result:nn { #3 } { #4 }
 }
\cs_new:Nn \__datatool_if_starts_with:
{
   \exp_args:No \tl_if_head_is_space:nTF { \l__datatool_tmpa_tl }
   {
     \tl_set:Nn \l__datatool_tmpc_tl { ~ }
     \tl_set:Nx \l__datatool_tmpa_tl
       { \tl_head:N \l__datatool_tmpa_tl \tl_tail:N \l__datatool_tmpa_tl }
   }
   {
     \tl_set:Nx \l__datatool_tmpc_tl
       { \tl_head:N \l__datatool_tmpa_tl }
     \tl_set:Nx \l__datatool_tmpa_tl
       { \tl_tail:N \l__datatool_tmpa_tl }
```

```
    }
    \exp_args:No \tl_if_head_is_space:nTF { \l__datatool_tmpb_tl }
    {
      \tl_set:Nn \l__datatool_tmpd_tl { ~ }
      \tl_set:Nx \l__datatool_tmpb_tl
        { \tl_head:N \l__datatool_tmpb_tl \tl_tail:N \l__datatool_tmpb_tl }
    }
    {
      \tl_set:Nx \l__datatool_tmpd_tl
        { \tl_head:N \l__datatool_tmpb_tl }
      \tl_set:Nx \l__datatool_tmpb_tl
        { \tl_tail:N \l__datatool_tmpb_tl }
    }
    \cs_set:Nn \__datatool_next: { }
    \if_meaning:w \q_recursion_tail \l__datatool_tmpd_tl
      \cs_set_eq:NN \__datatool_next: \__datatool_starts_with_true:
    \else
      \if_meaning:w \q_recursion_tail \l__datatool_tmpc_tl
        \cs_set_eq:NN \__datatool_next: \__datatool_starts_with_false:
      \else
        \str_if_eq:NNTF \l__datatool_tmpc_tl \l__datatool_tmpd_tl
         {
          \tl_put_right:No \l__datatool_prefix_tl { \l__datatool_tmpd_tl }
         }
         {
          \cs_set_eq:NN \__datatool_next: \__datatool_starts_with_false:
         }
      \fi
    \fi
    \__datatool_next:
    \__datatool_if_starts_with:
}
\cs_new:Nn \__datatool_starts_with_false:
 {
    \cs_set:Nn \__datatool_result:nn { ##2 }
    \tl_clear:N \l__datatool_prefix_tl
    \use_none_delimit_by_q_recursion_stop:w
 }
\cs_new:Nn \__datatool_starts_with_true:
 {
    \cs_set:Nn \__datatool_result:nn { ##1 }
    \if_meaning:w \q_recursion_tail \l__datatool_tmpc_tl
      \tl_clear:N \l__datatool_suffix_tl
    \else
      \tl_set_eq:NN \l__datatool_suffix_tl \l__datatool_tmpc_tl
      \exp_after:wN \__datatool_suffix_tail:w \l__datatool_tmpa_tl
    \fi
    \use_none_delimit_by_q_recursion_stop:w
 }
\cs_new:Npn \__datatool_suffix_tail:w #1\q_recursion_tail
```

```
  {
    \tl_put_right:Nn \l__datatool_suffix_tl { #1 }
  }
 \cs_new:Npn \__datatool_if_starts_with:NnT #1#2#3
  {
    \__datatool_if_starts_with:NnTF #1 { #2 } { #3 } { }
  }
```

Retain original internal command for checking if an argument is numerical but updated to use new parsing.

\@dtl@checknumerical

\@dtl@checknumerical{⟨*arg*⟩}

Checks if ⟨*arg*⟩ is numerical (includes decimal numbers, but not scientific notation.) Sets \@dtl@datatype.

```
\newcommand{\@dtl@checknumerical}[1]{%
 \tl_if_single_token:nTF { #1 }
 { \exp_args:No \__datatool_parse_datum:n { #1 } }
 { \__datatool_parse_datum:n { #1 }}
}
```

\DTLifnumerical

\DTLifnumerical{⟨*arg*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Tests the first argument, if it's numerical do second argument, otherwise do third argument.

```
\newrobustcmd{\DTLifnumerical}[3]{%
 \@dtl@checknumerical { #1 }
 \datatool_if_numeric_datum_type:nTF { \@dtl@datatype }
   { #2 } { #3 }
}
```

\DTLiftemporal

\DTLiftemporal{⟨*arg*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Tests the first argument, if it's temporal do second argument, otherwise do third argument.

```
\NewDocumentCommand \DTLiftemporal { m m m }
{
 \tl_if_single_token:nTF { #1 }
 { \exp_args:No \__datatool_parse_datum:n { #1 } }
 { \__datatool_parse_datum:n { #1 }}
 \datatool_if_temporal_datum_type:nTF { \@dtl@datatype }
   { #2 } { #3 }
}
```

\dtl@testbothnumerical

\dtl@testbothnumerical{⟨*arg1*⟩}{⟨*arg2*⟩}

Tests if both arguments are numerical. This sets the conditional `\if@dtl@condition`.

```
\newcommand*{\dtl@testbothnumerical}[2]{%
  \DTLifnumerical { #1 }
  {
    \DTLifnumerical { #2 } { \@dtl@conditiontrue } { \@dtl@conditionfalse }
  }
  { \@dtl@conditionfalse }
}
```

`\DTLifreal`

`\DTLifreal{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}`

Tests the first argument, if it's a real number (not an integer or currency) do second argument, otherwise do third argument.

```
\newrobustcmd{\DTLifreal}[3]{%
 \tl_if_single_token:nTF { #1 }
 { \exp_args:No \__datatool_parse_datum:n { #1 } }
 { \__datatool_parse_datum:n { #1 }}
 \int_compare:nNnTF { \@dtl@datatype } = { \c_datatool_decimal_int }
  { #2 } { #3 }
}
```

`\DTLifint`

`\DTLifint{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}`

Tests the first argument, if it's an integer do second argument, otherwise do third argument.

```
\newrobustcmd{\DTLifint}[3]{%
 \tl_if_single_token:nTF { #1 }
 { \exp_args:No \__datatool_parse_datum:n { #1 } }
 { \__datatool_parse_datum:n { #1 }}
 \int_compare:nNnTF { \@dtl@datatype } = { \c_datatool_integer_int }
  { #2 } { #3 }
}
```

`\DTLifdatetime`

`\DTLifdatetime{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}`

Tests the first argument, if it's a timestamp (date and time) do second argument, otherwise do third argument.

```
\NewDocumentCommand \DTLiftimestamp { m m m }
{
 \tl_if_single_token:nTF { #1 }
 { \exp_args:No \__datatool_parse_datum:n { #1 } }
 { \__datatool_parse_datum:n { #1 }}
 \int_compare:nNnTF { \@dtl@datatype } = { \c_datatool_datetime_int }
  { #2 } { #3 }
}
```

\DTLifdate

\DTLifdate{⟨*arg*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Tests the first argument, if it's a date (no time) do second argument, otherwise do third argument.

```
\NewDocumentCommand \DTLifdate { m m m }
{
 \tl_if_single_token:nTF { #1 }
 { \exp_args:No \__datatool_parse_datum:n { #1 } }
 { \__datatool_parse_datum:n { #1 }}
 \int_compare:nNnTF { \@dtl@datatype } = { \c_datatool_date_int }
  { #2 } { #3 }
}
```

\DTLiftime

\DTLiftime{⟨*arg*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Tests the first argument, if it's a time (no date) do second argument, otherwise do third argument.

```
\NewDocumentCommand \DTLiftime { m m m }
{
 \tl_if_single_token:nTF { #1 }
  { \exp_args:No \__datatool_parse_datum:n { #1 } }
  { \__datatool_parse_datum:n { #1 }}
 \int_compare:nNnTF { \@dtl@datatype } = { \c_datatool_time_int }
  { #2 } { #3 }
}
```

\DTLifstring

\DTLifstring{⟨*arg*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Tests the first argument, if it's a string do second argument, otherwise do third argument.

```
\newrobustcmd{\DTLifstring}[3]{%
 \tl_if_single_token:nTF { #1 }
 { \exp_args:No \__datatool_parse_datum:n { #1 } }
 { \__datatool_parse_datum:n { #1 }}
 \int_compare:nNnTF { \@dtl@datatype } = { \c_datatool_string_int }
  { #2 } { #3 }
}
```

\DTLifcurrency

\DTLifcurrency{⟨*arg*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Tests the first argument, if it's currency do second argument, otherwise do third argument.

```
\newrobustcmd{\DTLifcurrency}[3]{%
```

```
  \tl_if_single_token:nTF { #1 }
  { \exp_args:No \__datatool_parse_datum:n { #1 } }
  { \__datatool_parse_datum:n { #1 }}
  \int_compare:nNnTF { \@dtl@datatype } = { \c_datatool_currency_int }
   { #2 } { #3 }
}
```

\DTLifcurrencyunit

\DTLifcurrencyunit{⟨*arg*⟩}{⟨*symbol*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

This tests if ⟨*arg*⟩ is currency, and uses the currency unit ⟨*symbol*⟩. If true do third argument, otherwise do fourth argument.

```
\newrobustcmd*{\DTLifcurrencyunit}[4]{%
 \tl_if_single_token:nTF { #1 }
 { \exp_args:No \__datatool_parse_datum:n { #1 } }
 { \__datatool_parse_datum:n { #1 }}
 \ifnum\@dtl@datatype=\c_datatool_currency_int
   \tl_if_eq:NnTF \l__datatool_datum_currency_tl { #2 }
   { #3 }
   {
     \tl_if_eq:NnTF \l__datatool_datum_currency_tl { \@dtl@currency }
     {
       \tl_if_eq:NnTF \@dtl@currency { #2 }
       { #3 }
       { #4 }
     }
     { #4 }
   }
 \else
  #4
 \fi
}
```

\DTLifcasedatatype

\DTLifcasedatatype{⟨*arg*⟩}{⟨*string case*⟩}{⟨*int case*⟩}{⟨*real case*⟩}{⟨*currency case*⟩}

If ⟨*arg*⟩ is a string, do ⟨*string case*⟩, if ⟨*arg*⟩ is an integer do ⟨*int case*⟩, if ⟨*arg*⟩ is a real number, do ⟨*real case*⟩, if ⟨*arg*⟩ is currency, do ⟨*currency case*⟩. Does nothing with other cases (unknown, or any data types introduced in new versions, such as date/time).

Deprecated as it doesn't allow for new types. Note that it's more efficient to convert to a datum variable first and use `\int_case:nn` on the datum type (which can be obtained with `\DTLdatumtype`).

```
\newrobustcmd{\DTLifcasedatatype}[5]{%
  \tl_if_single_token:nTF { #1 }
  { \exp_args:No \__datatool_parse_datum:n { #1 } }
  { \__datatool_parse_datum:n { #1 }}
```

```
  \ifcase\@dtl@datatype
   #2% string
  \or
   #3% integer
  \or
   #4% number
  \or
   #5% currency
  \fi
}
```

2.7.2 Locale Numerical Comparisons

Parse two numerical values for comparison. Any non-numeric value will be treated as zero.

```
\cs_new:Nn \__datatool_parse_numbers_ii:nnNN
{
  \tl_if_single_token:nTF { #1 }
  { \exp_args:No \__datatool_parse_datum:n { #1 } }
  { \__datatool_parse_datum:n { #1 }}
  \ifnum\@dtl@datatype > \c_datatool_string_int
    \tl_set_eq:NN #3 \l__datatool_datum_value_tl
  \else
    \tl_set:Nn #3 { 0 }
  \fi
  \int_set_eq:NN \l__datatool_tmp_datatype_int \@dtl@datatype
  \tl_if_single_token:nTF { #2 }
  { \exp_args:No \__datatool_parse_datum:n { #2 } }
  { \__datatool_parse_datum:n { #2 }}
  \ifnum\@dtl@datatype > \c_datatool_string_int
    \tl_set_eq:NN #4 \l__datatool_datum_value_tl
  \else
    \tl_set:Nn #4 { 0 }
  \fi
```

Ensure that the dominant data type can be picked up afterwards.

```
  \int_compare:nNnT
   { \l__datatool_tmp_datatype_int } > { \@dtl@datatype }
  {
    \int_set_eq:NN \@dtl@datatype \l__datatool_tmp_datatype_int
  }
}
```

Parse three numerical values.

```
\cs_new:Nn \__datatool_parse_numbers_iii:nnnNNN
{
  \__datatool_parse_numbers_ii:nnNN { #1 } { #2 } #4 #5
  \tl_if_single_token:nTF { #3 }
  { \exp_args:No \__datatool_parse_datum:n { #3 } }
  { \__datatool_parse_datum:n { #3 }}
```

```
    \ifnum\@dtl@datatype > \c_datatool_string_int
      \tl_set_eq:NN #6 \l__datatool_datum_value_tl
    \else
      \tl_set:Nn #6 { 0 }
    \fi
  }
```

\DTLifnumlt

\DTLifnumlt{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if {⟨*num1*⟩} < {⟨*num2*⟩}. The numeric values need to be obtained to ensure that they work with \dtlifnumlt.

```
\newrobustcmd*{\DTLifnumlt}[4]{%
  \__datatool_parse_numbers_ii:nnNN { #1 } { #2 } \@dtl@numi \@dtl@numii
  \dtlifnumlt{\@dtl@numi}{\@dtl@numii} { #3 } { #4 }
}
```

\DTLifnumgt

\DTLifnumgt{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if {⟨*num1*⟩} > {⟨*num2*⟩}. The numeric values need to be obtained to ensure that they work with \dtlifnumgt.

```
\newrobustcmd*{\DTLifnumgt}[4]{%
  \__datatool_parse_numbers_ii:nnNN { #1 } { #2 } \@dtl@numi \@dtl@numii
  \dtlifnumgt{\@dtl@numi}{\@dtl@numii} { #3 } { #4 }
}
```

\DTLifnumeq

\DTLifnumeq{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if {⟨*num1*⟩} = {⟨*num2*⟩}. The numeric values need to be obtained to ensure that they work with \dtlifnumeq.

```
\newrobustcmd*{\DTLifnumeq}[4]{%
  \__datatool_parse_numbers_ii:nnNN { #1 } { #2 } \@dtl@numi \@dtl@numii
  \dtlifnumeq{\@dtl@numi}{\@dtl@numii} { #3 } { #4 }
}
```

\DTLifnumclosedbetween

\DTLifnumclosedbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩.

```
\newrobustcmd*{\DTLifnumclosedbetween}[5]{%
 \__datatool_parse_numbers_iii:nnnNNN { #1 } { #2 } { #3 }
  \@dtl@numi \@dtl@numii \@dtl@numiii
 \DTLifFPclosedbetween{\@dtl@numi}{\@dtl@numii}{\@dtl@numiii}{#4}{#5}%
}
```

\DTLifnumopenbetween

\DTLifnumopenbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩.

```
\newrobustcmd*{\DTLifnumopenbetween}[5]{%
 \__datatool_parse_numbers_iii:nnnNNN { #1 } { #2 } { #3 }
  \@dtl@numi \@dtl@numii \@dtl@numiii
 \DTLifFPopenbetween{\@dtl@numi}{\@dtl@numii}{\@dtl@numiii}{#4}{#5}%
}
```

\DTLnumcompare

\DTLnumcompare⟨*int var*⟩{⟨*num1*⟩}{⟨*num2*⟩}

```
\NewDocumentCommand \DTLnumcompare { m m m }
{
  \__datatool_parse_numbers_ii:nnNN { #2 } { #3 } \@dtl@numi \@dtl@numii
  \int_compare:nNnTF
    { \@dtl@datatype } < { \c_datatool_decimal_int }
  {
    \int_compare:nNnTF { \@dtl@numi } < { \@dtl@numii }
     { \int_set:Nn #1 { -1 } }
     {
       \int_compare:nNnTF { \@dtl@numi } > { \@dtl@numii }
       { \int_set_eq:NN #1 \c_one_int }
       { \int_zero:N #1 }
     }
  }
  {
    \fp_compare:nNnTF { \@dtl@numi } < { \@dtl@numii }
     { \int_set:Nn #1 { -1 } }
     {
       \fp_compare:nNnTF { \@dtl@numi } > { \@dtl@numii }
       { \int_set_eq:NN #1 \c_one_int }
       { \int_zero:N #1 }
     }
  }
}
```

2.7.3 String Comparisons

\dtlcompare

\dtlcompare{⟨*count*⟩}{⟨*string1*⟩}{⟨*string2*⟩}

Compares ⟨*string1*⟩ and ⟨*string2*⟩, and stores the result in the count register ⟨*count*⟩. The result may be one of:

-1 if ⟨*string1*⟩ is considered to be less than ⟨*string2*⟩
0 if ⟨*string1*⟩ is considered to be the same as ⟨*string2*⟩
1 if ⟨*string1*⟩ is considered to be greater than ⟨*string2*⟩

Note that for the purposes of string comparisons, commands within ⟨*string1*⟩ and ⟨*string2*⟩ are ignored, except for `\space` and `~`, which are both treated as a space (character code 32.) The following examples assume that the count register `\mycount` has been defined as follows:

```
%\newcount\mycount
%
```

Examples:

1. ```
   %\dtlcompare{\mycount}{Z\"oe}{Zoe}\number\mycount
   %
   ```

   produces: -1, since the accent command is ignored.

2. ```
   %\dtlcompare{\mycount}{foo}{Foo}\number\mycount
   %
   ```

   produces: 1, since the comparison is case sensitive, however, note the following example:

3. ```
   %\dtlcompare{\mycount}{foo}{\uppercase{f}oo}\number\mycount
   %
   ```

   which produces: 1, since the `\uppercase` command is ignored.

4. A control sequence is treated as having the character code value of 0. Pre version 2.32, `\dtlcompare` was advertised here as skipping control sequences when actually it was treating a control sequence as character 0. To avoid breaking backward-compatibility where a control sequence is expected to have this behaviour, we now have a switch to determine whether to treat control sequences as 0 or to skip them.

   So you can now "trick" `\dtlcompare` using a command which doesn't output any text if switch this conditional on. Suppose you have defined the following command:

   ```
   %\newcommand*{\noopsort}[1]{}
   %
   ```

   then `\noopsort{a}foo` produces the text: foo, however the following

   ```
   %\dtlcompare{\mycount}{\noopsort{a}foo}{bar}\number\mycount
   %
   ```

   produces: -1, since the command `\noopsort` is disregarded when the comparison is made, so `\dtlcompare` just compares `{a}foo` with `bar`, and since `a` is less than `b`, the first string is considered to be less than the second string.
   ```

5. Note that this also means that:

```
%\def\mystr{abc}%
%\dtlcompare{\mycount}{\mystr}{abc}\number\mycount
%
```

produces: -1, since the command `\mystr` is disregarded, which means that `\dtlcompare` is comparing an empty string with the string `abc`.

6. Spaces count in the comparison:

```
%\dtlcompare{\mycount}{ab cd}{abcd}\number\mycount
%
```

produces: -1, but sequential spaces are treated as a single space:

```
%\dtlcompare{\mycount}{ab  cd}{ab cd}\number\mycount
%
```

produces: 0.

7. As usual, spaces following command names are ignored, so

```
%\dtlcompare{\mycount}{ab\relax cd}{ab cd}\number\mycount
%
```

produces: -1.

8. ~ and `\space` are considered to be the same as a space:

```
%\dtlcompare{\mycount}{ab cd}{ab~cd}\number\mycount
%
```

produces: 0.

To ensure backward-compatibility, this still needs to strip commands so that the `\noop` example can continue to work, but it's better to prepare the sort values first for sorting lists.

```
\newcommand*{\dtlcompare}[3]{%
  \__datatool_get_compare_sort:Nn \l__datatool_tmpa_str { #2 }
  \__datatool_get_compare_sort:Nn \l__datatool_tmpb_str { #3 }
  \__datatool_strcmp:NNN #1 \l__datatool_tmpa_str \l__datatool_tmpb_str
}
```

`\dtlicompare`

```
\dtlicompare{⟨count⟩}{⟨string1⟩}{⟨string2⟩}
```

As `\dtlcompare` but ignores case.

```
\newcommand*{\dtlicompare}[3]{%
  \__datatool_get_icompare_sort:Nn \l__datatool_tmpa_str { #2 }
  \__datatool_get_icompare_sort:Nn \l__datatool_tmpb_str { #3 }
  \__datatool_strcmp:NNN #1 \l__datatool_tmpa_str \l__datatool_tmpb_str
}
```

Compare two strings provided as token lists and store the result in the count register provided in the first argument.

```
\cs_new:Nn \__datatool_strcmp:NNN
{
  \exp_args:NNoo \__datatool_strcmp:Nnn #1 { #2 } { #3 }
}
```

If `\pdfstrcmp` or `\strcmp` is defined, use that for string comparisons otherwise use LaTeX3 string comparison commands.

```
\cs_if_exist:NTF \strcmp
{
  \cs_new:Nn \__datatool_strcmp:Nnn
   {
     \int_set:Nn #1 { \strcmp { #2 } { #3 } }
   }
  \cs_new:Nn  \datatool_strcmp:nn
   {
     \strcmp { #1 } { #2 }
   }
}
{
  \cs_if_exist:NTF \pdfstrcmp
   {
     \cs_new:Nn \__datatool_strcmp:Nnn
      {
        \int_set:Nn #1 { \pdfstrcmp{ #2 } { #3 } }
      }
     \cs_new:Nn  \datatool_strcmp:nn
      {
        \pdfstrcmp { #1 } { #2 }
      }
   }
   {
```

Most likely LuaTeX.

```
     \cs_new:Nn \__datatool_strcmp:Nnn
      {
        \str_if_eq:nnTF { #2 } { #3 }
        { \int_zero:N #1 }
        {
          \str_compare:nNnTF { #2 } < { #3 }
          { \int_set:Nn #1 { -1 } }
          { \int_set_eq:NN #1 \c_one_int }
```

```
        }
      }
     \cs_new:Nn  \datatool_strcmp:nn
      {
        \str_if_eq:nnTF { #1 } { #2 }
        { \c_zero_int }
        {
          \str_compare:nNnTF { #1 } < { #2 }
           { -1 }
           { \c_one_int }
        }
      }
  }
}
```

Compare two numeric values and store the result in the count register provided in the first argument.

```
\cs_new:Nn \__datatool_numcmp:Nnn
{
 \dtlifnumlt { #2 } { #3 }
  { #1=-1\relax }
  {
    \dtlifnumgt { #2 } { #3} { #1=1\relax } { #1=0\relax }
  }
}
```

```
\tl_new:N \l__datatool_cmpa_tl
\tl_new:N \l__datatool_cmpb_tl
```

The token list mapping function skips spaces, so need to replace spaces with a token that represents a space.

```
\tl_const:Nn \c__datatool_space_tl { ~ }
```

Case-sensitive:

```
\cs_new:Nn \__datatool_get_compare_sort:Nn
 {
   \bool_if:NTF \l__datatool_compare_expand_cs_bool
   {
     \tl_set:Nx \l__datatool_cmpb_tl { \text_purify:n { #2 } }
   }
   {
     \tl_set:Nn \l__datatool_cmpb_tl { #2 }
   }
   \tl_replace_all:Nnn \l__datatool_cmpb_tl { ~ } { \c__datatool_space_tl }
   \tl_clear:N \l__datatool_cmpa_tl
   \exp_args:No \tl_map_function:nN
     { \l__datatool_cmpb_tl }
     \__datatool_get_compare_sort_fn:n
   \bool_if:NTF \l__datatool_compare_expand_cs_bool
   {
     \str_set:Nx #1 { \l__datatool_cmpa_tl }
```

```
    }
    {
      \str_set:Nx #1 { \text_purify:n { \l__datatool_cmpa_tl } }
    }
  }
```

Case-insensitive:

```
 \cs_new:Nn \__datatool_get_icompare_sort:Nn
 {
   \bool_if:NTF \l__datatool_compare_expand_cs_bool
   {
     \tl_set:Nx \l__datatool_cmpb_tl { \text_purify:n { #2 } }
   }
   {
     \tl_set:Nn \l__datatool_cmpb_tl { #2 }
   }
   \tl_replace_all:Nnn \l__datatool_cmpb_tl { ~ } { \c__datatool_space_tl }
   \tl_clear:N \l__datatool_cmpa_tl
   \exp_args:No \tl_map_function:nN
     { \l__datatool_cmpb_tl }
     \__datatool_get_compare_sort_fn:n
   \bool_if:NTF \l__datatool_compare_expand_cs_bool
   {
     \str_set:Nx #1 { \text_lowercase:n { \l__datatool_cmpa_tl } }
   }
   {
     \str_set:Nx #1
      { \text_purify:n { \text_lowercase:n { \l__datatool_cmpa_tl } } }
   }
 }
\cs_new:Nn \__datatool_get_compare_sort_fn:n
 {
   \tl_if_head_eq_catcode:nNTF { #1 } \relax
   {
     \tl_if_eq:nnTF { \c__datatool_space_tl } { #1 }
      {
        \tl_put_right:Nn \l__datatool_cmpa_tl { ~ }
      }
      {
        \ifdtlcompareskipcs
        \else
          \tl_put_right:Nn \l__datatool_cmpa_tl { ^^J }
        \fi
      }
   }
   { \tl_put_right:Nn \l__datatool_cmpa_tl { #1 } }
 }
```

`\dtlwordindexcompare` Word breaks come before all other letters of the alphabet (since the space character comes before all visible characters). Note that, as from v3.0, `\@dtl@wordbreak` is

no longer used, except to provide a non-empty fourth argument for \@dtldictcompare.

```
\newcommand*{\dtlwordindexcompare}[3]{%
  \@dtldictcompare{#1}{#2}{#3}{\@dtl@wordbreak}%
}
```

\dtlletterindexcompare Word breaks are ignored.

```
\newcommand*{\dtlletterindexcompare}[3]{%
  \@dtldictcompare{#1}{#2}{#3}{}%
}
```

\@dtldictcompare Word or letter compare. Fourth argument should be empty for letter compare.

```
\newcommand*{\@dtldictcompare}[4]{%
  \group_begin:
  \dtl@SortWordCommands@hook
  \exp_args:Nxx \__datatool_set_tmp_dictcomp:nn { #2 } { #3 }
  \tl_if_empty:nT { #4 }
  {
    \tl_replace_all:Nnn \l__datatool_dictcompa_tl { ~ } {}
    \tl_replace_all:Nnn \l__datatool_dictcompb_tl { ~ } {}
  }
  \str_set:NV \l__datatool_tmpa_str \l__datatool_dictcompa_tl
  \str_set:NV \l__datatool_tmpb_str \l__datatool_dictcompb_tl
  \__datatool_strcmp:NNN #1 \l__datatool_tmpa_str \l__datatool_tmpb_str
}
\tl_new:N \l__datatool_dictcompa_tl
\tl_new:N \l__datatool_dictcompb_tl
\cs_new:Nn \__datatool_set_tmp_dictcomp:nn
{
  \group_end:
  \DTLsortwordhandler { #1 } { \l__datatool_dictcompa_tl }
  \DTLsortwordhandler { #2 } { \l__datatool_dictcompb_tl }
}
```

Need to indicate type of inversion.

\datatoolpersoncomma

```
\newcommand*{\datatoolpersoncomma}{,\space}
```

\datatoolplacecomma

```
\newcommand*{\datatoolplacecomma}{,\space}
```

\datatoolsubjectcomma

```
\newcommand*{\datatoolsubjectcomma}{,\space}
```

\datatoolparenstart

```
\newcommand*{\datatoolparenstart}{\space}
```

\datatoolparen

```
\newcommand*{\datatoolparen}[1]{\space (#1)}
```

The following commands `\dtlicomparewords` and `\dtlcomparewords` were never documented and don't seem to do anything different from `\dtlicompare` and `\dtlcompare`. They have existed since at least v2.25.

`\dtlicomparewords`

```
\dtlicomparewords{⟨count⟩}{⟨word A⟩}{⟨word B⟩}
```

This does a case insensitive comparison.

```
\newcommand{\dtlicomparewords}[3]{%
  \dtlicompare{#1}{#2}{#3}%
}
```

`\dtlcomparewords`

```
\dtlcomparewords{⟨count⟩}{⟨word A⟩}{⟨word B⟩}
```

This does a case sensitive comparison.

```
\newcommand{\dtlcomparewords}[3]{%
  \dtlcompare{#1}{#2}{#3}%
}
```

macro

```
\dtlifcasechargroup{⟨char⟩}{⟨case letter⟩}{⟨case
digit⟩}{⟨case symbol⟩}
```

```
\newrobustcmd*{\dtlifcasechargroup}[4]{%
  \regex_match:nnTF { \A \d \Z} { #1 }
  { #3 }
  {
    \regex_match:nnTF { \A [[:alpha:]] \Z} { #1 }
    { #2 }
    { #3 }
  }
}
```

`\dtlparsewords`

```
\dtlparsewords{⟨phrase⟩}{⟨handler cs⟩}
```

Iterates through the given phrase. Hyphens are considered word boundaries. Punctuation and digits before and after words are discarded.

```
\newcommand*{\dtlparsewords}[2]{%
  \group_begin:
   \tl_set:Nx \l__datatool_tmpa_tl { \tl_trim_spaces:n { #1 } }
   \regex_replace_case_all:nN
    {
      \l_datatool_word_head_regex
        {
```

```
          \2 \cM\|
        }
      \l_datatool_word_hyphen_regex
        {
          \2 \cM\|
        }
      \l_datatool_word_tail_regex
        {
          \2
        }
      \l_datatool_symbols_regex
        { }
    }
    \l__datatool_tmpa_tl
    \forlistloop { #2 } { \l__datatool_tmpa_tl }
  \group_end:
}
```

\DTLifstringlt

\DTLifstringlt{⟨*string1*⟩}{⟨*string2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

String comparison (Starred version ignores case)

```
\NewDocumentCommand \DTLifstringlt { s m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifstringlt { #2 } { #3 } { #4 } { #5 }
   }
   {
     \@DTLifstringlt { #2 } { #3 } { #4 } { #5 }
   }
}
```

The internals are used by `\DTLiflt` as well.

\@DTLifstringlt Unstarred version (case-sensitive).

```
\newcommand*{\@DTLifstringlt}[4]{
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     <
    { \l__datatool_tmpb_str }
   { #3 } { #4 }
}
```

`\s@DTLifstringlt` Starred version (case-sensitive).

```
\newcommand*{\@sDTLifstringlt}[4]{
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     <
    { \l__datatool_tmpb_str }
   { #3 } { #4 }
}
```

`\DTLiflt` `\DTLiflt{⟨arg1⟩}{⟨arg2⟩}{⟨true part⟩}{⟨false part⟩}`

Does `\DTLifnumlt` if both ⟨*arg1*⟩ and ⟨*arg2*⟩ are numerical, otherwise do `\DTLifstringlt` (unstarred version) or `\DTLifstringlt*` (starred version).

```
\NewDocumentCommand \DTLiflt { s m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLiflt { #2 } { #3 } { #4 } { #5 }
   }
   {
     \@DTLiflt { #2 } { #3 } { #4 } { #5 }
   }
}
```

`\@DTLiflt` Unstarred version (also used in `\DTLislt`).

```
\newcommand*{\@DTLiflt}[4]{
  \dtl@testbothnumerical {#1} {#2}
  \if@dtl@condition
   \DTLifnumlt {#1} {#2} {#3} {#4}
  \else
   \@DTLifstringlt {#1} {#2} {#3} {#4}
  \fi
}
```

`\@sDTLiflt` Starred version (also used in `\DTLisilt`).

```
\newcommand*{\@sDTLiflt}[4]{
  \dtl@testbothnumerical {#1} {#2}
  \if@dtl@condition
   \DTLifnumlt {#1} {#2} {#3} {#4}
  \else
   \@sDTLifstringlt {#1} {#2} {#3} {#4}
  \fi
}
```

\DTLifstringgt

```
\DTLifstringgt{⟨string1⟩}{⟨string2⟩}{⟨true part⟩}
{⟨false part⟩}
```

String comparison (starred version ignores case)

```
\NewDocumentCommand \DTLifstringgt { s m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifstringgt { #2 } { #3 } { #4 } { #5 }
   }
   {
     \@DTLifstringgt { #2 } { #3 } { #4 } { #5 }
   }
}
```

The internals are used by `\DTLifgt` as well.

\@DTLifstringgt Unstarred version (case-sensitive).

```
\newcommand*{\@DTLifstringgt}[4]{
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     >
    { \l__datatool_tmpb_str }
   { #3 } { #4 }
}
```

\@sDTLifstringgt Starred version (case-insensitive).

```
\newcommand*{\@sDTLifstringgt}[4]{
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     >
    { \l__datatool_tmpb_str }
   { #3 } { #4 }
}
```

\DTLifgt

```
\DTLifgt{⟨arg1⟩}{⟨arg2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does `\DTLifnumgt` if both ⟨*arg1*⟩ and ⟨*arg2*⟩ are numerical, otherwise do `\DTLifstringgt` or `\DTLifstringgt*`.

```
\NewDocumentCommand \DTLifgt { s m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifgt { #2 } { #3 } { #4 } { #5 }
   }
   {
     \@DTLifgt { #2 } { #3 } { #4 } { #5 }
   }
}
```

`\@DTLifgt` Unstarred version (also used in `\DTLislt`).

```
\newcommand*{\@DTLifgt}[4]{
  \dtl@testbothnumerical {#1} {#2}
  \if@dtl@condition
   \DTLifnumgt {#1} {#2} {#3} {#4}
  \else
   \@DTLifstringgt {#1} {#2} {#3} {#4}
  \fi
}
```

`\@sDTLifgt` Starred version (also used in `\DTLisigt`).

```
\newcommand*{\@sDTLifgt}[4]{
  \dtl@testbothnumerical {#1} {#2}
  \if@dtl@condition
   \DTLifnumgt {#1} {#2} {#3} {#4}
  \else
   \@sDTLifstringgt {#1} {#2} {#3} {#4}
  \fi
}
```

`\DTLifstringeq`

```
\DTLifstringeq{⟨string1⟩}{⟨string2⟩}{⟨true part⟩}
{⟨false part⟩}
```

String comparison (starred version ignores case)

```
\NewDocumentCommand \DTLifstringeq { s m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifstringeq { #2 } { #3 } { #4 } { #5 }
   }
   {
     \@DTLifstringeq { #2 } { #3 } { #4 } { #5 }
   }
}
```

The internals are used by `\DTLifeq` as well.

`\@DTLifstringeq` Unstarred version (case-sensitive).

```
\newcommand*{\@DTLifstringeq}[4]{
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     =
    { \l__datatool_tmpb_str }
   { #3 } { #4 }
}
```

`\@sDTLifstringeq` Starred version (case-insensitive).

```
\newcommand*{\@sDTLifstringeq}[4]{
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     =
    { \l__datatool_tmpb_str }
   { #3 } { #4 }
}
```

`\DTLifeq` `\DTLifeq{`⟨*arg1*⟩`}{`⟨*arg2*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Does `\DTLifnumeq` if both ⟨*arg1*⟩ and ⟨*arg2*⟩ are numerical, otherwise do `\DTLifstringeq` or `\DTLifstringeq*`.

```
\NewDocumentCommand \DTLifeq { s m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifeq { #2 } { #3 } { #4 } { #5 }
   }
   {
     \@DTLifeq { #2 } { #3 } { #4 } { #5 }
   }
}
```

`\@DTLifeq` Unstarred version (also used in `\DTLiseq`).

```
\newcommand*{\@DTLifeq}[4]{
  \dtl@testbothnumerical {#1} {#2}
  \if@dtl@condition
   \DTLifnumeq {#1} {#2} {#3} {#4}
  \else
```

```
   \@DTLifstringeq {#1} {#2} {#3} {#4}
  \fi
}
```

`\@sDTLifeq` Starred version (also used in `\DTLisieq`).

```
\newcommand*{\@sDTLifeq}[4]{
  \dtl@testbothnumerical {#1} {#2}
  \if@dtl@condition
   \DTLifnumeq {#1} {#2} {#3} {#4}
  \else
   \@sDTLifstringeq {#1} {#2} {#3} {#4}
  \fi
}

\regex_new:N \l_datatool_space_regex
\regex_set:Nn \l_datatool_space_regex
 {
    ((?:[[:space:]\~])|(?:\c{protect}?\c{(?:nobreak)?space\ ?}))
 }
```

`\DTLifSubString` `\DTLifSubString{⟨string⟩}{⟨sub string⟩}{⟨true part⟩}{⟨false part⟩}`

If ⟨*sub string*⟩ is contained in ⟨*string*⟩ does ⟨*true part*⟩, otherwise does ⟨*false part*⟩. Spaces are replaced with ~ to prevent leading/trailing spaces from being trimmed.

```
\newrobustcmd*{\DTLifSubString}{\@ifstar\@sDTLifSubString\@DTLifSubString}
```

`\@DTLifSubString`

```
\newcommand*{\@DTLifSubString}[4]{%
  \tl_set:Nx \l__datatool_tmpa_tl { \text_purify:n { #1 } }
  \tl_set:Nx \l__datatool_tmpb_tl { \text_purify:n { #2 } }
  \regex_replace_all:NnN \l_datatool_space_regex { \~ } \l__datatool_tmpa_tl
  \regex_replace_all:NnN \l_datatool_space_regex { \~ } \l__datatool_tmpb_tl
  \exp_args:NVV \str_if_in:nnTF
   \l__datatool_tmpa_tl \l__datatool_tmpb_tl
   { #3 } { #4 }
}
```

`\@sDTLifSubString`

```
\newcommand*{\@sDTLifSubString}[2]{%
  \@DTLifSubString{ \text_lowercase:n { #1 } }{ \text_lowercase:n { #2 } }%
}
```

`\DTLifStartsWith` `\DTLifStartsWith{⟨string⟩}{⟨substring⟩}{⟨true part⟩}{⟨false part⟩}`

If ⟨*string*⟩ starts with ⟨*substring*⟩, this does ⟨*true part*⟩, otherwise it does ⟨*false part*⟩.

```
\newrobustcmd*{\DTLifStartsWith}{\@ifstar\@sDTLifStartsWith\@DTLifStartsWith}
```

\@DTLifStartsWith

```
\newcommand*{\@DTLifStartsWith}[4]{%
  \tl_set:Nx \l__datatool_tmpa_tl { \text_purify:n { #1 } }
  \tl_set:Nx \l__datatool_tmpb_tl { \text_purify:n { #2 } }
  \regex_replace_all:NnN \l_datatool_space_regex { \~ } \l__datatool_tmpa_tl
  \regex_replace_all:NnN \l_datatool_space_regex { \~ } \l__datatool_tmpb_tl
  \exp_args:NVV \__datatool_if_starts_with:nnTF
    \l__datatool_tmpa_tl \l__datatool_tmpb_tl
   { #3 } { #4 }
}
```

\@sDTLifStartsWith

```
\newcommand*{\@sDTLifStartsWith}[2]{%
  \@DTLifStartsWith{ \text_lowercase:n { #1 } }{ \text_lowercase:n { #2 } }%
}
```

\DTLifEndsWith{⟨*string*⟩}{⟨*substring*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

\DTLifEndsWith

If ⟨*string*⟩ ends with ⟨*substring*⟩, this does ⟨*true part*⟩, otherwise it does ⟨*false part*⟩.

```
\newrobustcmd*{\DTLifEndsWith}{\@ifstar\@sDTLifEndsWith\@DTLifEndsWith}
```

\@DTLifEndsWith

```
\newcommand*{\@DTLifEndsWith}[4]{%
  \tl_set:Nx \l__datatool_tmpa_tl { \text_purify:n { #1 } }
  \tl_set:Nx \l__datatool_tmpb_tl { \text_purify:n { #2 } }
  \regex_replace_all:NnN \l_datatool_space_regex { \~ } \l__datatool_tmpa_tl
  \regex_replace_all:NnN \l_datatool_space_regex { \~ } \l__datatool_tmpb_tl
  \tl_reverse:N \l__datatool_tmpa_tl
  \tl_reverse:N \l__datatool_tmpb_tl
  \exp_args:NVV \__datatool_if_starts_with:nnTF
   \l__datatool_tmpa_tl \l__datatool_tmpb_tl
   { #3 } { #4 }
}
```

\@sDTLifEndsWith

```
\newcommand*{\@sDTLifEndsWith}[2]{%
  \@DTLifEndsWith{ \text_lowercase:n { #1 } }{ \text_lowercase:n { #2 } }%
}
```

\DTLifstringclosedbetween{⟨*string*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

\DTLifstringclosedbetween

String comparison (starred version ignores case)

```
\NewDocumentCommand \DTLifstringclosedbetween { s m m m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifstringclosedbetween { #2 } { #3 } { #4 } { #5 } { #6 }
   }
   {
     \@DTLifstringclosedbetween { #2 } { #3 } { #4 } { #5 } { #6 }
   }
}
```

`DTLifstringclosedbetween` Unstarred version (case-sensitive).

```
\newcommand*{\@DTLifstringclosedbetween}[5]{%
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpc_str { #3 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     <
    { \l__datatool_tmpb_str }
   { #5 }
   {
    \str_compare:eNeTF
      { \l__datatool_tmpa_str }
       >
      { \l__datatool_tmpc_str }
     { #5 } { #4 }
   }
}
```

`DTLifstringclosedbetween` Starred version (case-sensitive).

```
\newcommand*{\@sDTLifstringclosedbetween}[5]{%
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpc_str { #3 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     <
    { \l__datatool_tmpb_str }
   { #5 }
   {
    \str_compare:eNeTF
```

```
      { \l__datatool_tmpa_str }
       >
      { \l__datatool_tmpc_str }
     { #5 } { #4 }
   }
}
```

\DTLifclosedbetween

\DTLifclosedbetween{⟨*arg*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does \DTLifnumclosedbetween if {⟨*arg*⟩}, ⟨*min*⟩ and ⟨*max*⟩ are numerical, otherwise do \DTLifstringclosedbetween or \DTLifstringclosedbetween*.

```
\NewDocumentCommand \DTLifclosedbetween { s m m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifclosedbetween { #2 } { #3 } { #4 } { #5 } { #6 }
   }
   {
     \@DTLifclosedbetween { #2 } { #3 } { #4 } { #5 } { #6 }
   }
}
```

\@DTLifclosedbetween Unstarred version

```
\newcommand*{\@DTLifclosedbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \DTLifnumerical { #1 }
     {
      \DTLifnumclosedbetween {#1} {#2} {#3} {#4} {#5}
     }
     {
      \@DTLifstringclosedbetween {#1} {#2} {#3} {#4} {#5}
     }
  \else
    \@DTLifstringclosedbetween {#1} {#2} {#3} {#4} {#5}
  \fi
}
```

\@sDTLifclosedbetween Starred version

```
\newcommand*{\@sDTLifclosedbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \DTLifnumerical { #1 }
     {
      \DTLifnumclosedbetween {#1} {#2} {#3} {#4} {#5}
     }
```

```
      {
       \@sDTLifstringclosedbetween {#1} {#2} {#3} {#4} {#5}
      }
    \else
      \@sDTLifstringclosedbetween {#1} {#2} {#3} {#4} {#5}
    \fi
  }
```

\DTLifstringopenbetween

\DTLifstringopenbetween{⟨*string*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

String comparison (starred version ignores case)

```
\NewDocumentCommand \DTLifstringopenbetween { s m m m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifstringopenbetween { #2 } { #3 } { #4 } { #5 } { #6 }
   }
   {
     \@DTLifstringopenbetween { #2 } { #3 } { #4 } { #5 } { #6 }
   }
}
```

Unstarred version:

```
\newcommand*{\@DTLifstringopenbetween}[5]{%
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \exp_args:NNo \__datatool_get_compare_sort:Nn
    \l__datatool_tmpc_str { #3 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     >
    { \l__datatool_tmpb_str }
   {
    \str_compare:eNeTF
      { \l__datatool_tmpa_str }
       <
      { \l__datatool_tmpc_str }
      { #4 } { #5 }
   }
   { #5 }
}
```

\@sDTLifstringopenbetween

Starred version (case-sensitive).

```
\newcommand*{\@sDTLifstringopenbetween}[5]{%
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
```

```
    \l__datatool_tmpa_str { #1 }
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpb_str { #2 }
  \exp_args:NNo \__datatool_get_icompare_sort:Nn
    \l__datatool_tmpc_str { #3 }
  \str_compare:eNeTF
    { \l__datatool_tmpa_str }
     >
    { \l__datatool_tmpb_str }
   {
    \str_compare:eNeTF
      { \l__datatool_tmpa_str }
       <
      { \l__datatool_tmpc_str }
      { #4 } { #5 }
   }
   { #5 }
}
```

\DTLifopenbetween

\DTLifopenbetween{⟨*arg*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does \DTLifnumopenbetween if {⟨*arg*⟩}, ⟨*min*⟩ and ⟨*max*⟩ are numerical, otherwise do \DTLifstringopenbetween or \DTLifstringopenbetween*.

```
\NewDocumentCommand \DTLifopenbetween { s m m m m m }
{
  \IfBooleanTF { #1 }
   {
     \@sDTLifopenbetween { #2 } { #3 } { #4 } { #5 } { #6 }
   }
   {
     \@DTLifopenbetween { #2 } { #3 } { #4 } { #5 } { #6 }
   }
}
```

\@DTLifopenbetween Unstarred version

```
\newcommand*{\@DTLifopenbetween}[5]{
  \dtl@testbothnumerical {#2} {#3}
  \if@dtl@condition
    \DTLifnumerical { #1 }
     {
      \DTLifnumopenbetween {#1} {#2} {#3} {#4} {#5}
     }
     {
      \@DTLifstringopenbetween {#1} {#2} {#3} {#4} {#5}
     }
  \else
    \@DTLifstringopenbetween {#1} {#2} {#3} {#4} {#5}
```

```
  \fi
}
```

\@sDTLifopenbetween Starred version

```
\newcommand*{\@sDTLifopenbetween}[5]{
  \dtl@testbothnumerical {#2} {#3}
  \if@dtl@condition
    \DTLifnumerical { #1 }
     {
      \DTLifnumopenbetween {#1} {#2} {#3} {#4} {#5}
     }
     {
      \@sDTLifstringopenbetween {#1} {#2} {#3} {#4} {#5}
     }
  \else
    \@sDTLifstringopenbetween {#1} {#2} {#3} {#4} {#5}
  \fi
}

\ExplSyntaxOff
```

\DTLifFPopenbetween

```
\DTLifFPopenbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}
{⟨false part⟩}
```

Determines if $\langle min \rangle < \langle num \rangle < \langle max \rangle$ where all arguments are in standard fixed point notation. (Command name maintained for backward compatibility.)

```
\let\DTLifFPopenbetween\dtlifnumopenbetween
```

\DTLifFPclosedbetween

```
\DTLifFPclosedbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}
{⟨false part⟩}
```

Determines if $\langle min \rangle \leq \langle num \rangle \leq \langle max \rangle$. (Command name maintained for backward compatibility.)

```
\let\DTLifFPclosedbetween\dtlifnumclosedbetween
```

2.7.4 ifthen Conditionals

The following commands provide conditionals `\DTLis`... which can be used in `\ifthenelse`.

\dtl@testlt Command to test if first argument is less than second argument. If either argument is a string, a case sensitive string comparison is used instead. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testlt}[2]{%
  \@DTLiflt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLislt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLislt}[2]{%
  \TE@throw\noexpand\dtl@testlt{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testiclt` Command to test if first argument is less than second argument. If either argument is a string, a case insensitive string comparison is used instead. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testiclt}[2]{%
  \@sDTLiflt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisilt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisilt}[2]{%
  \TE@throw\noexpand\dtl@testiclt{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testgt` Command to test if first argument is greater than second argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testgt}[2]{%
  \@DTLifgt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisgt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisgt}[2]{%
  \TE@throw\noexpand\dtl@testgt{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testicgt` Command to test if first argument is greater than second argument (ignores case). This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testicgt}[2]{%
  \@sDTLifgt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisigt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisigt}[2]{%
  \TE@throw\noexpand\dtl@testicgt{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testeq` Command to test if first argument is equal to the second argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testeq}[2]{%
  \@DTLifeq{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLiseq` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLiseq}[2]{%
  \TE@throw\noexpand\dtl@testeq{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testiceq` Command to test if first number is equal to the second number (ignores case). This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testiceq}[2]{%
  \@sDTLifeq{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisieq` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisieq}[2]{%
  \TE@throw\noexpand\dtl@testiceq{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testifsubstring` Command to test if second argument is a substring of the first argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testifsubstring}[2]{%
  \@DTLifSubString{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisSubString` Tests if second argument is contained in first argument.

```
\newcommand*{\DTLisSubString}[2]{%
  \TE@throw\noexpand\dtl@testifsubstring{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\dtl@testifisubstring` Command to test if second argument is a substring of the first argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testifisubstring}[2]{%
  \@sDTLifSubString{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisiSubString` Tests if second argument is contained in first argument (no case).

```
\newcommand*{\DTLisiSubString}[2]{%
  \TE@throw\noexpand\dtl@testifisubstring{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\dtl@teststartswith` Command to test if second argument is a prefix of the first argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@teststartswith}[2]{%
  \@DTLifStartsWith{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisPrefix` Tests if first argument starts with second argument.

```
\newcommand*{\DTLisPrefix}[2]{%
  \TE@throw\noexpand\dtl@teststartswith{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\dtl@testistartswith` Command to test if second argument is a prefix of the first argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testistartswith}[2]{%
  \@sDTLifStartsWith{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisiPrefix` Tests if first argument starts with second argument.

```
\newcommand*{\DTLisiPrefix}[2]{%
  \TE@throw\noexpand\dtl@testistartswith{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\dtl@testendswith` Command to test if second argument is a suffix of the first argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testendswith}[2]{%
  \@DTLifEndsWith{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisSuffix` Tests if first argument ends with second argument.

```
\newcommand*{\DTLisSuffix}[2]{%
  \TE@throw\noexpand\dtl@testendswith{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\dtl@testiendswith` Command to test if second argument is a suffix of the first argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testiendswith}[2]{%
  \@sDTLifEndsWith{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisiSuffix` Tests if first argument ends with second argument.

```
\newcommand*{\DTLisiSuffix}[2]{%
  \TE@throw\noexpand\dtl@testiendswith{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\DTLisinlist` Tests if first argument is an element of the comma-separated list given in the second argument.

```
\newcommand*{\DTLisinlist}[2]{%
  \TE@throw\noexpand\dtl@testinlist{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\dtl@testinlist`

```
\newcommand*{\dtl@testinlist}[2]{%
  \DTLifinlist{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`dtl@testnumclosedbetween` Command to test if first number lies between second and third numbers. (End points included, all arguments are fixed point numbers in standard format.) This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testnumclosedbetween}[3]{%
  \DTLifnumclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

Provide conditional command for use in `\ifthenelse`

`\DTLisnumclosedbetween`

```
\newcommand*{\DTLisnumclosedbetween}[3]{%
  \TE@throw\noexpand\dtl@testnumclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

`\DTLisFPclosedbetween` Keep old command name for backwards compatibility:

```
\let\DTLisFPclosedbetween\DTLisnumclosedbetween
```

`\dtl@testnumopenbetween` Command to test if first number lies between second and third numbers. (End points excluded, all arguments are fixed point numbers in standard format.) This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testnumopenbetween}[3]{%
  \DTLifnumopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisnumopenbetween` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisnumopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testnumopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

`\DTLisFPopenbetween` Keep old command name for backwards compatibility:

```
\let\DTLisFPopenbetween\DTLisnumopenbetween
```

`\dtl@testclosedbetween` Command to test if first value lies between second and third values. (End points included, case sensitive.) This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testclosedbetween}[3]{%
  \@DTLifclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisclosedbetween` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisclosedbetween}[3]{%
  \TE@throw\noexpand\dtl@testclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

\dtl@testiclosedbetween Command to test if first value lies between second and third values. (End points included, case ignored.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testiclosedbetween}[3]{%
  \@sDTLifclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

\DTLisiclosedbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisiclosedbetween}[3]{%
  \TE@throw\noexpand\dtl@testiclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

\dtl@testopenbetween Command to test if first value lies between second and third values. (End points excluded, case sensitive.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testopenbetween}[3]{%
  \@DTLifopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

\DTLisopenbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

\dtl@testiopenbetween Command to test if first value lies between second and third values. (End points excluded, case ignored.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testiopenbetween}[3]{%
  \@sDTLifopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

\DTLisiopenbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisiopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testiopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

\dtl@testFPislt Command to test if first number is less than second number where both numbers are in standard format (dot for decimal separator and no group separators). This sets \if@dtl@condition.

```
\newcommand*{\dtl@testFPislt}[2]{%
  \dtlifnumlt{#1}{#2}%
  {%
    \@dtl@conditiontrue
  }%
  {%
```

```
    \@dtl@conditionfalse
  }%
}
```

`\DTLisnumlt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisnumlt}[2]{%
  \TE@throw\noexpand\dtl@testFPislt{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\DTLisFPlt` Synonym of `\DTLisnumlt`.

```
\let\DTLisFPlt\DTLisnumlt
```

`\dtl@testFPisgt` Command to test if first number is greater than second number where both numbers are in standard format. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testFPisgt}[2]{%
  \dtlifnumgt{#1}{#2}%
  {%
    \@dtl@conditiontrue
  }%
  {%
   \@dtl@conditionfalse
  }%
}
```

`\DTLisnumgt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisnumgt}[2]{%
  \TE@throw\noexpand\dtl@testFPisgt{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

`\DTLisFPgt` Synonym

```
\let\DTLisFPgt\DTLisnumgt
```

`\dtl@testFPiseq` Command to test if two numbers are equal, where both numbers are in standard decimal format

```
\newcommand*{\dtl@testFPiseq}[2]{%
 \dtlifnumeq{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
  \@dtl@conditionfalse
 }%
}
```

`\DTLisnumeq` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisnumeq}[2]{%
  \TE@throw\noexpand\dtl@testFPiseq{#1}{#2}%
```

```
  \noexpand\if@dtl@condition
}
```

`\DTLisFPeq` Synonym.

```
\let\DTLisFPeq\DTLisnumeq
```

`\dtl@testFPislteq` Command to test if first number is less than or equal to second number where both numbers are in standard format. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testFPislteq}[2]{%
 \dtlifnumlt{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
   \@dtl@conditionfalse
 }%
 \if@dtl@condition
 \else
  \dtl@testFPiseq{#1}{#2}%
 \fi
}
```

`\DTLisnumlteq` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisnumlteq}[2]{%
 \TE@throw\noexpand\dtl@testFPislteq{#1}{#2}%
 \noexpand\if@dtl@condition
}
```

`\DTLisFPlteq` Synonym.

```
\let\DTLisFPlteq\DTLisnumlteq
```

`\dtl@testFPisgteq` Command to test if first number is greater than or equal to second number where both numbers are in standard format. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testFPisgteq}[2]{%
 \dtlifnumgt{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
   \@dtl@conditionfalse
 }%
 \if@dtl@condition
 \else
  \dtl@testFPiseq{#1}{#2}%
 \fi
}
```

`\DTLisnumgteq` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisnumgteq}[2]{%
```

```
    \TE@throw\noexpand\dtl@testFPisgteq{#1}{#2}%
    \noexpand\if@dtl@condition}
```

`\DTLisFPgteq` Synonym.

```
\let\DTLisFPgteq\DTLisnumgteq
```

`\dtl@teststring` Command to test if argument is a string. This sets `\if@dtl@condition`

```
\newcommand*{\dtl@teststring}[1]{%
  \DTLifstring{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

`\DTLisstring` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisstring}[1]{%
  \TE@throw\noexpand\dtl@teststring{#1}\noexpand\if@dtl@condition}
```

`\dtl@testnumerical` Command to test if argument is a numerical. This sets `\if@dtl@condition`

```
\newcommand*{\dtl@testnumerical}[1]{%
  \DTLifnumerical{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisnumerical` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisnumerical}[1]{%
  \TE@throw\noexpand\dtl@testnumerical{#1}\noexpand\if@dtl@condition}
```

`\dtl@testint` Command to test if argument is an integer. This sets `\if@dtl@condition`

```
\newcommand*{\dtl@testint}[1]{%
  \DTLifint{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

`\DTLisint` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisint}[1]{%
  \TE@throw\noexpand\dtl@testint{#1}\noexpand\if@dtl@condition}
```

`\dtl@testreal` Command to test if argument is a real. This sets `\if@dtl@condition`

```
\newcommand*{\dtl@testreal}[1]{%
  \DTLifreal{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

`\DTLisreal` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisreal}[1]{%
  \TE@throw\noexpand\dtl@testreal{#1}\noexpand\if@dtl@condition}
```

`\dtl@testcurrency` Command to test if argument is a currency. This sets `\if@dtl@condition`

```
\newcommand*{\dtl@testcurrency}[1]{%
  \DTLifcurrency{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

`\DTLiscurrency` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLiscurrency}[1]{%
  \TE@throw\noexpand\dtl@testcurrency{#1}\noexpand\if@dtl@condition}
```

\dtl@testcurrencyunit Command to test if argument is a currency with given unit. This sets \if@dtl@condition

```
\newcommand*{\dtl@testcurrencyunit}[2]{%
  \DTLifcurrencyunit{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

\DTLiscurrencyunit Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLiscurrencyunit}[2]{%
  \TE@throw\noexpand\dtl@testcurrencyunit{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

2.8 Loops

\dtlbreak Break out of loop at the end of current iteration.

```
\newcommand*{\dtlbreak}{%
  \PackageError{datatool}{Can't break out of anything}{}%
}
```

\dtlforint NB LaTeX3 now provides a better integer step function, which should be used instead. This is retained for backward-compatibility.

\dtlforint⟨*ct*⟩=⟨*start*⟩\to⟨*end*⟩\step ⟨*inc*⟩\do{⟨*body*⟩}

⟨*ct*⟩ is a count register, ⟨*start*⟩, ⟨*end*⟩ and ⟨*inc*⟩ are integers. Group if nested or use \dtlgforint. An infinite loop may result if ⟨*inc*⟩$= 0$ and ⟨*start*⟩ $\leq$ ⟨*end*⟩ and \dtlbreak isn't used.

```
\long\def\dtlforint#1=#2\to#3\step#4\do#5{%
```

Make a copy of old version of break function

```
  \let\@dtl@orgbreak\dtlbreak
  \def\@dtl@endloophook{}%
```

Setup break function for the loop (sets ⟨*ct*⟩ to ⟨*end*⟩ at the end of the current iteration).

```
  \def\dtlbreak{\def\@dtl@endloophook{#1=#3}}%
```

Initialise ⟨*ct*⟩

```
  #1=#2\relax
```

Check if the steps are positive or negative.

```
  \ifnum#4<0\relax
```

Counting down

```
    \whiledo{\(#1>#3\)\TE@or\(#1=#3\)}%
    {%
      #5%
      \@dtl@endloophook
      \advance#1 by #4\relax
    }%
  \else
```

Counting up

```
      \whiledo{\(#1<#3\)\TE@or\(#1=#3\)}%
      {%
        #5%
        \@dtl@endloophook
        \advance#1 by #4\relax
      }%
    \fi
```

Restore break function.

```
    \let\dtlbreak\@dtl@orgbreak
  }
```

`\@dtl@foreach@level` Count register to keep track of global nested loops.

```
\newcount\@dtl@foreach@level
```

`\dtlgforint`

```
\dtlgforint⟨ct⟩=⟨start⟩\to⟨end⟩\step
⟨inc⟩\do{⟨body⟩}
```

⟨*ct*⟩ is a count register, ⟨*start*⟩, ⟨*end*⟩ and ⟨*inc*⟩ are integers. An infinite loop may result if ⟨*inc*⟩=0 and ⟨*start*⟩ ≤ ⟨*end*⟩ and `\dtlbreak` isn't used.

```
\long\def\dtlgforint#1=#2\to#3\step#4\do#5{%
```

Initialise

```
    \global#1=#2\relax
```

Increment level counter to allow for nested loops

```
    \global\advance\@dtl@foreach@level by 1\relax
```

Set up end loop hook

```
    \expandafter\global\expandafter
      \let\csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
      \relax
```

Set up the break function: Copy current definition

```
    \expandafter\global\expandafter
      \let\csname @dtl@break@\the\@dtl@foreach@level\endcsname
      \dtlbreak
```

Set up definition for this level (sets <ct> to <end> at the end of the current iteration).

```
    \gdef\dtlbreak{\expandafter
       \gdef\csname @dtl@endhook@\the\@dtl@foreach@level\endcsname{%
         #1=#3}}%
```

check the direction

```
    \ifnum#4<0\relax
```

Counting down

```
      \whiledo{\(#1>#3\)\TE@or\(#1=#3\)}%
      {%
```

```
      #5%
      \csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
      \global\advance#1 by #4\relax
    }%
  \else
```

Counting up (or 0 increments)

```
    \whiledo{\(#1<#3\)\TE@or\(#1=#3\)}%
    {%
      #5%
      \csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
      \global\advance#1 by #4\relax
    }%
  \fi
```

Restore break function

```
  \expandafter\global\expandafter\let\expandafter\dtlbreak
    \csname @dtl@break@\the\@dtl@foreach@level\endcsname
```

Decrement level counter

```
  \global\advance\@dtl@foreach@level by -1\relax
}
```

`dtlenvgforint` (*env.*) Environment form (contents are gathered, so verbatim can't be used):

```
\NewDocumentEnvironment{dtlenvgforint}
 { m +b }
 {\dtlgforint#1\do{#2}}
 {}
```

`dtlenvgforint*` (*env.*) Starred form suppresses space-trimming:

```
\NewDocumentEnvironment{dtlenvgforint*}
 { m +!b }
 {\dtlgforint#1\do{#2}}
 {}
```

2.9 Localisation Support

Load localisation support. None provided with datatool, but support can be added by defining a file called `datatool-`⟨*tag*⟩`.ldf` that adds a redefinition of `\DTLCurrentLocaleWordHandler` and `\DTLCurrentLocaleGetInitialLetter` to the language hook. If a region is supplied, also set the currency.

```
\ExplSyntaxOn
```

Provide quick way to reset all localisation support.

`\DTLresetLanguage` Reset all language (not region) related commands.

```
\newcommand\DTLresetLanguage
{
  \tl_clear:N \l_datatool_current_language_tl
  \renewcommand \DTLandname { \& }
```

```
  \renewcommand \DTLdatatypeunsetname { unset }
  \renewcommand \DTLdatatypestringname { string }
  \renewcommand \DTLdatatypeintegername { integer }
  \renewcommand \DTLdatatypedecimalname { decimal }
  \renewcommand \DTLdatatypecurrencyname { currency }
  \renewcommand \DTLdatatypedatetimename { date-time }
  \renewcommand \DTLdatatypedatename { date }
  \renewcommand \DTLdatatypetimename { time }
  \renewcommand \DTLdatatypeinvalidname { invalid }
  \renewcommand \DTLCurrentLocaleWordHandler [1] { }
  \renewcommand \DTLCurrentLocaleGetInitialLetter [2]
   {
    \datatool_get_first_letter:nN { ##1 } ##2
   }
  \renewcommand \DTLCurrentLocaleGetGroupString [3]
   {
     \tl_set:Nn ##3 { ##1 }
   }
  \renewcommand \DTLCurrentLocaleGetMonthNameMap [1]
   {
    \tl_set_eq:NN \l_datatool_monthname_map_value_tl \q_no_value
   }
  \renewcommand \DTLCurrentLocaleIfpmTF [ 3 ]
   {
     \tl_if_eq:nnTF { ##1 } { pm } { ##2 } { ##3 }
   }
  \renewcommand \dtllettergroup [1]
   {
     \text_titlecase_first:n { ##1 }
   }
  \renewcommand \dtlnonlettergroup [1]
   {
     \detokenize { ##1 }
   }
  \renewcommand \dtlnumbergroup [1] { ##1 }
  \renewcommand \dtlcurrencygroup [2] { ##1 }
  \renewcommand \dtldatetimegroup [1] { ##1 }
  \renewcommand \dtldategroup [1] { ##1 }
  \renewcommand \dtltimegroup [1] { ##1 }
}
```

`\DTLresetRegion` Reset all region related commands.

```
\newcommand\DTLresetRegion
{
  \tl_clear:N \l_datatool_current_region_tl
  \DTLsetnumberchars { , } { . }
  \DTLsetdefaultcurrency { XXX }
  \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
  \renewcommand \dtlcurrfmtsymsep { }
  \renewcommand \DTLCurrentLocaleParseTimeStamp [ 4 ] { ##4 }
```

```
    \renewcommand \DTLCurrentLocaleParseDate [ 4 ] { ##4 }
    \renewcommand \DTLCurrentLocaleParseTime [ 4 ] { ##4 }
    \renewcommand \DTLCurrentLocaleGetTimeZoneMap [ 1 ]
     {
       \tl_set_eq:NN \l_datatool_timezone_map_value_tl \q_no_value
     }
    \renewcommand \DTLCurrentLocaleFormatDate [ 4 ]
     {
       \datatool_default_date_fmt:nnnn { ##1 } { ##2 } { ##3 } { ##4 }
     }
    \renewcommand \DTLCurrentLocaleFormatTime [ 3 ]
     {
       \datatool_default_time_fmt:nnn { ##1 } { ##2 } { ##3 }
     }
    \renewcommand \DTLCurrentLocaleFormatTimeZone [ 2 ]
     {
       \datatool_default_timezone_fmt:nn { ##1 } { ##2 }
     }
    \renewcommand \DTLCurrentLocaleFormatTimeStampNoZone [7]
     {
       \datatool_default_timestamp_fmt:nnnnnnn
         { ##1 } { ##2 } { ##3 } { ##4 } { ##5 } { ##6 } { ##7 }
     }
    \renewcommand \DTLCurrentLocaleFormatTimeStampWithZone [9]
     {
      \datatool_default_timestamp_fmt:nnnnnnnnn
        { ##1 } { ##2 } { ##3 } { ##4 } { ##5 } { ##6 } { ##7 } { ##8 } { ##9 }
     }
    \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
  }
```

`\RequireDatatoolDialect`

```
 \newcommand \RequireDatatoolDialect [1]
 {
  \tl_clear:N \l_datatool_current_language_tl
  \__datatool_require_ldf:
    [
```

Ensure region language file is loaded, if it exists.

```
     \tl_if_empty:NF \CurrentTrackedRegion
      {
        \TrackLangRequireResource { \CurrentTrackedRegion }
```

Note that the region file is only loaded once. This means that if there are multiple dialects that share a region, the region hook would only be added to the first dialect with that region. Therefore, the region file shouldn't add anything to the captions hook but should instead just define a command called `\DTL`⟨*Region*⟩`LocaleHook` where ⟨*Region*⟩ is the two letter (uppercase) region code. This means that it can be added here, even if the file wasn't loaded at this point.

```
        \tl_if_exist:cTF { DTL \CurrentTrackedRegion LocaleHook }
```

```
        {
          \exp_args:Nc \TrackLangAddToCaptions
            { DTL \CurrentTrackedRegion LocaleHook }
        }
        {
          \tl_new:c { DTL \CurrentTrackedRegion LocaleHook }
          \datatool_locale_warn:nn
           { datatool-base }
           {
             No ~ locale ~ hook ~ available ~ for ~ region ~ ` \CurrentTrackedRegion '
           }
        }
     }
```

Search as usual from the locale tag to pick up any specific language + region, sub-languages or scripts.

```
    \TrackLangRequireResource { \CurrentTrackedTag }
   ]
   { datatool } { #1 }
  \tl_if_empty:NT \l_datatool_current_language_tl
   {
```

No support for this language so clear the token list in the language hook.

```
     \@TrackLangAddToHook
      {
        \tl_clear:N \l_datatool_current_language_tl
      }
      { captions }
   }
  \tl_if_empty:NF \CurrentTrackedRegion
   {
```

If the region isn't empty but the hook hasn't been defined then the region file wasn't loaded. This may be because there's no language support.

```
     \tl_if_exist:cF { DTL \CurrentTrackedRegion LocaleHook }
      {
        \cs_if_exist:NT \TrackLangRequireDialectOmitDialectLabel
         {
           \TrackLangRequireDialectOmitDialectLabel { datatool} { #1 }
           \csuse { DTL \CurrentTrackedRegion LocaleHook }
         }
      }
   }
 }
```

It's best to omit dialect label and region code from search to ensure the search isn't prematurely terminated.

```
 \cs_new:Nn \__datatool_require_ldf:
  {
   \TrackLangRequireDialectOmitDialectLabelOmitOnlyRegion
  }
```

Provide interface for locale options. Define options using l3keys interface.

```
\cs_new:Nn \datatool_locale_define_keys:nn
{
  \keys_define:nn { datatool / locale / #1 } { #2 }
}
```

\DTLsetLocaleOptions

```
\DTLsetLocaleOptions[⟨parent module(s)⟩]{⟨module(s)⟩}
{⟨key-val list⟩}
```

User command for setting options for the given locale. The starred version ignores unknown keys.

```
\NewDocumentCommand \DTLsetLocaleOptions { s O{} m m }
 {
   \IfBlankTF { #2 }
    {
      \IfBooleanTF { #1 }
       {
         \datatool_set_known_locale_options:nn { #3 } { #4 }
       }
       {
         \datatool_set_locale_options:nn { #3 } { #4 }
       }
    }
    {
      \clist_map_inline:nn { #2 }
       {
         \IfBooleanTF { #1 }
          {
            \datatool_set_known_locale_options:nn { ##1 / #3 } { #4 }
          }
          {
            \datatool_set_locale_options:nn { ##1 / #3 } { #4 }
          }
       }
    }
 }
```

Set options (error on unknown):

```
\cs_new:Nn \datatool_set_locale_options:nn
 {
   \clist_map_inline:nn { #1 }
    {
      \keys_set:nn { datatool / locale / ##1 } { #2 }
    }
 }
```

Set options (ignore unknown):

```
\cs_new:Nn \datatool_set_known_locale_options:nn
```

```
{
  \clist_map_inline:nn { #1 }
   {
     \keys_set_known:nn { datatool / locale / ##1 } { #2 }
   }
}
```

Provided for the ldf files to add to the captions hook to allow the preferred label for language and region (that may be different from the dialect or language label).

```
\tl_new:N \l_datatool_current_language_tl
\tl_new:N \l_datatool_current_region_tl
```

`\datatool_if_current_lang_region:nn {`⟨*lang*⟩`} {`⟨*region*⟩`}`

Check if current language and region match ⟨*lang*⟩ and ⟨*region*⟩.

```
\prg_new_conditional:Npnn \datatool_if_current_lang_region:nn #1 #2
   { T, F, TF }
 {
  \tl_if_eq:NnTF \l_datatool_current_language_tl { #1 }
   {
    \tl_if_eq:NnTF \l_datatool_current_region_tl { #2 }
     { \prg_return_true: }
     { \prg_return_false: }
   }
   { \prg_return_false: }
 }
```

Check if empty language.

```
\cs_new:Nn \datatool_warn_check_language_empty:nnn
 {
   \tl_if_empty:NTF \l_datatool_current_language_tl
    {
      \cs_if_exist:NTF \TrackLangRequireDialectOmitDialectLabelOmitOnlyRegion
       {
         \datatool_locale_warn:nn
          { #1 }
          {
            #3 ~
            (check ~ your ~ localisation ~ setting ~ includes ~
             language ~ and ~ region)
          }
       }
       {
         \datatool_locale_warn:nn
          { #1 }
          {
            #2 ~
            (try ~ updating ~ tracklang ~ to ~ at ~ least ~ version ~ 1.6.3)
```

```
          }
        }
      }
      { \datatool_locale_warn:nn { #1 } { #3 } }
  }
```

Check if token list starts with or is `\l_datatool_current_language_tl` and issue applicable warning.

```
 \cs_new:Nn \datatool_warn_check_head_language_empty:nnnn
  {
    \tl_if_head_eq_meaning:nNTF { #1 } \l_datatool_current_language_tl
     {
       \datatool_warn_check_language_empty:nnn { #2 } { #3 } { #4 }
     }
     {
       \datatool_locale_warn:nn { #2 } { #4 }
     }
  }
 \cs_generate_variant:Nn  \datatool_warn_check_head_language_empty:nnnn
  { Vnnn }
```

Track each additional locale requested in the locales/lang package option. Note that `\TrackLanguageTag` requires the language first but, for datatool, allow just the region. This needs to test if the supplied tag has exactly two uppercase characters. If extra information, such as the script is required, then the language part must be included.

```
 \clist_map_inline:Nn \l__datatool_extra_locales_clist
  {
    \regex_match:nnTF { \A [A-Z]{2} \Z } { #1 }
     {
       \tl_set:Nn \l__datatool_tmpa_tl
        {
          \dtl@message { Adding ~ tracked ~ locale ~ ` und-#1 ' }
          \TrackLanguageTag { und-#1 }
        }
       \ForEachTrackedDialect{\l__datatool_dialect_tl}
        {
          \IfTrackedIsoCode
            { \TwoLetterIsoCountryCode }
            { \l__datatool_dialect_tl }
            {
              \dtl@message
               {
                 Dialect ~ ` \l__datatool_dialect_tl' ~ already ~ has ~ region
               }
            }
            {
              \dtl@message
               { Adding ~ region ~ `#1' ~ to ~ locale ~ ` \l__datatool_dialect_tl ' }
              \AddTrackedRegion { #1 } { \l__datatool_dialect_tl }
              \tl_clear:N \l__datatool_tmpa_tl
```

```
            }
          }
        \l__datatool_tmpa_tl
      }
      {
        \TrackIfKnownLanguage { #1 }
        {
          \dtl@message { Adding ~ tracked ~ locale ~ ` #1 ' }
        }
        {
          \datatool_locale_warn:nn
           { datatool-base }
          { Unrecognised ~ language ~ in ~ locale ~ tag ~ ` #1 '}
        }
      }
  }
```

`datatool@do@load@locales` The actual ldf loading needs to be after LaTeX3 syntax has been switched off, so define a command for use afterwards.

```
\newcommand{\datatool@do@load@locales}{}
\datatool@load@locales
{
  \renewcommand\datatool@do@load@locales
   {
    \AnyTrackedLanguages
     {
       \cs_if_exist:NF
         \TrackLangRequireDialectOmitDialectLabelOmitOnlyRegion
        {
          \datatool_locale_warn:nn
           {
            old ~ version ~ of ~ tracklang: ~ localisation ~ files ~
            may ~ not ~ all ~ load. ~ Upgrade ~ to ~ at ~ least ~
            tracklang ~ v1.6.3 ~ for ~ better ~ support
           }
          \cs_set:Nn \__datatool_require_ldf:
           { \TrackLangRequireDialect }
        }
       \ForEachTrackedDialect { \l__datatool_dialect_tl }
        {
         \RequireDatatoolDialect { \l__datatool_dialect_tl }
        }
     }
     { }
   }
}
```

Switch off LaTeX3 syntax:

```
\ExplSyntaxOff
```

Now load applicable locale files for each tracked dialect.

```
\datatool@do@load@locales
```

`\@dtl@gobbletonil` NB still used by datatool.sty. May be removed in future.

```
\def\@dtl@gobbletonil#1\@nil{}
```

`\@dtl@countdigits` NB still used by datatool-pgfmath.def May be removed in future.

```
\def\@dtl@countdigits#1#2\relax{%
  \advance\@dtl@tmpcount by 1\relax
  \ifx.#2\relax
    \let\@dtl@countnext=\@gobble
  \else
    \let\@dtl@countnext=\@dtl@countdigits
  \fi
  \@dtl@countnext#2\relax
}
```

2.10 Deprecated

These commands are being phased out and should not be used.

`\dtlenableUTFviii` Deprecated.

```
\newcommand*{\dtlenableUTFviii}{\booltrue{@dtl@utf8}}
```

`\dtldisableUTFviii` Deprecated.

```
\newcommand*{\dtldisableUTFviii}{\boolfalse{@dtl@utf8}}
```

`\long@collect@body` Need long versions of amsmath's `\collect@body`. These macros are adapted from the macros defined by amsmath. Use xparse instead.

```
\long\def\long@collect@body#1{%
  \@envbody{\@xp#1\@xp{\the\@envbody}}%
  \edef\process@envbody{\the\@envbody\@nx\end{\@currenvir}}%
  \@envbody\@emptytoks \def\begin@stack{b}%
  \begingroup
  \@xp\let\csname\@currenvir\endcsname\long@collect@@body
  \edef\process@envbody{\@xp\@nx\csname\@currenvir\endcsname}%
  \process@envbody
}
```

`\long@addto@envbody` Adapted from amsmath's `\addto@envbody`

```
\long\def\long@addto@envbody#1{%
  \toks@{#1}%
  \edef\@dtl@tmp{\the\@envbody\the\toks@}%
  \global\@envbody\@xp{\@dtl@tmp}%
}
```

\long@collect@@body Adapted from amsmath's \collect@body

```
\long\def\long@collect@@body#1\end#2{%
  \protected@edef\begin@stack{%
    \long@push@begins#1\begin\end \@xp\@gobble\begin@stack
  }%
  \ifx\@empty\begin@stack
    \endgroup
    \@checkend{#2}%
    \long@addto@envbody{#1}%
  \else
    \long@addto@envbody{#1\end{#2}}%
  \fi
  \process@envbody
}
```

\long@push@begins Adapted from amsmath's \push@begins

```
\long\def\long@push@begins#1\begin#2{%
  \ifx\end#2\else b\@xp\long@push@begins\fi
}
```

\dtl@ifsingle

\dtl@ifsingle{⟨*arg*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

If there is only one object in ⟨*arg*⟩ (without expansion) do ⟨*true part*⟩, otherwise do false part. This now just uses the new LaTeX3 command to test for a single token. Old versions of glossaries use \dtl@ifsingle.

```
\ExplSyntaxOn
\newcommand \dtl@ifsingle [ 3 ]
 {
  \tl_if_single_token:nTF { #1 } { #2 } { #3 }
 }
\ExplSyntaxOff
```

\@dtl@ifsingle

Count registers to store character codes:

```
\newcount\dtl@codeA
\newcount\dtl@codeB
```

l@listelement@outofrange

```
\newcommand{\@dtl@listelement@outofrange}[1]{%
  \PackageWarning{datatool-base}{List index `\number#1' out of range}%
}
```

\dtl@sortlist I made a bit of a blunder here. \dtl@sortlist was supposed to work with commands like \dtlcompare, but those commands require three arguments, the first being the register in which to store the result. This contradicts the requirements of \dtl@sortlist. The "bug fix" in v2.26 fixed it to work with commands like

\dtlcompare, but that broke the documented design (which breaks the glossaries package). The other problem is that \dtl@insertinto actually sorts in the reverse order. So v2.27 undoes the change from v2.26 to ensure backward compatibility and provides an alternative user-level command \dtlsortlist, that's designed to work with the three-argument handler commands like \dtlcompare.

\dtl@sortlist{⟨*list*⟩}{⟨*criteria cmd*⟩}

Performs an insertion sort on ⟨*list*⟩, where ⟨*criteria cmd*⟩ is a macro which takes two arguments ⟨*a*⟩ and ⟨*b*⟩. ⟨*criteria cmd*⟩ must set the count register \dtl@sortresult to either −1 (⟨*b*⟩ less than ⟨*a*⟩), 0 (⟨*a*⟩ is equal to ⟨*b*⟩) or 1 (⟨*b*⟩ is greater than ⟨*a*⟩.)

DEPRECATED. To be removed. (NB used by glossaries.sty)

```
\newcommand{\dtl@sortlist}[2]{%
\def\@dtl@sortedlist{}%
\@for\@dtl@currentrow:=#1\do{%
\expandafter\dtl@insertinto\expandafter
  {\@dtl@currentrow}{\@dtl@sortedlist}{#2}%
\@endforfalse}%
\let#1=\@dtl@sortedlist
}
```

\dtl@insertinto

\dtl@insertinto{⟨*element*⟩}{⟨*sorted-list*⟩}{⟨*criteria cmd*⟩}

Inserts ⟨*element*⟩ into the sorted list ⟨*sorted-list*⟩ according to the criteria given by ⟨*criteria cmd*⟩ (see above.) DEPRECATED. To be removed. NB used by glossaries.sty

```
\newcommand{\dtl@insertinto}[3]{%
  \def\@dtl@newsortedlist{}%
  \@dtl@insertdonefalse
  \@for\dtl@srtelement:=#2\do{%
    \if@dtl@insertdone
      \expandafter\toks@\expandafter{\dtl@srtelement}%
      \edef\@dtl@newstuff{{\the\toks@}}%
    \else

      \expandafter#3\expandafter{\dtl@srtelement}{#1}%

      \ifnum\dtl@sortresult<0\relax
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \@dtl@toks{#1}%
        \edef\@dtl@newstuff{{\the\@dtl@toks},{\the\toks@}}%
        \@dtl@insertdonetrue
      \else
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \edef\@dtl@newstuff{{\the\toks@}}%
      \fi
    \fi
```

```
    \ifdefempty{\@dtl@newsortedlist}%
    {%
      \expandafter\toks@\expandafter{\@dtl@newstuff}%
      \edef\@dtl@newsortedlist{\the\toks@}%
    }%
    {%
      \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
      \expandafter\@dtl@toks\expandafter{\@dtl@newstuff}%
      \edef\@dtl@newsortedlist{\the\toks@,\the\@dtl@toks}%
    }%
    \@endforfalse
  }%
  \ifdefempty{\@dtl@newsortedlist}%
  {%
    \@dtl@toks{#1}%
    \edef\@dtl@newsortedlist{{\the\@dtl@toks}}%
  }%
  {%
    \if@dtl@insertdone
    \else
      \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
      \@dtl@toks{#1}%
      \edef\@dtl@newsortedlist{\the\toks@,{\the\@dtl@toks}}%
    \fi
  }%
  \global\let#2=\@dtl@newsortedlist
}
```

`\dtl@ifsingleorUTFviii` Test if the argument is a single token or, if utf8 setting on, a two-octet UTF-8 character. This only tests for two octet UTF-8 characters.

```
\ExplSyntaxOn
\newcommand \dtl@ifsingleorUTFviii [ 3 ]
{
 \tl_if_single_token:nTF { #1 }
 { #2 }
 {
   \ifbool{@dtl@utf8}
   {
     \tl_if_head_is_N_type:nTF { #1 }
     {
       \__datatool_isheadactivechar:w#1\q_stop
       {
         \ifnum\tl_count_tokens:n { #1 } = 2
           \expandafter \__datatool_isheadUTFviiitwooctets \expandafter
           { \__datatool_xpactiveheadchar }
           { #2 }
           { #3 }
         \else
           #3
         \fi
```

```
        }
        { #3 }
      }
      { #3 }
    }
    { #3 }
  }
}
```

Tests if the argument starts with an active character.

```
\catcode`\@13\relax
\cs_new:Npn \__datatool_isheadactivechar:w #1 #2 \q_stop #3 #4
{
  \tl_if_head_eq_catcode:nNTF { #1 } @
  { \tl_set:No \__datatool_xpactiveheadchar { #1 } #3 }
  { #4 }
}
\catcode`\@11\relax
```

Tests if the first argument starts with `\UTFviii@two@octets`.

```
\cs_new:Npn \__datatool_isheadUTFviiitwooctets #1 #2 #3
{
 \tl_if_head_eq_meaning:nNTF { #1 } \UTFviii@two@octets
 { #2 } { #3 }
}
\ExplSyntaxOff
```

`\dtl@if@two@octets` Check if argument starts with `\UTFviii@two@octets` Deprecated.

```
\def\dtl@if@two@octets#1#2\dtl@end@if@two@octets#3#4{%
  \ifbool{@dtl@utf8}
  {%
    \ifx\UTFviii@two@octets#1\relax
     #3%
    \else
     #4%
    \fi
  }%
  {%
    #4%
  }%
}
```

`\dtl@getfirst@UTFviii` Deprecated.

```
\def\dtl@getfirst@UTFviii#1#2#3\end@dtl@getfirst@UTFviii{%
  \def\dtl@first{#1#2}%
  \ifx\@nil#3\relax
    \def\dtl@rest{}%
  \else
    \expandafter\def\expandafter\dtl@rest\expandafter{\@dtl@firsttonil#3}%
  \fi
```

```
}
```

\@dtl@firsttonil

```
\def\@dtl@firsttonil#1\@nil{#1}
```

\@dtlgetfirstchar

\@dtlgetfirstchar{⟨*text*⟩}{⟨*cs1*⟩}{⟨*cs2*⟩}

Stores the first character in ⟨*text*⟩ in ⟨*cs1*⟩ and the remainder in ⟨*cs2*⟩. Largely redundant, but can be used to obtain the letter group of a UTF-8 string.

```
\ExplSyntaxOn
\newcommand{\@dtlgetfirstchar}[3]{
   \tl_if_empty:nTF { #1 }
   { \cs_set:Nn #2 {} \cs_set:Nn #3 {} }
   {
     \tl_if_single_token:nTF { #1 }
     { \cs_set:Nn #2 { #1 } \cs_set:Nn #3 {} }
     {
       \edef #2 { \tl_head:n { #1 } }
       \edef #3 { \tl_tail:n { #1 } }
       \ifbool{@dtl@utf8}
       {
         \tl_if_head_is_N_type:nTF { #1 }
         {
           \__datatool_isheadactivechar:w#1\q_stop
           {
             \expandafter \__datatool_isheadUTFviiitwooctets \expandafter
             { \__datatool_xpactiveheadchar }
             {
               \tl_put_right:Nx #2 { \tl_head:N #3 }
               \edef #3 { \tl_tail:N #3 }
             }
             { }
           }
           { }
         }
         { }
       }
       { }
     }
   }
 }
\ExplSyntaxOff
```

\dtl@getfirst Gets the first object, and stores in `\dtl@first`. The remainder is stored in `\dtl@rest`. Deprecated.

```
\def\dtl@getfirst#1#2\end@dtl@getfirst{%
  \dtlgetfirstchar{#1#2}{\dtl@first}{\dtl@rest}%
}%
```

\dtl@setcharcode

```
\dtl@setcharcode{⟨c⟩}{⟨count register⟩}
```

Sets ⟨*count register*⟩ to the character code of ⟨*c*⟩, or to −1 if ⟨*c*⟩ is empty, or to 0 if ⟨*c*⟩ is a control sequence, unless ⟨*c*⟩ is either `\space` or | | in which case it sets ⟨*count register*⟩ to the character code of the space character. Largely redundant. Likely to be removed in future.

```
\ExplSyntaxOn
\newcommand*{\dtl@setcharcode}[2]{%
 \exp_args:Nx \__datatool_setcharcode:nN { \text_purify:n { #1 } } #2
}
\cs_new:Npn \__datatool_setcharcode:nN #1#2
{
  \tl_if_empty:nTF { #1 }
  { #2=-1\relax }
  {
    \tl_if_head_is_space:nTF { #1 }
    { #2=32\relax }
    {
      \tl_if_head_eq_catcode:nNTF { #1 } \relax
      { #2=0\relax }
      {
        \tl_if_single_token:nTF { #1 }
        { \dtlsetcharcode{#1}{#2} }
        {
          \exp_args:Nx \dtlsetcharcode { \tl_head:n { #1 } } {#2}
          \ifbool{@dtl@utf8}
          {
            \ifnum#2>127
              \dtlsetUTFviiicharcode { #1 }{ #2 }
            \fi
          }
          { }
        }
      }
    }
  }
}
\ExplSyntaxOff
```

\dtlsetcharcode Set the code for the given character. May be redefined by user for non-UTF8 encodings (e.g. Latin-1). Deprecated.

```
\newcommand*{\dtlsetcharcode}[2]{#2=`#1\relax}
```

\dtlsetlccharcode Set the lowercase code for the given character. May be redefined by user for non-UTF8 encodings (e.g. Latin-1). Deprecated.

```
\newcommand*{\dtlsetlccharcode}[2]{#2=\lccode`#1\relax}
```

\dtlsetUTFviiicharcode Default behaviour is to set all UTF8 characters to code 64 (before A). This will need to be redefined according to the relevant alphabet. Deprecated.

```
\newcommand*\dtlsetUTFviiicharcode[2]{\dtlsetdefaultUTFviiicharcode{#1}{#2}}
```

\dtlsetUTFviiilccharcode Default behaviour is to set all UTF8 characters to code 96 (before a). This will need to be redefined according to the relevant alphabet. Deprecated.

```
\newcommand*\dtlsetUTFviiilccharcode[2]{\dtlsetdefaultUTFviiilccharcode{#1}{#2}}
```

etdefaultUTFviiicharcode Default codes for some supplemental Latin characters. Deprecated.

```
\newcommand*\dtlsetdefaultUTFviiicharcode[2]{%
 \ifboolexpr
 {
      test {\ifstrequal{#1}{À}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Ã}}
   or test {\ifstrequal{#1}{Ä}}
 }%
 {%
   #2=`A\relax
 }%
 {%
   \ifstrequal{#1}{Ç}%
   {%
     #2=`C\relax
   }%
   {%
     \ifboolexpr
     {
          test {\ifstrequal{#1}{È}}
       or test {\ifstrequal{#1}{É}}
       or test {\ifstrequal{#1}{Ê}}
       or test {\ifstrequal{#1}{Ë}}
     }%
     {%
       #2=`E\relax
     }%
     {%
       \ifboolexpr
       {
            test {\ifstrequal{#1}{Ì}}
         or test {\ifstrequal{#1}{Í}}
         or test {\ifstrequal{#1}{Î}}
         or test {\ifstrequal{#1}{Ï}}
       }%
       {%
         #2=`I\relax
       }%
       {%
```

```
\ifstrequal{#1}{Ñ}%
{%
  #2=`N\relax
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{Ò}}
    or test {\ifstrequal{#1}{Ó}}
    or test {\ifstrequal{#1}{Ô}}
    or test {\ifstrequal{#1}{Õ}}
    or test {\ifstrequal{#1}{Ö}}
  }%
  {%
    #2=`O\relax
  }%
  {%
    \ifboolexpr
    {
         test {\ifstrequal{#1}{Ù}}
      or test {\ifstrequal{#1}{Ú}}
      or test {\ifstrequal{#1}{Û}}
      or test {\ifstrequal{#1}{Ü}}
    }%
    {%
      #2=`U\relax
    }%
    {%
      \ifstrequal{#1}{Ý}%
      {%
        #2=`Y\relax
      }%
      {%
        \ifboolexpr
        {
             test {\ifstrequal{#1}{à}}
          or test {\ifstrequal{#1}{á}}
          or test {\ifstrequal{#1}{á}}
          or test {\ifstrequal{#1}{ã}}
          or test {\ifstrequal{#1}{ä}}
        }%
        {%
          #2=`a\relax
        }%
        {%
          \ifstrequal{#1}{ç}%
          {%
            #2=`c\relax
          }%
          {%
```

```
\ifboolexpr
{
     test {\ifstrequal{#1}{è}}
  or test {\ifstrequal{#1}{é}}
  or test {\ifstrequal{#1}{ê}}
  or test {\ifstrequal{#1}{ë}}
}%
{%
  #2=`e\relax
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{ì}}
    or test {\ifstrequal{#1}{í}}
    or test {\ifstrequal{#1}{î}}
    or test {\ifstrequal{#1}{ï}}
  }%
  {%
    #2=`i\relax
  }%
  {%
    \ifstrequal{#1}{ñ}%
    {%
      #2=`n\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{ò}}
        or test {\ifstrequal{#1}{ó}}
        or test {\ifstrequal{#1}{ô}}
        or test {\ifstrequal{#1}{õ}}
        or test {\ifstrequal{#1}{ö}}
      }%
      {%
        #2=`o\relax
      }%
      {%
        \ifboolexpr
        {
             test {\ifstrequal{#1}{ù}}
          or test {\ifstrequal{#1}{ú}}
          or test {\ifstrequal{#1}{û}}
          or test {\ifstrequal{#1}{ü}}
        }%
        {%
          #2=`u\relax
        }%
        {%
```

```
                                    \ifstrequal{#1}{ý}%
                                    {%
                                      #2=`y\relax
                                    }%
                                    {%
                                      #2=64\relax
                                    }%
                                  }%
                                }%
                              }%
                            }%
                          }%
                        }%
                      }%
                    }%
                  }%
                }%
              }%
            }%
          }%
        }%
      }%
    }%
  }%
}
```

`defaultUTFviiilccharcode` As above but for case-insensitive comparison. Deprecated.

```
\newcommand*\dtlsetdefaultUTFviiilccharcode[2]{%
 \ifboolexpr
 {
      test {\ifstrequal{#1}{à}}
   or test {\ifstrequal{#1}{á}}
   or test {\ifstrequal{#1}{á}}
   or test {\ifstrequal{#1}{ã}}
   or test {\ifstrequal{#1}{ä}}
   or test {\ifstrequal{#1}{À}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Ã}}
   or test {\ifstrequal{#1}{Ä}}
 }%
 {%
   #2=`a\relax
 }%
 {%
   \ifboolexpr
   {
        test {\ifstrequal{#1}{ç}}
     or test {\ifstrequal{#1}{Ç}}
   }
   {%
     #2=`c\relax
```

```
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{è}}
    or test {\ifstrequal{#1}{é}}
    or test {\ifstrequal{#1}{ê}}
    or test {\ifstrequal{#1}{ë}}
    or test {\ifstrequal{#1}{È}}
    or test {\ifstrequal{#1}{É}}
    or test {\ifstrequal{#1}{Ê}}
    or test {\ifstrequal{#1}{Ë}}
  }%
  {%
    #2=`e\relax
  }%
  {%
    \ifboolexpr
    {
         test {\ifstrequal{#1}{ì}}
      or test {\ifstrequal{#1}{í}}
      or test {\ifstrequal{#1}{î}}
      or test {\ifstrequal{#1}{ï}}
      or test {\ifstrequal{#1}{Ì}}
      or test {\ifstrequal{#1}{Í}}
      or test {\ifstrequal{#1}{Î}}
      or test {\ifstrequal{#1}{Ï}}
    }%
    {%
      #2=`i\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{ñ}}
        or test {\ifstrequal{#1}{Ñ}}
      }
      {%
        #2=`n\relax
      }%
      {%
        \ifboolexpr
        {
             test {\ifstrequal{#1}{ò}}
          or test {\ifstrequal{#1}{ó}}
          or test {\ifstrequal{#1}{ô}}
          or test {\ifstrequal{#1}{õ}}
          or test {\ifstrequal{#1}{ö}}
          or test {\ifstrequal{#1}{Ò}}
          or test {\ifstrequal{#1}{Ó}}
```

```
              or test {\ifstrequal{#1}{Ô}}
              or test {\ifstrequal{#1}{Õ}}
              or test {\ifstrequal{#1}{Ö}}
            }%
            {%
              #2=`o\relax
            }%
            {%
              \ifboolexpr
              {
                   test {\ifstrequal{#1}{ù}}
                or test {\ifstrequal{#1}{ú}}
                or test {\ifstrequal{#1}{û}}
                or test {\ifstrequal{#1}{ü}}
                or test {\ifstrequal{#1}{Ù}}
                or test {\ifstrequal{#1}{Ú}}
                or test {\ifstrequal{#1}{Û}}
                or test {\ifstrequal{#1}{Ü}}
              }%
              {%
                #2=`u\relax
              }%
              {%
                \ifboolexpr
                {
                     test {\ifstrequal{#1}{ý}}
                  or test {\ifstrequal{#1}{Ý}}
                }%
                {%
                  #2=`y\relax
                }%
                {%
                  #2=96\relax
                }%
              }%
            }%
          }%
        }%
      }%
    }%
  }%
}
```

3 datatool-undetermined.ldf

Minimal support for undetermined language.

```
\TrackLangProvidesResource{undetermined}[2026/07/31 v3.4.4]
```

Provide hook to customized:

```
\ExplSyntaxOn
\newcommand \DTLundLocaleHook
 {
  \DTLresetLanguage
  \tl_set:Nn \l_datatool_current_language_tl { und }
 }
\ExplSyntaxOff
```

Add to captions hook.

```
\TrackLangAddToCaptions{\DTLundLocaleHook}
```

4 datatool-latin1.ldf

ISO-8859-1 (Latin-1) strings.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{datatool-latin1.ldf}[2026/07/31 v3.4.4 (NLCT)]
\ExplSyntaxOn
```

```
\datatool_set_currencysign_from_charcode:nn
  { cent } { "A2 }
\datatool_set_currencysign_from_charcode:nn
  { pound } { "A3 }
\datatool_set_currencysign_from_charcode:nn
  { currency } { "A4 }
\datatool_set_currencysign_from_charcode:nn
  { yen } { "A5 }
\datatool_set_symbol_from_charcode:nn
  { middot } { "B7 }
\ExplSyntaxOff
```

No available ISO-8859-1 characters to support other commands

5 datatool-utf8.ldf

UTF-8 strings.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{datatool-utf8.ldf}[2026/07/31 v3.4.4 (NLCT)]
\ExplSyntaxOn
```

Common numeric and currency formatting symbols.

```
\datatool_set_currencysign:nn { cent } { ¢ }
\datatool_set_currencysign:nn { pound } { £ }
\datatool_set_currencysign:nn { currency } { ¤ }
\datatool_set_currencysign:nn { yen } { ¥ }
\datatool_set_symbol:nn { middot } { · }
\datatool_set_currencysign:nn { florin } { ƒ }
\datatool_set_currencysign:nn { baht } { ฿ }
\datatool_set_currencysign:nn { ecu } { ₠ }
```

```
\datatool_set_currencysign:nn { colonsign } { ₡ }
\datatool_set_currencysign:nn { cruzerio } { ₢ }
\datatool_set_currencysign:nn { frenchfranc } { ₣ }
\datatool_set_currencysign:nn { lira } { £ }
\datatool_set_currencysign:nn { mill } { ₥ }
\datatool_set_currencysign:nn { naira } { ₦ }
\datatool_set_currencysign:nn { peseta } { ₧ }
\datatool_set_currencysign:nn { rupee } { ₨ }
\datatool_set_currencysign:nn { won } { ₩ }
\datatool_set_currencysign:nn { shekel } { ₪ }
\datatool_set_currencysign:nn { dong } { ₫ }
\datatool_set_currencysign:nn { euro } { € }
\datatool_set_currencysign:nn { kip } { ₭ }
\datatool_set_currencysign:nn { tugrik } { ₮ }
\datatool_set_currencysign:nn { drachma } { ₯ }
\datatool_set_currencysign:nn { germanpenny } { ₰ }
\datatool_set_currencysign:nn { peso } { ₱ }
\datatool_set_currencysign:nn { guarani } { ₲ }
\datatool_set_currencysign:nn { austral } { ₳ }
\datatool_set_currencysign:nn { hryvnia } { ₴ }
\datatool_set_currencysign:nn { cedi } { ₵ }
\datatool_set_currencysign:nn { livretournois } { ▫ }
\datatool_set_currencysign:nn { spesmilo } { Sm }
\datatool_set_currencysign:nn { tenge } { ▫ }
\datatool_set_currencysign:nn { indianrupee } { ▫ }
\datatool_set_currencysign:nn { turkishlira } { ▫ }
\datatool_set_currencysign:nn { nordicmark } { ▫ }
\datatool_set_currencysign:nn { manat } { ▫ }
\datatool_set_currencysign:nn { ruble } { ▫ }
\datatool_set_currencysign:nn { lari } { ▫ }
\datatool_set_currencysign:nn { bitcoin } { ฿ }
\datatool_set_currencysign:nn { som } { ▫ }
```

Regular expression to match apostrophe.

```
\regex_set:Nn \l_datatool_apos_regex { \' | ’ }
\ExplSyntaxOff
```

6 datatool-l3fp.def

Definitions of fixed-point commands that use LaTeX3 commands.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{datatool-l3fp.def}[2026/07/31 v3.4.4 (NLCT)]
```

This file provides commands that use l3fp interfaces. The commands defined here match the name and syntax of the commands in the alternative `datatool-`⟨*processor*⟩`.def` files.

6.1 Comparisons

\dtlifnumeq

\dtlifnumeq{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does ⟨*true part*⟩ if ⟨*num1*⟩=⟨*num2*⟩, otherwise does ⟨*false part*⟩.

```
\ExplSyntaxOn
\newcommand*{\dtlifnumeq}[4]{%
 \fp_compare:nNnTF { #1 } = { #2 } { #3 } { #4 }
}
```

\dtlifnumlt

\dtlifnumlt{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does ⟨*true part*⟩ if ⟨*num1*⟩ <⟨*num2*⟩, otherwise does ⟨*false part*⟩.

```
\newcommand*{\dtlifnumlt}[4]{%
 \fp_compare:nNnTF { #1 } < { #2 } { #3 } { #4 }
}
```

\dtlifnumgt

\dtlifnumgt{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does ⟨*true part*⟩ if ⟨*num1*⟩ >⟨*num2*⟩, otherwise does ⟨*false part*⟩.

```
\newcommand*{\dtlifnumgt}[4]{%
 \fp_compare:nNnTF { #1 } > { #2 } { #3 } { #4 }
}
```

\dtlifnumopenbetween

\dtlifnumopenbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩.

```
\newcommand*{\dtlifnumopenbetween}[5]{%
 \fp_compare:nTF { #2 < #1 < #3 }
   { #4 } { #5 }
}
```

\dtlifnumclosedbetween

\dtlifnumclosedbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩.

```
\newcommand*{\dtlifnumclosedbetween}[5]{%
```

```
  \fp_compare:nTF { #2 <= #1 <= #3 }
   { #4 } { #5 }
}
```

6.2 Functions

\dtladd

\dtladd{⟨cs⟩}{⟨num1⟩}{⟨num2⟩}

Adds two numbers and stores the result in ⟨cs⟩.

```
\NewDocumentCommand \dtladd { m m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { #2 + #3 }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtladdall

\dtladdall{⟨cs⟩}{⟨num list⟩}

Defines ⟨cs⟩ to the sum of all the values in the comma-separated list of numbers.

```
\NewDocumentCommand \dtladdall { m m }
{
  \fp_zero:N \l__datatool_tmpa_fp
  \exp_args:Nx \clist_map_inline:nn { #2 }
   { \fp_add:Nn \l__datatool_tmpa_fp { ##1 } }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlsub

\dtlsub{⟨cs⟩}{⟨num1⟩}{⟨num2⟩}

Subtracts two numbers and stores the result in ⟨cs⟩.

```
\NewDocumentCommand \dtlsub { m m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { #2 - #3 }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlmul

\dtlmul{⟨cs⟩}{⟨num1⟩}{⟨num2⟩}

Multiplies two numbers and stores the result in ⟨cs⟩.

```
\NewDocumentCommand \dtlmul { m m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { #2 * #3 }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtldiv

```
\dtldiv{⟨cs⟩}{⟨num1⟩}{⟨num2⟩}
```

Divides two numbers and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand \dtldiv { m m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { #2 / #3 }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlsqrt

```
\dtlsqrt{⟨cs⟩}{⟨num⟩}
```

Square root number and store the result in ⟨*cs*⟩.

```
\NewDocumentCommand \dtlsqrt { m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { sqrt ( #2 ) }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlroot

```
\dtlroot{⟨cs⟩}{⟨num⟩}{⟨n⟩}
```

Nth root number and store the result in ⟨*cs*⟩.

```
\NewDocumentCommand \dtlroot { m m m }
{
  \exp_args:Nx \tl_if_eq:nnTF { #3 } { 2 }
  {
    \fp_set:Nn \l__datatool_tmpa_fp { sqrt ( #2 ) }
    \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
  }
  {
    \fp_set:Nn \l__datatool_tmpa_fp { ( #2 ) ^ ( 1 / ( #3 ) ) }
    \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
  }
}
```

\dtlround

```
\dtlround{⟨cs⟩}{⟨num⟩}{⟨dp⟩}
```

Rounds ⟨*num*⟩ to ⟨*dp*⟩ decimal places and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand \dtlround { m m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { round ( #2, #3 ) }
  \tl_set:Nx #1 { \fp_to_decimal:N \l__datatool_tmpa_fp }
  \datatool_pad_trailing_zeros:Nn #1 { #3 }
}
```

\dtltrunc

\dtltrunc{⟨*cs*⟩}{⟨*num*⟩}{⟨*dp*⟩}

Truncates ⟨*num*⟩ to ⟨*dp*⟩ decimal places and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand \dtltrunc { m m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { trunc ( #2, #3 ) }
  \tl_set:Nx #1 { \fp_to_decimal:N \l__datatool_tmpa_fp }
  \datatool_pad_trailing_zeros:Nn #1 { #3 }
}
```

\dtlclip

\dtlclip{⟨*cs*⟩}{⟨*num*⟩}

Removes redundant trailing zeros from ⟨*num*⟩ and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand \dtlclip { m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { #2 }
  \tl_set:Nx #1 { \fp_to_decimal:N \l__datatool_tmpa_fp }
}
```

\dtlmin

\dtlmin{⟨*cs*⟩}{⟨*num1*⟩}{⟨*num2*⟩}

Defines ⟨*cs*⟩ to the smaller of the two numbers.

```
\NewDocumentCommand \dtlmin { m m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { min ( #2, #3 ) }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlminall

\dtlminall{⟨*cs*⟩}{⟨*num list*⟩}

Defines ⟨*cs*⟩ to the minimum in the comma-separated list of numbers.

```
\NewDocumentCommand \dtlminall { m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { min ( #2 ) }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlmax

\dtlmax{⟨*cs*⟩}{⟨*num1*⟩}{⟨*num2*⟩}

Defines ⟨*cs*⟩ to the larger of the two numbers.

```
\NewDocumentCommand \dtlmax { m m m }
{
```

```
  \fp_set:Nn \l__datatool_tmpa_fp { max ( #2, #3 ) }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlmaxall `\dtlmaxall{⟨cs⟩}{⟨num list⟩}`

Defines ⟨*cs*⟩ to the maximum in the comma-separated list of numbers.

```
\NewDocumentCommand \dtlmaxall { m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { max ( #2 ) }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlabs `\dtlabs{⟨cs⟩}{⟨num⟩}`

Defines ⟨*cs*⟩ to the absolute value of ⟨*num*⟩.

```
\NewDocumentCommand \dtlabs { m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { abs ( #2 ) }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlneg `\dtlneg{⟨cs⟩}{⟨num⟩}`

Defines ⟨*cs*⟩ to the negative of ⟨*num*⟩.

```
\NewDocumentCommand \dtlneg {m m }
{
  \fp_set:Nn \l__datatool_tmpa_fp { - ( #2 ) }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_tmpa_fp }
}
```

\dtlmeanforall `\dtlmeanforall{⟨cs⟩}{⟨num list⟩}`

Defines ⟨*cs*⟩ to the mean (average) of all the values in the comma-separated list of numbers.

```
\NewDocumentCommand \dtlmeanforall { m m }
{
  \fp_zero:N \l__datatool_total_fp
  \int_zero:N \l__datatool_count_int
  \exp_args:No \clist_map_inline:nn { #2 }
    {
      \int_incr:N \l__datatool_count_int
      \fp_add:Nn \l__datatool_total_fp { ##1 }
```

```
    }
  \fp_set:Nn \l__datatool_mean_fp
    { \l__datatool_total_fp / \l__datatool_count_int }
  \tl_set:Nx #1 { \fp_use:N \l__datatool_mean_fp }
}
```

\dtlvarianceforall

\dtlvarianceforall[⟨*mean*⟩]{⟨*cs*⟩}{⟨*num list*⟩}

Defines ⟨*cs*⟩ to the variance of all the values in the comma-separated list of numbers. If the mean value has already been calculated, it can be supplied in the optional argument.

```
\NewDocumentCommand \dtlvarianceforall { o m m }
{
  \IfNoValueTF { #1 }
  {
    \fp_zero:N \l__datatool_total_fp
    \int_zero:N \l__datatool_count_int
    \exp_args:No \clist_map_inline:nn { #3 }
      {
        \int_incr:N \l__datatool_count_int
        \fp_add:Nn \l__datatool_total_fp { ##1 }
      }
    \fp_set:Nn \l__datatool_mean_fp
      { \l__datatool_total_fp / \l__datatool_count_int }
    \fp_zero:N \l__datatool_total_fp
    \exp_args:No \clist_map_inline:nn { #3 }
      {
        \fp_set:Nn \l__datatool_tmpa_fp
          { ##1 - \l__datatool_mean_fp }
        \fp_add:Nn \l__datatool_total_fp
          { \l__datatool_tmpa_fp  * \l__datatool_tmpa_fp }
      }
  }
  {
    \fp_set:Nn { \l__datatool_mean_fp } { #1 }
    \fp_zero:N \l__datatool_total_fp
    \int_zero:N \l__datatool_count_int
    \exp_args:No \clist_map_inline:nn { #3 }
      {
        \int_incr:N \l__datatool_count_int
        \fp_set:Nn \l__datatool_tmpa_fp
          { ##1 - \l__datatool_mean_fp }
        \fp_add:Nn \l__datatool_total_fp
          { \l__datatool_tmpa_fp  * \l__datatool_tmpa_fp }
      }
  }
  \fp_set:Nn \l__datatool_tmpa_fp
    { \l__datatool_total_fp / \l__datatool_count_int }
  \tl_set:Nx #2 { \fp_use:N \l__datatool_tmpa_fp }
```

```
}
```

\dtlsdforall

\dtlsdforall[⟨*mean*⟩]{⟨*cs*⟩}{⟨*num list*⟩}

Defines ⟨*cs*⟩ to the standard deviation of all the values in the comma-separated list of numbers. If the mean value has already been calculated, it can be supplied in the optional argument.

```
\NewDocumentCommand \dtlsdforall { o m m }
{
  \IfNoValueTF { #1 }
  {
    \fp_zero:N \l__datatool_total_fp
    \int_zero:N \l__datatool_count_int
    \exp_args:No \clist_map_inline:nn { #3 }
      {
        \int_incr:N \l__datatool_count_int
        \fp_add:Nn \l__datatool_total_fp { ##1 }
      }
    \fp_set:Nn \l__datatool_mean_fp
      { \l__datatool_total_fp / \l__datatool_count_int }
    \fp_zero:N \l__datatool_total_fp
    \exp_args:No \clist_map_inline:nn { #3 }
      {
        \fp_set:Nn \l__datatool_tmpa_fp
          { ##1 - \l__datatool_mean_fp }
        \fp_add:Nn \l__datatool_total_fp
          { \l__datatool_tmpa_fp  * \l__datatool_tmpa_fp }
      }
  }
  {
    \fp_set:Nn \l__datatool_mean_fp { #1 }
    \fp_zero:N \l__datatool_total_fp
    \int_zero:N \l__datatool_count_int
    \exp_args:No \clist_map_inline:nn { #3 }
      {
        \int_incr:N \l__datatool_count_int
        \fp_set:Nn \l__datatool_tmpa_fp { ##1 - \l__datatool_mean_fp }
        \fp_add:Nn \l__datatool_total_fp
         { \l__datatool_tmpa_fp  * \l__datatool_tmpa_fp }
      }
  }
  \fp_set:Nn \l__datatool_tmpa_fp
    { sqrt ( \l__datatool_total_fp / \l__datatool_count_int ) }
  \tl_set:Nx #2 { \fp_use:N \l__datatool_tmpa_fp }
}

\ExplSyntaxOff
```

7 datatool-lua.def

Definitions of fixed-point commands that use Lua. Only for use with LuaLaTeX.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{datatool-lua.def}[2026/07/31 v3.4.4 (NLCT)]
```

This file provides commands that use `\directlua`. The commands defined here match the name and syntax of the commands in the alternative `datatool-`⟨*processor*⟩`.def` files.

7.1 Comparisons

`\dtlifnumeq`

`\dtlifnumeq{`⟨*num1*⟩`}{`⟨*num2*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Does ⟨*true part*⟩ if ⟨*num1*⟩=⟨*num2*⟩, otherwise does ⟨*false part*⟩.

```
\newcommand*{\dtlifnumeq}[4]{%
  \ifnum\directlua{if #1==#2 then tex.print(1) else tex.print(0) end}=1
   #3%
  \else
   #4%
  \fi
}
```

`\dtlifnumlt`

`\dtlifnumlt{`⟨*num1*⟩`}{`⟨*num2*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Does ⟨*true part*⟩ if ⟨*num1*⟩ <⟨*num2*⟩, otherwise does ⟨*false part*⟩.

```
\newcommand*{\dtlifnumlt}[4]{%
  \ifnum\directlua{if #1<#2 then tex.print(1) else tex.print(0) end}=1
   #3%
  \else
   #4%
  \fi
}
```

`\dtlifnumgt`

`\dtlifnumgt{`⟨*num1*⟩`}{`⟨*num2*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Does ⟨*true part*⟩ if ⟨*num1*⟩ >⟨*num2*⟩, otherwise does ⟨*false part*⟩.

```
\newcommand*{\dtlifnumgt}[4]{%
  \ifnum\directlua{if #1>#2 then tex.print(1) else tex.print(0) end}=1
   #3%
  \else
```

```
    #4%
  \fi
}
```

\dtlifnumopenbetween

\dtlifnumopenbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩.

```
\newcommand*{\dtlifnumopenbetween}[5]{%
  \ifnum\directlua{if #2 < #1 and #1 < #3 then tex.print(1) else tex.print(0) end}=1
    #4%
  \else
    #5%
  \fi
}
```

\dtlifnumclosedbetween

\dtlifnumclosedbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩.

```
\newcommand*{\dtlifnumclosedbetween}[5]{%
  \ifnum\directlua{if #2 <= #1 and #1 <= #3 then tex.print(1) else tex.print(0) end}=1
    #4%
  \else
    #5%
  \fi
}
```

7.2 Functions

\dtladd

\dtladd{⟨*cs*⟩}{⟨*num1*⟩}{⟨*num2*⟩}

Adds two numbers and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtladd{mmm}{%
  \edef#1{\directlua{ tex.print(#2+(#3)) }}%
}
```

\dtladdall

\dtladdall{⟨*cs*⟩}{⟨*num list*⟩}

Adds all numbers and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtladdall{mm}{%
  \edef#1{\directlua{
    x = 0;
```

```
    array = { #2 };
    for key,val in ipairs(array) do
     x = x + val;
    end
    tex.print(x);
  }}%
}
```

\dtlsub `\dtlsub{⟨cs⟩}{⟨num1⟩}{⟨num2⟩}`

Subtracts two numbers and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtlsub{mmm}{%
  \edef#1{\directlua{ tex.print(#2-(#3)) }}%
}
```

\dtlmul `\dtlmul{⟨cs⟩}{⟨num1⟩}{⟨num2⟩}`

Multiplies two numbers and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtlmul{mmm}{%
  \edef#1{\directlua{ tex.print(#2*#3) }}%
}
```

\dtldiv `\dtldiv{⟨cs⟩}{⟨num1⟩}{⟨num2⟩}`

Divides two numbers and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtldiv{mmm}{%
  \edef#1{\directlua{ tex.print(#2/#3) }}%
}
```

\dtlsqrt `\dtlsqrt{⟨cs⟩}{⟨num⟩}`

Square root number and store the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtlsqrt{mm}{%
  \edef#1{\directlua{ tex.print(math.sqrt(#2)) }}%
}
```

\dtlroot `\dtlroot{⟨cs⟩}{⟨num⟩}{⟨n⟩}`

Nth root number and store the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtlroot{mmm}{%
  \edef#1{\directlua{ tex.print((#2)^(1/(#3))) }}%
}
```

\dtlround

```
\dtlround{⟨cs⟩}{⟨num⟩}{⟨dp⟩}
```

Rounds ⟨*num*⟩ to ⟨*dp*⟩ decimal places and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtlround{mmm}{%
  \edef#1{"\@percentchar0.\number#3f", #2}%
  \edef#1{\directlua{ tex.print(string.format(#1)) }}%
}
```

\dtltrunc

```
\dtltrunc{⟨cs⟩}{⟨num⟩}{⟨dp⟩}
```

Truncates ⟨*num*⟩ to ⟨*dp*⟩ decimal places and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtltrunc{mmm}{%
 \edef#1{\directlua{
   local m = 10^(#3);
   tex.print(string.format("\@percentchar.#3f", math.floor(#2 * m)/m))
 }}%
}
```

\dtlclip

```
\dtlclip{⟨cs⟩}{⟨num⟩}
```

Removes redundant trailing zeros from ⟨*num*⟩ and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtlclip{mm}{%
 \edef#1{\directlua{
  local s = "#2";
  tex.print(s:match("^(\@percentchar d*\@percentchar.?0?\@percentchar d-
)0*$"))
 }}%
}
```

\dtlmin

```
\dtlmin{⟨cs⟩}{⟨num1⟩}{⟨num2⟩}
```

Defines ⟨*cs*⟩ to the smaller of the two numbers.

```
\NewDocumentCommand\dtlmin{mmm}{%
 \edef#1{\directlua{tex.print(math.min(#2,#3))}}%
}
```

\dtlminall

```
\dtlminall{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the minimum in the comma-separated list of numbers.

```
\NewDocumentCommand\dtlminall{mm}{%
 \edef#1{\directlua{tex.print(math.min(#2))}}%
}
```

\dtlmax

\dtlmax{⟨*cs*⟩}{⟨*num1*⟩}{⟨*num2*⟩}

Defines ⟨*cs*⟩ to the larger of the two numbers.

```
\NewDocumentCommand\dtlmax{mmm}{%
 \edef#1{\directlua{tex.print(math.max(#2,#3))}}%
}
```

\dtlmaxall

\dtlmaxall{⟨*cs*⟩}{⟨*num list*⟩}

Defines ⟨*cs*⟩ to the maximum in the comma-separated list of numbers.

```
\NewDocumentCommand\dtlmaxall{mm}{%
 \edef#1{\directlua{tex.print(math.max(#2))}}%
}
```

\dtlabs

\dtlabs{⟨*cs*⟩}{⟨*num*⟩}

Defines ⟨*cs*⟩ to the absolute value of ⟨*num*⟩.

```
\NewDocumentCommand\dtlabs{mm}{%
 \edef#1{\directlua{tex.print(math.abs(#2))}}%
}
```

\dtlneg

\dtlneg{⟨*cs*⟩}{⟨*num*⟩}

Defines ⟨*cs*⟩ to the negative of ⟨*num*⟩.

```
\NewDocumentCommand\dtlneg{mm}{%
 \edef#1{\directlua{tex.print(-(#2))}}%
}
```

\dtlmeanforall

\dtlmeanforall{⟨*cs*⟩}{⟨*num list*⟩}

Computes the mean and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand\dtlmeanforall{mm}{%
  \edef#1{\directlua{
    x = 0;
    n = 0;
    array = { #2 };
    for key,val in ipairs(array) do
     n = n + 1;
     x = x + val;
    end
    tex.print(x/n);
  }}%
}
```

\dtlvarianceforall

\dtlvarianceforall[⟨*mean*⟩]{⟨*cs*⟩}{⟨*num list*⟩}

Computes the variance and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand \dtlvarianceforall { o m m }
{%
  \IfNoValueTF{#1}%
  {%
     \edef#2{\directlua{
       n = 0;
       mean = 0;
       array = { #3 };
       for key,val in ipairs(array) do
        n = n + 1;
        mean = mean + val;
       end
       mean = mean / n;
       variance = 0;
       for key,val in ipairs(array) do
        x = val - mean;
        variance = variance + x * x;
       end
       variance = variance / n;
       tex.print(variance);
     }}%
  }%
  {%
     \edef#2{\directlua{
       n = 0;
       mean = #1;
       array = { #3 };
       variance = 0;
       for key,val in ipairs(array) do
        n = n + 1;
        x = val - mean;
        variance = variance + x * x;
       end
       variance = variance / n;
       tex.print(variance);
     }}%
  }%
}
```

\dtlsdforall

\dtlsdforall[⟨*mean*⟩]{⟨*cs*⟩}{⟨*num list*⟩}

Computes the standard deviation and stores the result in ⟨*cs*⟩.

```
\NewDocumentCommand \dtlsdforall { o m m }
{%
```

```
  \IfNoValueTF{#1}%
  {%
     \edef#2{\directlua{
       n = 0;
       mean = 0;
       array = { #3 };
       for key,val in ipairs(array) do
        n = n + 1;
        mean = mean + val;
       end
       mean = mean / n;
       variance = 0;
       for key,val in ipairs(array) do
        x = val - mean;
        variance = variance + x * x;
       end
       variance = variance / n;
       tex.print(math.sqrt(variance));
     }}%
  }%
  {%
     \edef#2{\directlua{
       n = 0;
       mean = #1;
       array = { #3 };
       variance = 0;
       for key,val in ipairs(array) do
        n = n + 1;
        x = val - mean;
        variance = variance + x * x;
       end
       variance = variance / n;
       tex.print(math.sqrt(variance));
     }}%
  }%
}
```

8 datatool-fp.def

Definitions of fixed-point commands that use the fp package. Provided for backward-compatibility.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{datatool-fp.def}[2026/07/31 v3.4.4 (NLCT)]
```

Required package:

```
\RequirePackage{fp}
```

If fp's verbose option set, switch on verbose for datatool-base as well:

```
\ifFPmessages
```

```
    \dtlverbosetrue
  \fi
```

8.1 Comparison Commands

\dtlifnumeq

\dtlifnumeq{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does ⟨*true part*⟩ if ⟨*num1*⟩=⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\newrobustcmd*{\dtlifnumeq}[4]{%
  \FPifeq{#1}{#2}%
    #3%
  \else
    #4%
  \fi
}
```

\dtlifnumlt

\dtlifnumlt{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does ⟨*true part*⟩ if ⟨*num1*⟩ <⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\newrobustcmd*{\dtlifnumlt}[4]{%
  \FPiflt{#1}{#2}%
   #3%
  \else
   #4%
  \fi
}
```

\dtlifnumgt

\dtlifnumgt{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does ⟨*true part*⟩ if ⟨*num1*⟩ >⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\newrobustcmd*{\dtlifnumgt}[4]{%
  \FPifgt{#1}{#2}%
   #3%
  \else
   #4%
  \fi
}
```

\dtlifnumopenbetween

\dtlifnumopenbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩ where all numerical arguments are in standard fixed point notation.

```
\newrobustcmd*{\dtlifnumopenbetween}[5]{%
  \let\@dtl@dovalue\relax
  \dtlifnumgt{#1}{#2}%
  {}%
  {%
    \def\@dtl@dovalue{#5}%
  }%
  \dtlifnumlt{#1}{#3}%
  {%
    \ifx\@dtl@dovalue\relax
      \def\@dtl@dovalue{#4}%
    \fi
  }%
  {%
    \def\@dtl@dovalue{#5}%
  }%
  \@dtl@dovalue
}
```

\dtlifnumclosedbetween

\dtlifnumclosedbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩ where all numerical arguments are in standard fixed point notation.

```
\newrobustcmd*{\dtlifnumclosedbetween}[5]{%
  \let\@dtl@dovalue\relax
  \dtlifnumgt{#1}{#2}%
  {}%
  {%
    \dtlifnumeq{#1}{#2}%
    {%
      \def\@dtl@dovalue{#4}%
    }%
    {%
      \def\@dtl@dovalue{#5}%
    }%
  }%
  \dtlifnumlt{#1}{#3}%
  {%
    \ifx\@dtl@dovalue\relax
      \def\@dtl@dovalue{#4}%
    \fi
```

```
    }%
    {%
      \dtlifnumeq{#1}{#3}%
      {%
        \def\@dtl@dovalue{#4}%
      }%
      {%
        \def\@dtl@dovalue{#5}%
      }%
    }%
    \@dtl@dovalue
  }
```

8.2 Functions

`\dtladd` Adds two numbers using fp.

```
\NewDocumentCommand{\dtladd}{mmm}{%
  \FPadd{#1}{#2}{#3}%
}
```

`\dtladdall`

```
\dtladdall{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the sum of all the numbers in the comma-separated list of numbers.

```
\NewDocumentCommand{\dtladdall}{mm}{%
  \def#1{0}%
  \@for\@dtl@tmp:=#2\do{%
    \FPadd{#1}{#1}{\@dtl@tmp}%
  }%
}
```

`\dtlsub` Subtracts two numbers using fp.

```
\NewDocumentCommand{\dtlsub}{mmm}{%
  \FPsub{#1}{#2}{#3}%
}
```

`\dtlmul` Multiplies two numbers using fp.

```
\NewDocumentCommand{\dtlmul}{mmm}{%
  \FPmul{#1}{#2}{#3}%
}
```

`\dtldiv` Divides two numbers using fp.

```
\NewDocumentCommand{\dtldiv}{mmm}{%
  \FPdiv{#1}{#2}{#3}%
}
```

`\dtlsqrt` Square root using fp.

```
\NewDocumentCommand{\dtlsqrt}{mm}{%
```

```
    \FProot{#1}{#2}{2}%
  }
```

`\dtlroot` Nth root using fp.

```
  \NewDocumentCommand{\dtlroot}{mmm}{%
    \FProot{#1}{#2}{#3}%
  }
```

`\dtlround` Rounds using fp.

```
  \NewDocumentCommand{\dtlround}{mmm}{%
    \FPround{#1}{#2}{#3}%
  }
```

`\dtltrunc` Truncates using fp. (Third argument is the number of digits.)

```
  \NewDocumentCommand{\dtltrunc}{mmm}{%
    \FPtrunc{#1}{#2}{#3}%
  }
```

`\dtlclip`

```
  \NewDocumentCommand{\dtlclip}{mm}{%
    \FPclip{#1}{#2}%
  }
```

`\dtlmin` Minimum of two numbers using fp.

```
  \NewDocumentCommand{\dtlmin}{mmm}{%
    \FPmin{#1}{#2}{#3}%
  }
```

`\dtlminall`

```
\dtlminall{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the minimum in the comma-separated list of numbers.

```
  \NewDocumentCommand{\dtlminall}{mm}{%
    \let#1\empty
    \@for\@dtl@tmp:=#2\do{%
      \ifx\empty
       \let#1\@dtl@tmp
      \else
       \FPmin#1{#1}{\@dtl@tmp}%
      \fi
    }%
  }
```

`\dtlmax` Maximum of two numbers using fp.

```
  \NewDocumentCommand{\dtlmax}{mmm}{%
    \FPmax{#1}{#2}{#3}%
  }
```

\dtlmaxall

```
\dtlmaxall{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the maximum in the comma-separated list of numbers.

```
\NewDocumentCommand{\dtlmaxall}{mm}{%
  \let#1\empty
  \@for\@dtl@tmp:=#2\do{%
    \ifx\empty
     \let#1\@dtl@tmp
    \else
     \FPmax#1{#1}{\@dtl@tmp}%
    \fi
  }%
}
```

\dtlabs Absolute value using fp.

```
\NewDocumentCommand{\dtlabs}{mm}{%
  \FPabs{#1}{#2}%
}
```

\dtlneg Negative of a value using fp.

```
\NewDocumentCommand{\dtlneg}{mm}{%
  \FPneg{#1}{#2}%
}
```

\dtlmeanforall

```
\dtlmeanforall{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the mean (average) of all the numbers in the comma-separated list of numbers.

```
\NewDocumentCommand{\dtlmeanforall}{mm}{%
  \def#1{0}%
  \count@=0\relax
  \@for\@dtl@tmp:=#2\do{%
    \advance\count@ by \@ne
    \FPadd{#1}{#1}{\@dtl@tmp}%
  }%
  \FPdiv{#1}{#1}{\number\count@}%
}
```

\dtlvarianceforall

```
\dtlvarianceforall[⟨mean⟩]{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the variance of all the numbers in the comma-separated list of numbers. If the mean has already been calculated it can be supplied in the optional argument.

```
\NewDocumentCommand{\dtlvarianceforall}{omm}{%
  \IfNoValueTF{#1}%
  {%
```

```
    \def\@dtl@mean{0}%
    \count@=0\relax
    \@for\@dtl@tmp:=#3\do{%
      \advance\count@ by \@ne
      \FPadd{\@dtl@mean}{\@dtl@mean}{\@dtl@tmp}%
    }%
    \FPdiv{\@dtl@mean}{\@dtl@mean}{\number\count@}%
    \def#2{0}%
    \@for\@dtl@tmp:=#3\do{%
      \FPsub{\@dtl@tmp}{\@dtl@tmp}{\@dtl@mean}%
      \FPmul{\@dtl@tmp}{\@dtl@tmp}{\@dtl@tmp}%
      \FPadd{#2}{#2}{\@dtl@tmp}%
    }%
    \FPdiv{#2}{#2}{\number\count@}%
  }%
  {%
    \def\@dtl@mean{#1}%
    \count@=0\relax
    \def#2{0}%
    \@for\@dtl@tmp:=#3\do{%
      \advance\count@ by \@ne
      \FPsub{\@dtl@tmp}{\@dtl@tmp}{\@dtl@mean}%
      \FPmul{\@dtl@tmp}{\@dtl@tmp}{\@dtl@tmp}%
      \FPadd{#2}{#2}{\@dtl@tmp}%
    }%
    \FPdiv{#2}{#2}{\number\count@}%
  }%
}
```

\dtlsdforall

```
\dtlsdforall[⟨mean⟩]{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the standard deviation of all the numbers in the comma-separated list of numbers. If the mean has already been calculated it can be supplied in the optional argument.

```
\NewDocumentCommand{\dtlsdforall}{omm}{%
  \IfNoValueTF{#1}%
  {%
    \def\@dtl@mean{0}%
    \count@=0\relax
    \@for\@dtl@tmp:=#3\do{%
      \advance\count@ by \@ne
      \FPadd{\@dtl@mean}{\@dtl@mean}{\@dtl@tmp}%
    }%
    \FPdiv{\@dtl@mean}{\@dtl@mean}{\number\count@}%
    \def#2{0}%
    \@for\@dtl@tmp:=#3\do{%
      \FPsub{\@dtl@tmp}{\@dtl@tmp}{\@dtl@mean}%
      \FPmul{\@dtl@tmp}{\@dtl@tmp}{\@dtl@tmp}%
```

```
      \FPadd{#2}{#2}{\@dtl@tmp}%
    }%
  }%
  {%
    \def\@dtl@mean{#1}%
    \count@=0\relax
    \def#2{0}%
    \@for\@dtl@tmp:=#3\do{%
      \advance\count@ by \@ne
      \FPsub{\@dtl@tmp}{\@dtl@tmp}{\@dtl@mean}%
      \FPmul{\@dtl@tmp}{\@dtl@tmp}{\@dtl@tmp}%
      \FPadd{#2}{#2}{\@dtl@tmp}%
    }%
  }%
  \FPdiv{#2}{#2}{\number\count@}%
  \FProot{#2}{#2}{2}%
}
```

9 datatool-fp.sty

Definitions of fixed-point commands that use the fp package. Note that the newer datatool-l3fp or datatool-lua are preferable.

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{datatool-fp-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{datatool-fp}[2026/07/31 v3.4.4 (NLCT)]
```

This package is deprecated.

```
\PackageWarning{datatool-fp}%
{datatool-fp.sty deprecated. Use
\string\usepackage[math=fp]{datatool} instead or
(better still) \string\usepackage{datatool}
}
\RequirePackage[math=fp]{datatool}
```

10 datatool-pgfmath.sty

Definitions of fixed-point commands that use the pgfmath package. Provided for backward-compatibility. The newer `datatool-l3fp.def` or `datatool-lua.def` are preferable.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{datatool-pgfmath.def}[2026/07/31 v3.4.4 (NLCT)]
```

Required packages:

```
\RequirePackage{pgfrcs,pgfkeys,pgfmath}
```

Old code has only been partially rewritten with LaTeX3.

```
\ExplSyntaxOn
```

10.1 Comparison Commands

\dtlifnumeq

```
\dtlifnumeq{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩=⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator. The `\number0` part allows an empty argument to be treated as zero. (`\number` required to prevent a zero prefix indicating an octal number.)

```
\NewDocumentCommand \dtlifnumeq { m m m m }
{
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
  \exp_args:Nx \pgfmathifthenelse {#1==#2}%
    {"\exp_not:N \@dtl@truepart"}{"\exp_not:N \@dtl@falsepart"}%
  \pgfmathresult
}
```

\dtlifnumlt

```
\dtlifnumlt{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩ <⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\NewDocumentCommand \dtlifnumlt { m m m m }
{
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
  \exp_args:Nx \pgfmathifthenelse {#1 < #2}%
    {"\exp_not:N \@dtl@truepart"}{"\exp_not:N \@dtl@falsepart"}%
  \pgfmathresult
}
```

\dtlifnumgt

```
\dtlifnumgt{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩ >⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\NewDocumentCommand \dtlifnumgt { m m m m }
{
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
```

```
    \exp_args:Nx \pgfmathifthenelse {#1 > #2}%
      {"\exp_not:N \@dtl@truepart"}{"\exp_not:N \@dtl@falsepart"}%
    \pgfmathresult
  }
```

\dtlifnumopenbetween

\dtlifnumopenbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩ where all numerical arguments are in standard fixed point notation.

```
\NewDocumentCommand \dtlifnumopenbetween { m m m m m }
{
  \dtlifnumlt { #1 } { #3 }
  { \dtlifnumgt { #1 } { #2 } { #4 } { #5 } }
  { #5 }
}
```

\dtlifnumclosedbetween

\dtlifnumclosedbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩ where all numerical arguments are in standard fixed point notation.

```
\NewDocumentCommand \dtlifnumclosedbetween { m m m m m }
{
  \dtlifnumlt { #1 } { #2 }
  { #5 }
  { \dtlifnumgt { #1 } { #3 } { #5 } { #4 } }
}
```

10.2 Functions

\dtladd Adds two numbers using PGF math engine.

```
\NewDocumentCommand \dtladd { m m m }
{
  \pgfmathadd{#2}{#3}%
  \let#1\pgfmathresult
}
```

\dtladdall

\dtladdall{⟨*cs*⟩}{⟨*num list*⟩}

Defines ⟨*cs*⟩ to the sum of all the numbers in the comma-separated list of numbers.

```
\NewDocumentCommand \dtladdall { m m }
{
  \tl_set:Nn \pgfmathresult { 0 }
```

```
    \exp_args:No \clist_map_inline:nn { #2 }
    {
      \pgfmathadd { \pgfmathresult } { ##1 }
    }
    \tl_set_eq:NN #1 \pgfmathresult
  }
```

`\dtlsub` Subtracts two numbers using PGF math engine.

```
\NewDocumentCommand \dtlsub { m m m }
{
  \pgfmathsubtract{#2}{#3}%
  \let#1\pgfmathresult
}
```

`\dtlmul` Multiplies two numbers using PGF math engine.

```
\NewDocumentCommand \dtlmul { m m m }
{%
  \pgfmathmultiply{#2}{#3}%
  \let#1\pgfmathresult
}
```

`\dtldiv` Divides two numbers using PGF math engine.

```
\NewDocumentCommand \dtldiv { m m m }
{%
  \pgfmathdivide{#2}{#3}%
  \let#1\pgfmathresult
}
```

`\dtlsqrt` Root using PGF math engine.

```
\NewDocumentCommand \dtlsqrt { m m }
{%
  \pgfmathsqrt{#2}%
  \let#1\pgfmathresult
}
```

`\dtlroot` Root using PGF math engine.

```
\NewDocumentCommand \dtlroot { m m m }
{
  \exp_args:Nx \tl_if_eq:nnTF { #3 } { 2 }
  {
    \exp_args:Nx \pgfmathsqrt { #2 }
    \let#1\pgfmathresult
  }
  {
    \exp_args:Nxx \pgfmathpow { #2 } { 1 / ( #3 ) }
    \let#1\pgfmathresult
  }
}
```

`\dtlround` Rounds using PGF math engine.

```
\NewDocumentCommand{\dtlround}{mmm}{%
  \ifnum#3=0\relax
    \exp_args:Nx \pgfmathparse { int( round ( #2 ) ) }
    \let#1\pgfmathresult
  \else
    \exp_args:Nx \pgfmathparse { int ( 10 \exp_not:N ^ #3 )}
    \let\dtl@tmpshift\pgfmathresult
```

Need to be careful not to trigger the dimension too large error, so this is a bit convoluted.

```
    \exp_args:Nx \pgfmathparse { int ( floor ( #2 ) ) }%
    \let\dtl@int@round\pgfmathresult
    \exp_args:Nx \pgfmathparse
      { int ( round ( ( #2 - \dtl@int@round ) * \dtl@tmpshift ) ) }
```

This bit is awkward because simply dividing by multiples of 10 in pgfmath can cause rounding errors, so need to employ another method.

```
    \@dtl@tmpcount=0\relax
    \expandafter\@dtl@countdigits\pgfmathresult.\relax
    \advance\@dtl@tmpcount by -#3\relax
    \def\@dtl@intpart{}%
    \def\@dtl@fracpart{}%
    \expandafter\@dtl@gatherintfrac\pgfmathresult\relax
    \tl_if_empty:NT \@dtl@intpart { \tl_set:Nn \@dtl@intpart { 0 } }
    \edef\@dtl@intpart{\number\numexpr\dtl@int@round + \@dtl@intpart}%
    \edef#1{\@dtl@intpart.\@dtl@fracpart}%
  \fi
}
```

`\@dtl@gatherintfrac`

```
\newcommand*{\@dtl@gatherintfrac}[1]{%
  \ifx\relax#1\relax
  \else
    \advance\@dtl@tmpcount by -1\relax
    \ifnum\@dtl@tmpcount<0\relax
      \edef\@dtl@fracpart{\@dtl@fracpart#1}%
    \else
      \edef\@dtl@intpart{\@dtl@intpart#1}%
    \fi
    \expandafter\@dtl@gatherintfrac
  \fi
}
```

`\dtltrunc` Truncates using PGF math engine. (Third argument is the number of digits.) This suffers from the same problems as `\dtlround`. Can cause dimension too large error or rounding errors.

```
\NewDocumentCommand \dtltrunc { m m m }
{
  \ifnum#3=0\relax
    \exp_args:Nx \pgfmathparse { int ( floor ( #2 ) ) }%
```

```
    \let#1\pgfmathresult
  \else
    \exp_args:Nx \pgfmathparse { int ( 10 \exp_not:N ^ #3 ) }
    \let\dtl@tmpshift\pgfmathresult
```

Need to be careful not to trigger the dimension too large error, so this is a bit convoluted.

```
    \exp_args:Nx \pgfmathparse { int ( floor ( #2 ) ) }
    \let\dtl@int@trunc\pgfmathresult
    \exp_args:Nx \pgfmathparse
     { int ( floor ( ( #2 - \dtl@int@trunc ) * \dtl@tmpshift ) ) }
```

This bit is awkward because simply dividing by multiples of 10 in pgfmath can cause rounding errors, so need to employ another method.

```
    \@dtl@tmpcount=0\relax
    \expandafter\@dtl@countdigits\pgfmathresult.\relax
    \advance\@dtl@tmpcount by -#3\relax
    \def\@dtl@intpart{}%
    \def\@dtl@fracpart{}%
    \expandafter\@dtl@gatherintfrac\pgfmathresult\relax
    \tl_if_empty:NT \@dtl@intpart { \tl_set:Nn \@dtl@intpart { 0 } }
    \edef\@dtl@intpart{\number\numexpr\dtl@int@trunc
      + \@dtl@intpart}%
    \edef#1{\@dtl@intpart.\@dtl@fracpart}%
  \fi
}
```

`\dtlclip` There isn't a clip in pgfmath so use LaTeX3 instead.

```
\NewDocumentCommand \dtlclip { m m }
{
  \fp_set:Nn { \l__datatool_tmpa_fp } { #2 }
  \tl_set:Nx #1 { \fp_use:N { \l__datatool_tmpa_fp } }
}
```

`\dtlmin` Minimum of two numbers using PGF math engine.

```
\NewDocumentCommand \dtlmin { m m m }
{
  \exp_args:Nxx \pgfmathmin { #2 } { #3 }
  \let#1\pgfmathresult
}
```

`\dtlminall` `\dtlminall{⟨cs⟩}{⟨num list⟩}`

Defines ⟨*cs*⟩ to the minimum in the comma-separated list of numbers.

```
\NewDocumentCommand \dtlminall { m m }
{
  \exp_args:Nx \pgfmathmin { #2 } { }
  \let#1\pgfmathresult
}
```

`\dtlmax` Maximum of two numbers using PGF math engine.

```
\NewDocumentCommand \dtlmax { m m m }
{
  \exp_args:Nx \pgfmathmax { #2 } { #3 }
  \let#1\pgfmathresult
}
```

`\dtlmaxall`

```
\dtlmaxall{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the maximum in the comma-separated list of numbers.

```
\NewDocumentCommand \dtlmaxall { m m }
{
  \exp_args:Nx \pgfmathmax { #2 } { }
  \let#1\pgfmathresult
}
```

`\dtlabs` Absolute value using PGF math engine.

```
\NewDocumentCommand \dtlabs { m m }
{
  \exp_args:Nx \pgfmathabs { #2 }
  \let#1\pgfmathresult
}
```

`\dtlneg` Negative of a value using PGF math engine.

```
\NewDocumentCommand \dtlneg { m m }
{
  \exp_args:Nx \pgfmathneg { #2 }
  \let#1\pgfmathresult
}
```

`\dtlmeanforall`

```
\dtlmeanforall{⟨cs⟩}{⟨num list⟩}
```

Defines ⟨*cs*⟩ to the mean of all the numbers in the comma-separated list of numbers.

```
\NewDocumentCommand \dtlmeanforall { m m }
{
  \tl_set:Nn \pgfmathresult { 0 }
  \int_zero:N \l__datatool_count_int
  \exp_args:No \clist_map_inline:nn { #2 }
  {
    \int_incr:N \l__datatool_count_int
    \exp_args:Nxx \pgfmathadd { \pgfmathresult } { ##1 }
  }
  \exp_args:Nxx \pgfmathdivide
    { \pgfmathresult }
    { \int_use:N \l__datatool_count_int }
  \tl_set_eq:NN #1 \pgfmathresult
}
```

\dtlvarianceforall

\dtlvarianceforall[⟨*mean*⟩]{⟨*cs*⟩}{⟨*num list*⟩}

Defines ⟨*cs*⟩ to the variance of all the numbers in the comma-separated list of numbers. If the mean has already been computed it can be supplied in the optional argument.

```
\NewDocumentCommand {\dtlvarianceforall} { o m m }
{%
  \IfNoValueTF { #1 }
  {
    \tl_set:Nn \pgfmathresult { 0 }
    \int_zero:N \l__datatool_count_int
    \exp_args:No \clist_map_inline:nn { #3 }
    {
      \int_incr:N \l__datatool_count_int
      \exp_args:Nxx \pgfmathadd { \pgfmathresult } { ##1 }
    }
    \exp_args:Nxx \pgfmathdivide
      { \pgfmathresult }
      { \int_use:N \l__datatool_count_int }
    \tl_set_eq:NN \@dtl@mean \pgfmathresult
    \tl_set:Nn #2 { 0 }
    \exp_args:No \clist_map_inline:nn { #3 }
    {
      \exp_args:Nxx \pgfmathsubtract { ##1 } { \@dtl@mean }
      \exp_args:Nxx \pgfmathmultiply { \pgfmathresult } { \pgfmathresult }
      \exp_args:Nxx \pgfmathadd { #2 } { \pgfmathresult }
      \tl_set_eq:NN #2 \pgfmathresult
    }
  }
  {
    \tl_set:Nx \@dtl@mean { #1 }
    \int_zero:N \l__datatool_count_int
    \tl_set:Nn #2 { 0 }
    \exp_args:No \clist_map_inline:nn { #3 }
    {
      \int_incr:N \l__datatool_count_int
      \exp_args:Nxx \pgfmathsubtract { ##1 } { \@dtl@mean }
      \exp_args:Nxx \pgfmathmultiply
       { \pgfmathresult } { \pgfmathresult }
      \exp_args:Nxx \pgfmathadd { #2 } { \pgfmathresult }
      \tl_set_eq:NN #2 \pgfmathresult
    }
  }
  \exp_args:Nxx \pgfmathdivide
    { #2 }
    { \int_use:N \l__datatool_count_int }
  \tl_set_eq:NN #2 \pgfmathresult
}
```

\dtlsdforall

\dtlsdforall[⟨*mean*⟩]{⟨*cs*⟩}{⟨*num list*⟩}

Defines ⟨*cs*⟩ to the standard deviation of all the numbers in the comma-separated list of numbers. If the mean has already been computed it can be supplied in the optional argument.

```
\NewDocumentCommand{\dtlsdforall} { o m m }
{%
  \IfNoValueTF{#1}%
  {%
    \tl_set:Nn \pgfmathresult { 0 }
    \int_zero:N \l__datatool_count_int
    \exp_args:No \clist_map_inline:nn { #3 }
    {
      \int_incr:N \l__datatool_count_int
      \exp_args:Nxx \pgfmathadd { \pgfmathresult } { ##1 }
    }
    \exp_args:Nxx \pgfmathdivide
      { \pgfmathresult }
      { \int_use:N \l__datatool_count_int }
    \tl_set_eq:NN \@dtl@mean \pgfmathresult
    \tl_set:Nn #2 { 0 }
    \exp_args:No \clist_map_inline:nn { #3 }
    {
      \exp_args:Nxx \pgfmathsubtract { ##1 } { \@dtl@mean }
      \exp_args:Nxx \pgfmathmultiply { \pgfmathresult } { \pgfmathresult }
      \exp_args:Nxx \pgfmathadd { #2 } { \pgfmathresult }
      \tl_set_eq:NN #2 \pgfmathresult
    }
  }
  {
    \tl_set:Nn \@dtl@mean { #1 }
    \int_zero:N \l__datatool_count_int
    \tl_set:Nn #2 { 0 }
    \exp_args:No \clist_map_inline:nn { #3 }
    {
      \int_incr:N \l__datatool_count_int
      \exp_args:Nxx \pgfmathsubtract { ##1 } { \@dtl@mean }
      \exp_args:Nxx \pgfmathmultiply
        { \pgfmathresult } { \pgfmathresult }
      \exp_args:Nxx \pgfmathadd { #2 } { \pgfmathresult }
      \tl_set_eq:NN #2 \pgfmathresult
    }
  }
  \exp_args:Nxx \pgfmathdivide
     { #2 } { \int_use:N \l__datatool_count_int }
  \exp_args:Nx \pgfmathsqrt { \pgfmathresult }
  \tl_set_eq:NN #2 \pgfmathresult
}
```

```
\ExplSyntaxOff
```

11 datatool-pgfmath.sty

Definitions of fixed-point commands that use the pgfmath package. Note that the newer datatool-l3fp or datatool-lua are preferable.

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{datatool-pgfmath-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{datatool-pgfmath}[2026/07/31 v3.4.4 (NLCT)]
```

This package is deprecated.

```
\PackageWarning{datatool-pgfmath}%
{datatool-pgfmath.sty deprecated. Use
\string\usepackage[math=pgfmath]{datatool} instead or
(better still) \string\usepackage{datatool}
}
\RequirePackage[math=pgfmath]{datatool}
```

12 datatool.sty

12.1 Package Declaration

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{datatool-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{datatool}[2026/07/31 v3.4.4 (NLCT)]
```

Version 3.0: no longer using xkeyval. Load required packages:

```
\RequirePackage{ifthen}
\RequirePackage{xfor}
\RequirePackage{etoolbox}
\RequirePackage{tracklang}
```

Version 3.0: partial rewrite using LaTeX3 (mainly for the CSV file parsing) so there's a mix of new and old style commands. This rewrite needs to take into account the format of dbtex files, which can be read and written by datatooltk. There are three file format versions: 1, 2 and 3. The first two are almost identical. Version 2 has the final line that defines `\dtllastloadeddb`. The internal commands are defined to allow a fast load that doesn't require any parsing to to split on separators or to determine data types. This means that the internal structure of the database needs to remain the same. Also, it's not possible to enable LaTeX3 syntax in the dbtex file because the category

code change will affect the data (the space character in particular). The version 3 file format is new to datatool v3.0. An undefined command on loading a dbtex will likely indicate that a newer dbtex is being used with an old datatool version and so should be considered incompatible.

```
\ExplSyntaxOn
```

Scratch variable that may be needed before datatool-base is loaded, but take into account that datatool-base may have already been loaded (for example, implicitly by glossaries).

```
\tl_clear_new:N \l__datatool_tmpa_tl
```

Token list to hold current index or empty if no current index.

```
\tl_new:N \DTLcurrentindex
```

Temporary variable for database name.

```
\tl_new:N \@dtl@dbname
```

12.2 Package Options

```
\cs_new:Nn \__datatool_char_to_hex:nN
 {
   \tl_set:Nx #2 { \int_to_hex:n { `#1 } }
 }
```

`\@dtl@separator` The data separator character (comma by default) is stored in `\@dtl@separator`. This is the separator used in external data files, not in the LaTeX code, which always uses a comma separator.

```
\newcommand*{\@dtl@separator}{,}
```

`\DTLsetseparator` `\DTLsetseparator{`⟨*char*⟩`}`

The sets `\@dtl@separator`, and constructs the relevant macros that require this character to be hardcoded into their definition.

```
\NewDocumentCommand \DTLsetseparator { m }
{
  \tl_if_single_token:nTF { #1 }
  {
    \tl_set:Nx \@dtl@separator { \tl_to_str:n { #1 } }
    \__datatool_char_to_hex:nN { #1 } \l__datatool_tmpa_tl
    \exp_args:NNx \regex_set:Nn \l__datatool_blank_row_regex
      {
        \exp_not:N \A
        \exp_not:N \cO (?: \exp_not:N \x { \l__datatool_tmpa_tl } ) *
        \exp_not:N \Z
      }
  }
  {
    \PackageError{datatool}{Separator ~ must ~ be ~ a ~ single ~
    token ~ (found ~ \tl_count_tokens:n { #1 } ~ tokens) }{}
  }
```

```
}
\regex_new:N \l__datatool_blank_row_regex
\DTLsetseparator{,}
\ExplSyntaxOff

\begingroup
\catcode`\^^I12
```

`\DTLsettabseparator` `\DTLsettabseparator` makes it easier to set a tab separator.

```
\gdef\DTLsettabseparator{%
  \catcode`\^^I12
  \DTLsetseparator{^^I}%
}
```

`\DTLmaketabspace`

```
\gdef\DTLmaketabspace{%
  \catcode`\^^I10\relax
}
\endgroup
```

`\@dtl@delimiter` The data delimiter character (double quote by default) is stored in `\@dtl@delimiter`. This is used in external data files, not in the LaTeX code.

```
\newcommand{\@dtl@delimiter}{"}

\ExplSyntaxOn
```

This may have already been defined if datatool was loaded first.

```
\tl_clear_new:N \l__datatool_tmpa_tl
```

`\DTLsetdelimiter`

`\DTLsetdelimiter{⟨char⟩}`

This sets the delimiter.

```
\NewDocumentCommand \DTLsetdelimiter { m }
{
  \tl_if_single_token:nTF { #1 }
  {
    \tl_set:Nx \@dtl@delimiter { \tl_to_str:n { #1 } }
    \__datatool_char_to_hex:nN { #1 } \l__datatool_tmpa_tl
    \exp_args:NNx \regex_set:Nn \l__datatool_delim_left_regex
      {
        \exp_not:N \A \exp_not:N \s *
        \exp_not:N \cO (?: \exp_not:N \x { \l__datatool_tmpa_tl } )
      }
    \exp_args:NNx \regex_set:Nn \l__datatool_delim_both_regex
      {
        \exp_not:N \A \exp_not:N \s *
        \exp_not:N \cO (?: \exp_not:N \x { \l__datatool_tmpa_tl } )
        ( .* )
```

```
    \exp_not:N \cO (?: \exp_not:N \x { \l__datatool_tmpa_tl } )
    \exp_not:N \s * \exp_not:N \Z
  }
\exp_args:NNx \regex_set:Nn \l__datatool_delim_right_regex
  {
    \exp_not:N \cO (?: \exp_not:N \x { \l__datatool_tmpa_tl } )
    \exp_not:N \s * \exp_not:N \Z
  }
\exp_args:NNx \regex_set:Nn \l__datatool_escape_delim_bksl_regex
  {
    ( \exp_not:N \\ | \exp_not:N \cO ( \exp_not:N \x { \l__datatool_tmpa_tl } ) )
  }
\exp_args:NNx \regex_set:Nn \l__datatool_escape_delim_regex
  {
    ( \exp_not:N \cO ( \exp_not:N \x { \l__datatool_tmpa_tl } ) )
  }
\exp_args:NNx \regex_set:Nn \l__datatool_unescape_str_delim_bksl_regex
  {
     \exp_not:N \cO (?: \exp_not:N \x { 5c } )
     ( \exp_not:N \cO
       ( \exp_not:N \x { \l__datatool_tmpa_tl } | \exp_not:N \x { 5c } )
     )
  }
\exp_args:NNx \regex_set:Nn \l__datatool_unescape_str_double_delim_regex
  {
      \exp_not:N \cO (?: \exp_not:N \x { \l__datatool_tmpa_tl } )
     ( \exp_not:N \cO ( \exp_not:N \x { \l__datatool_tmpa_tl } ) )
  }
\exp_args:NNx \regex_set:Nn \l__datatool_unescape_cs_double_delim_regex
  {
     (
       \exp_not:N \c
       {
         ( \exp_not:N \x { \l__datatool_tmpa_tl } )
       }
     )
     \exp_not:N \x { \l__datatool_tmpa_tl }
  }
\exp_args:NNx \regex_set:Nn \l__datatool_unescape_str_delim_regex
  {
     \exp_not:N \cO (?: \exp_not:N \x { 5c } )
     ( \exp_not:N \cO ( \exp_not:N \x { \l__datatool_tmpa_tl } ))
  }
\exp_args:NNx \regex_set:Nn \l__datatool_unescape_cs_delim_regex
  {
     \exp_not:N \c
     {
       ( \exp_not:N \x { \l__datatool_tmpa_tl } )
     }
  }
```

```
  }
  {
    \PackageError{datatool}{Delimiter ~ must ~ be ~ a ~ single ~
    token ~ (found ~ \tl_count_tokens:n { #1 } ~ tokens) }{}
  }
}
\regex_new:N \l__datatool_delim_left_regex
\regex_new:N \l__datatool_delim_right_regex
\regex_new:N \l__datatool_delim_both_regex
\regex_new:N \l__datatool_escape_delim_regex
\regex_new:N \l__datatool_escape_delim_bksl_regex
\regex_new:N \l__datatool_unescape_str_delim_regex
\regex_new:N \l__datatool_unescape_str_delim_bksl_regex
\regex_new:N \l__datatool_unescape_cs_delim_regex
\regex_new:N \l__datatool_unescape_str_double_delim_regex
\regex_new:N \l__datatool_unescape_cs_double_delim_regex
```

Initialise:

```
\DTLsetdelimiter{"}
```

Default database name.

```
\tl_new:N \l__datatool_default_dbname_tl
\tl_set:Nn \l__datatool_default_dbname_tl { untitled }
```

Determine whether to use the local or global function.

```
\bool_new:N \l__datatool_db_global_bool
\bool_set_true:N \l__datatool_db_global_bool
```

Determine whether or not to trim spaces around new entries.

```
\bool_new:N \l__datatool_new_element_trim_bool
\bool_set_true:N \l__datatool_new_element_trim_bool
```

Version 3.0: provide boolean to check on the current new value expansion setting:

```
\bool_new:N \l__datatool_new_element_expand_bool
```

Determine whether or not to expand values before adding them to the database.

`\dtlexpandnewvalue` Expand new value before adding to database

```
\newcommand*{\dtlexpandnewvalue}{
  \bool_set_true:N \l__datatool_new_element_expand_bool
  \def\@dtl@setnewvalue##1{
    \protected@edef\@dtl@tmp{ ##1 }
    \bool_if:NT\l__datatool_new_element_trim_bool
    {
      \tl_trim_spaces:N \@dtl@tmp
    }
    \expandafter\@dtl@toks\expandafter{\@dtl@tmp}
  }
}
```

`\dtlnoexpandnewvalue` Don't expand new value before adding to database

```
\newcommand*{\dtlnoexpandnewvalue}{%
  \bool_set_false:N \l__datatool_new_element_expand_bool
  \def\@dtl@setnewvalue##1{
    \bool_if:NTF\l__datatool_new_element_trim_bool
    {
      \tl_set:Nn \l__datatool_item_value_tl { ##1 }
      \tl_trim_spaces:N \l__datatool_item_value_tl
      \exp_args:NV \@dtl@toks \l__datatool_item_value_tl
    }
    {
      \@dtl@toks{##1}
    }
  }
}
```

`\@dtl@setnewvalue` Do this by default. This will define `\@dtl@setnewvalue`.

```
\dtlnoexpandnewvalue
```

Note that `\__datatool_process_new_value:n` should be used to preprocess values. That uses `\dtl@setnewvalue` to follow the trim and expansion settings but then checks the store datum boolean. TODO remove `\@dtl@setnewvalue` and replace with another boolean?

Determine whether or not to store values in the database in datum format.

```
\bool_new:N \l__datatool_db_store_datum_bool
```

Define options. The default-name value is expanded for consistency because it will be expanded if used as a package option so it should therefore also expand when used in `\DTLsetup`.

```
\keys_define:nn { datatool }
{
   default-name .str_set_x:N = \l__datatool_default_dbname_tl,
   global .bool_set:N = \l__datatool_db_global_bool,
   store-datum .bool_set:N = \l__datatool_db_store_datum_bool,
   separator .code:n = { \DTLsetseparator { #1 } },
   delimiter .code:n = { \DTLsetdelimiter { #1 } },
   new-value-trim .bool_set:N = \l__datatool_new_element_trim_bool,
   new-value-expand .choice:,
   new-value-expand / true .code:n = { \dtlexpandnewvalue },
   new-value-expand / false .code:n = { \dtlnoexpandnewvalue },
   new-value-expand .default:n = true,
}
\ExplSyntaxOff
```

If datatool-base may have already been loaded by another package, in which case the options need processing now.

```
\IfPackageLoadedTF{datatool-base}
{
  \ProcessKeyOptions[datatool]
```

```
}
{
```

The datatool-base package hasn't been loaded, so pass all options to that.

```
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datatool-
base}}
  \ProcessOptions
```

Load base package:

```
  \RequirePackage{datatool-base}
}
```

```
\ExplSyntaxOn
```

Private variables.

Token lists:

```
\tl_new:N \l__datatool_item_key_tl
\tl_new:N \l__datatool_item_type_tl
\tl_new:N \l__datatool_item_head_tl
\tl_new:N \l__datatool_item_value_tl
\tl_new:N \l__datatool_item_currency_tl
\tl_new:N \l__datatool_item_currency_ii_tl
\tl_new:N \l__datatool_row_tl
\tl_new:N \l__datatool_keydata_tl
\tl_new:N \l__datatool_content_tl
\tl_new:N \l__datatool_align_tl
\tl_new:N \l__datatool_before_tl
\tl_new:N \l__datatool_after_tl
\tl_new:N \l__datatool_cs_builder_tl
```

Numeric values that may be set to null to need a token list variable.

```
\tl_new:N \l__datatool_item_col_tl
```

Replaces `\dtl@rowidx`:

```
\tl_new:N \l__datatool_row_idx_tl
```

Integers:

```
\int_new:N \l__datatool_max_cols_int
\int_new:N \l__datatool_col_idx_int
\int_new:N \l__datatool_row_idx_int
\int_new:N \l__datatool_item_type_int
```

Sequences:

```
\seq_new:N \l__datatool_column_indexes_seq
\seq_new:N \l__datatool_column_keys_seq
```

Floating point:

```
\fp_new:N \l__datatool_min_fp
\fp_new:N \l__datatool_min_ii_fp
\fp_new:N \l__datatool_max_fp
\fp_new:N \l__datatool_max_ii_fp
\fp_new:N \l__datatool_total_ii_fp
```

Temporary list:

```
\clist_new:N \l__datatool_tmpa_clist
```

\DTLpar Many of the commands used by this package are short commands. This means that you can't use \par in the data. This command needs to be robust so it doesn't get expanded when written to a file. We also can't just use a synonym for \@@par because it may be used in a context where \par has a different meaning to \@@par.

```
\newrobustcmd{\DTLpar}{\par}
```

\DTLaction

\DTLaction[⟨*options*⟩]{⟨*action*⟩}

General purpose action command to avoid typing long command names. Note that since some commands may only have a local effect, it's not possible to include grouping in \DTLaction (although some of the underlying commands may introduce grouping). Therefore the options need to be reset at the start, and so there's no provision for use of those options in \DTLsetup.

```
\NewDocumentCommand \DTLaction { o m }
{
```

Initialise settings. Column index:

```
  \int_zero:N \l__datatool_action_column_int
  \int_zero:N \l__datatool_action_column_ii_int
```

Column key:

```
  \tl_clear:N \l__datatool_action_key_tl
  \tl_clear:N \l__datatool_action_key_ii_tl
```

Data type:

```
  \int_set_eq:NN
    \l__datatool_action_type_int
    \c_datatool_unknown_int
```

Row index:

```
  \int_zero:N \l__datatool_action_row_int
  \int_zero:N \l__datatool_action_row_ii_int
```

Value:

```
  \tl_set:Nn \l__datatool_action_value_tl { \q_no_value }
```

List settings:

```
  \clist_clear:N \l__datatool_action_options_clist
  \clist_clear:N \l__datatool_action_assign_clist
  \clist_clear:N \l__datatool_action_keys_clist
  \clist_clear:N \l__datatool_action_columns_clist
  \seq_clear:N \l__datatool_action_columns_seq
  \seq_clear:N \l__datatool_action_names_seq
```

Return formatting:

```
  \bool_set_false:N \l__datatool_action_datum_bool
  \int_set:Nn \l__datatool_action_datum_round_int { -1 }
```

```
\bool_set_true:N \l__datatool_action_datum_locale_decimal_bool
\bool_set_false:N \l__datatool_action_datum_locale_integer_bool
\tl_set:Nn \l__datatool_action_datum_currency_tl
 {  \l__datatool_item_currency_tl }
```

Return Values:

```
\clist_clear:N \l__datatool_action_return_clist
\tl_set_eq:NN \l__datatool_action_return_tl \dtlnovalue
\prop_clear:N \l__datatool_action_return_prop
\IfValueT { #1 }
  { \keys_set:nn { datatool/action } { #1 } }
```

Get database name for actions that require a single name:

```
\seq_if_empty:NTF \l__datatool_action_names_seq
 {
```

Name not set so use default name.

```
    \tl_set_eq:NN
      \l__datatool_action_name_tl
      \l__datatool_default_dbname_tl
 }
 {
```

Set name to first one in the list.

```
    \tl_set:Nx \l__datatool_action_name_tl
     {
       \seq_item:Nn \l__datatool_action_names_seq
         { \c_one_int }
     }
 }
```

Save action name:

```
\tl_set:Nx \l__datatool_action_tl { \tl_trim_spaces:n { #2 } }
```

Do action:

```
\cs_if_exist_use:cF { __datatool_action_ \l__datatool_action_tl : }
{
  \PackageError { datatool }
  { Unknown ~ action ~ ` \l__datatool_action_tl ' }
  { The ~ action ~ in ~ the ~ mandatory ~ argument ~ of ~
    \token_to_str:N \DTLaction \c_space_tl ~ is ~ not ~ recognised }
}
```

Assign return values to provided token list variables, if set.

```
\clist_if_empty:NF \l__datatool_action_return_clist
 {
   \keyval_parse:NNV
     \__datatool_cskey_missing_val:n
     \__datatool_action_get:nn
     \l__datatool_action_return_clist
 }
}
```

Internal variables for `\DTLaction`:

```
\tl_new:N \l__datatool_action_tl
\tl_new:N \l__datatool_action_name_tl
\tl_new:N \l__datatool_action_key_tl
\tl_new:N \l__datatool_action_key_ii_tl
\int_new:N \l__datatool_action_column_int
\int_new:N \l__datatool_action_column_ii_int
\int_new:N \l__datatool_action_row_int
\int_new:N \l__datatool_action_row_ii_int
\int_new:N \l__datatool_action_type_int
\int_set_eq:NN \l__datatool_action_type_int \c_datatool_unknown_int
\tl_new:N \l__datatool_action_value_tl
\tl_set:Nn \l__datatool_action_value_tl { \q_no_value }
\bool_new:N \l__datatool_action_datum_bool
\seq_new:N \l__datatool_action_names_seq
\clist_new:N \l__datatool_action_options_clist
\clist_new:N \l__datatool_action_assign_clist
\clist_new:N \l__datatool_action_return_clist
\clist_new:N \l__datatool_action_keys_clist
\clist_new:N \l__datatool_action_columns_clist
\seq_new:N \l__datatool_action_columns_seq
```

Scratch variables.

```
\tl_new:N \l__datatool_action_tmpa_tl
\tl_new:N \l__datatool_action_tmpb_tl
\seq_new:N \l__datatool_action_tmp_options_seq
\seq_new:N \l__datatool_action_tmp_data_seq
```

Define keys. (These can only be set in the optional argument of `\DTLaction`.)

```
\keys_define:nn { datatool/action }
{
  name .code:n =
   {
     \exp_args:NNx \seq_set_from_clist:Nn
       \l__datatool_action_names_seq { #1 }
   },
  name .value_required:n = true,
  key .str_set_x:N = \l__datatool_action_key_tl,
  key .value_required:n = true,
  key2 .str_set_x:N = \l__datatool_action_key_ii_tl,
  key2 .value_required:n = true,
  column .int_set:N = \l__datatool_action_column_int,
  column .value_required:n = true,
  column2 .int_set:N = \l__datatool_action_column_ii_int,
  column2 .value_required:n = true,
  row .int_set:N = \l__datatool_action_row_int,
  row .value_required:n = true,
  row2 .int_set:N = \l__datatool_action_row_ii_int,
  row2 .value_required:n = true,
  value .tl_set:N = \l__datatool_action_value_tl,
```

```
  value .value_required:n = true,
  expand-value .tl_set_x:N = \l__datatool_action_value_tl,
  expand-value .value_required:n = true,
  expand-once-value .code:n =
   { \tl_set:No \l__datatool_action_value_tl { #1 } },
  expand-once-value .value_required:n = true,
  options .clist_set:N = \l__datatool_action_options_clist,
  options .value_required:n = true,
  assign .clist_set:N = \l__datatool_action_assign_clist,
  assign .value_required:n = true,
  keys .clist_set:N = \l__datatool_action_keys_clist,
  keys .value_required:n = true,
  columns .clist_set:N = \l__datatool_action_columns_clist,
  columns .value_required:n = true,
  return .clist_set:N = \l__datatool_action_return_clist,
  return .value_required:n = true,
  type .choice:,
  type .value_required:n = true,
  type / string .code:n =
    {
      \int_set_eq:NN \l__datatool_action_type_int \c_datatool_string_int
    },
  type / integer .code:n =
    {
      \int_set_eq:NN \l__datatool_action_type_int \c_datatool_integer_int
    },
```

Synonym:

```
  type / int .code:n =
    {
      \int_set_eq:NN \l__datatool_action_type_int \c_datatool_integer_int
    },
  type / decimal .code:n =
    {
      \int_set_eq:NN \l__datatool_action_type_int \c_datatool_decimal_int
    },
```

Synonym:

```
  type / real .code:n =
    {
      \int_set_eq:NN \l__datatool_action_type_int \c_datatool_real_int
    },
  type / currency .code:n =
    {
      \int_set_eq:NN \l__datatool_action_type_int \c_datatool_currency_int
    },
  type / datetime .code:n =
    {
      \int_set_eq:NN \l__datatool_action_type_int \c_datatool_datetime_int
    },
```

Synonym:

```
   type / timestamp .code:n =
     {
       \int_set_eq:NN \l__datatool_action_type_int \c_datatool_datetime_int
     },
   type / date .code:n =
     {
       \int_set_eq:NN \l__datatool_action_type_int \c_datatool_date_int
     },
   type / time .code:n =
     {
       \int_set_eq:NN \l__datatool_action_type_int \c_datatool_time_int
     },
   datum .code:n =
    {
      \tl_if_eq:nnTF { #1 } { false }
       {
         \bool_set_false:N \l__datatool_action_datum_bool
       }
       {
         \bool_set_true:N \l__datatool_action_datum_bool
         \tl_if_eq:nnF { #1 } { true }
         {
            \keys_set:nn { datatool/action/datum } { #1 }
         }
       }
    },
   datum .default:n = true ,
 }
```

Sub keys for datum option:

```
 \int_new:N \l__datatool_action_datum_round_int
 \int_set:Nn \l__datatool_action_datum_round_int { -1 }
 \bool_new:N \l__datatool_action_datum_locale_decimal_bool
 \bool_set_true:N \l__datatool_action_datum_locale_decimal_bool
 \bool_new:N \l__datatool_action_datum_locale_integer_bool
 \tl_new:N \l__datatool_action_datum_currency_tl
 \tl_set:Nn \l__datatool_action_datum_currency_tl
  {  \l__datatool_item_currency_tl }
```

Define keys:

```
 \keys_define:nn { datatool/action/datum }
 {
   round .code:n =
    {
      \tl_if_eq:nnTF { #1 } { false }
       {
         \int_set:Nn \l__datatool_action_datum_round_int { -1 }
       }
       {
```

```
        \int_set:Nn \l__datatool_action_datum_round_int { #1 }
      }
   } ,
  round .default:n = 0 ,
  locale-decimal .bool_set:N =
    \l__datatool_action_datum_locale_decimal_bool ,
  locale-integer .bool_set:N =
    \l__datatool_action_datum_locale_integer_bool ,
  currency .choice: ,
  currency / false .code:n =
   {
     \tl_clear:N \l__datatool_action_datum_currency_tl
   } ,
  currency / match .code:n =
   {
     \tl_set:Nn
       \l__datatool_action_datum_currency_tl
       { \l__datatool_item_currency_tl }
   } ,
  currency / default .code:n =
   {
     \tl_set:Nn
       \l__datatool_action_datum_currency_tl
       { \@dtl@currency }
   } ,
  currency / unknown .code:n =
   {
     \tl_set:Nn
       \l__datatool_action_datum_currency_tl
       { #1 }
   } ,
  currency .default:n = default ,
}
```

Return value(s):

```
\tl_new:N \l__datatool_action_return_tl
\tl_set_eq:NN \l__datatool_action_return_tl \dtlnovalue
\tl_new:N \l__datatool_action_secondary_tl
\prop_new:N \l__datatool_action_return_prop
```

The datum option only governs the secondary return results. The primary return result isn't converted.

Secondary return result is a string:

```
\cs_new:Nn \__datatool_put_return_action_string:nn
{
  \tl_if_head_eq_meaning:nNTF
    { #2 } \__datatool_datum:w
  {
```

In weird format. Needs conversion.

```
    \prop_put:Nnx \l__datatool_action_return_prop
```

```
      { #1 } { #2 }
  }
  {
    \tl_if_head_eq_meaning:nNTF
      { #2 }  \__datatool_datum:nnnn
    {
```
Already in datum format. Leave it as is.
```
        \prop_put:Nnn \l__datatool_action_return_prop
         { #1 } { #2 }
    }
    {
       \bool_if:NTF \l__datatool_action_datum_bool
        {
          \prop_put:Nnn \l__datatool_action_return_prop
           { #1 }
           {
             \__datatool_datum:nnnn
               { #2 } { } { } { \c_datatool_string_int }
           }
        }
        {
          \prop_put:Nnn \l__datatool_action_return_prop
           { #1 } { #2 }
        }
     }
   }
}
\cs_generate_variant:Nn \__datatool_put_return_action_string:nn
 { nV, nv, nx, xV }
```
Secondary return result is an integer:
```
\cs_new:Nn \__datatool_put_return_action_int:nn
{
  \bool_if:NTF \l__datatool_action_datum_bool
   {
     \bool_if:NTF \l__datatool_action_datum_locale_integer_bool
      {
        \DTLdecimaltolocale { #2 }
          \l__datatool_action_secondary_tl
        \prop_put:Nnx \l__datatool_action_return_prop
         { #1 }
         {
           \exp_not:N \__datatool_datum:nnnn
             { \l__datatool_action_secondary_tl }
             { #2 }
             { }
             { \exp_not:N \c_datatool_integer_int }
         }
      }
      {
```

```
        \tl_if_head_eq_meaning:nNTF
          { #2 } \__datatool_datum:w
        {
```

In weird format. Needs conversion.

```
          \prop_put:Nnx \l__datatool_action_return_prop
           { #1 } { #2 }
        }
        {
          \tl_if_head_eq_meaning:nNTF
            { #2 } \__datatool_datum:nnnn
           {
```

Already in datum format. Leave it as is.

```
             \prop_put:Nnn \l__datatool_action_return_prop
              { #1 } { #2 }
           }
           {
             \prop_put:Nnn \l__datatool_action_return_prop
              { #1 }
              {
                \__datatool_datum:nnnn
                  { #2 } { #2 } { } { \c_datatool_integer_int }
              }
           }
         }
      }
   }
   {
     \prop_put:Nnn \l__datatool_action_return_prop
      { #1 } { #2 }
   }
}
\cs_generate_variant:Nn \__datatool_put_return_action_int:nn
 { nV, nv, nx }
```

Secondary return result is a decimal.

```
\cs_new:Nn \__datatool_put_return_action_decimal:nn
{
  \tl_set:Nn \l__datatool_action_secondary_tl { #2 }
  \__datatool_rm_weird_datum:N
    \l__datatool_action_secondary_tl
  \exp_args:NV \tl_if_head_eq_meaning:nNTF
    \l__datatool_action_secondary_tl
    \__datatool_datum:nnnn
  {
    \tl_set:Nx \l__datatool_action_secondary_tl
     { \DTLdatumvalue { \l__datatool_action_secondary_tl } }
  }
  \bool_if:NTF \l__datatool_action_datum_bool
   {
```

```
\int_compare:nNnT
  { \l__datatool_action_datum_round_int } > { -1 }
{
  \dtlround
    \l__datatool_action_secondary_tl
    \l__datatool_action_secondary_tl
   { \int_use:N \l__datatool_action_datum_round_int }
}
\bool_if:NT \l__datatool_action_datum_locale_decimal_bool
 {
   \exp_args:NV \DTLdecimaltolocale
    \l__datatool_action_secondary_tl
    \l__datatool_action_secondary_tl
 }
\tl_if_empty:NF \l__datatool_action_datum_currency_tl
 {
   \tl_if_eq:NnTF
    \l__datatool_action_datum_currency_tl
    { \l__datatool_item_currency_tl }
   {
     \tl_if_empty:NF \l__datatool_item_currency_tl
      {
       \tl_set:Nx \l__datatool_action_secondary_tl
        {
          \exp_not:N \DTLfmtcurrency
          { \exp_not:V \l__datatool_item_currency_tl }
          { \exp_not:V \l__datatool_action_secondary_tl }
        }
      }
   }
   {
     \tl_if_eq:NnTF
      \l__datatool_action_datum_currency_tl
      { \@dtl@currency }
     {
       \tl_set:Nx \l__datatool_action_secondary_tl
        {
          \exp_not:N \DTLfmtcurrency
          { \exp_not:V \@dtl@currency }
          { \exp_not:V \l__datatool_action_secondary_tl }
        }
     }
     {
       \tl_set:Nx \l__datatool_action_secondary_tl
        {
          \exp_not:N \DTLfmtcurrency
          { \exp_not:V \l__datatool_action_datum_currency_tl }
          { \exp_not:V \l__datatool_action_secondary_tl }
        }
     }
```

```
        }
      }
      \prop_put:Nnx \l__datatool_action_return_prop
       { #1 }
       {
         \exp_not:N \__datatool_datum:nnnn
           { \exp_not:V \l__datatool_action_secondary_tl }
           { #2 } { }
           { \exp_not:N \c_datatool_decimal_int }
       }
    }
    {
      \prop_put:NnV \l__datatool_action_return_prop
       { #1 } \l__datatool_action_secondary_tl
    }
}
\cs_generate_variant:Nn \__datatool_put_return_action_decimal:nn
 { nV, nv, nx }
```

Secondary result should be parsed if applicable.

```
\cs_new:Nn \__datatool_put_return_action_parse:nn
{
  \bool_if:NTF \l__datatool_action_datum_bool
   {
     \__datatool_parse:Nn
       \l__datatool_action_secondary_tl { #2 }
   }
   {
     \tl_set:Nn \l__datatool_action_secondary_tl { #2 }
     \__datatool_rm_weird_datum:N
       \l__datatool_action_secondary_tl
   }
  \prop_put:NnV \l__datatool_action_return_prop
    { #1 }  \l__datatool_action_secondary_tl
}
\cs_generate_variant:Nn \__datatool_put_return_action_parse:nn
 { VV, xV }
```

Assign secondary return properties using column keys as name and value obtained from corresponding element in the given row specs.

```
\cs_new:Nn \__datatool_set_return_from_row:nn
{
  \int_step_inline:nn
     { \DTLcolumncount { #2 } }
   {
     \__datatool_get_entry_from_row:Nnn \l__datatool_item_value_tl
      { ##1 } { #1 }
     \datatool_if_null:NF \l__datatool_item_value_tl
      {
        \@dtl@getkeyforcolumn
          \l__datatool_item_key_tl
```

```
          { #2 }
          { ##1 }
        \__datatool_put_return_action_parse:VV
          \l__datatool_item_key_tl
          \l__datatool_item_value_tl
      }
   }
}
\cs_generate_variant:Nn \__datatool_set_return_from_row:nn
 { Vn }
```

Add to options sequence unless already present:

```
\cs_new:Nn \__datatool_add_action_option:n
{
  \seq_if_in:NnF
    \l__datatool_action_tmp_options_seq { #1 }
   {
     \seq_put_right:Nn
       \l__datatool_action_tmp_options_seq { #1 }
   }
}
```

Tests if the given option has been added:

```
\cs_new:Nn \__datatool_if_action_option:nTF
{
  \seq_if_in:NnTF
   \l__datatool_action_tmp_options_seq { #1 }
   { #2 } { #3 }
}
\cs_new:Nn \__datatool_if_action_option:nT
{
  \seq_if_in:NnT
   \l__datatool_action_tmp_options_seq { #1 }
   { #2 }
}
```

\DTLifaction

```
\DTLifaction{⟨prop-key⟩}{⟨true⟩}{⟨false⟩}
```

```
\newcommand{\DTLifaction}[3]{
 \tl_if_empty:nTF { #1 }
  {
    \tl_if_eq:NNTF \l__datatool_action_return_tl \dtlnovalue
     { #3 } { #2 }
  }
  {
    \prop_if_in:NnTF \l__datatool_action_return_prop { #1 }
     { #2 } { #3 }
  }
```

```
}
```

`\DTLget` Get the return value. The first argument should either be empty or a property key. The second argument should be a token list in which to store the return value.

```
\NewDocumentCommand \DTLget { O{} m }
{
  \tl_if_blank:nTF { #1 }
  {
    \tl_set_eq:NN #2 \l__datatool_action_return_tl
  }
  {
    \__datatool_action_get:nn { #2 } { #1 }
  }
}

\cs_new:Nn \__datatool_action_get:nn
{
  \tl_if_single:nTF { #1 }
  {
    \prop_get:NnN \l__datatool_action_return_prop { #2 }
      #1
    \quark_if_no_value:NT #1
     {
       \tl_set_eq:NN #1 \dtlnovalue
     }
  }
  {
    \PackageError { datatool }
    {
      Control ~ sequence ~ expected ~ (to ~ store ~ return ~
      value): ~ found ~ \tl_to_str:n { #1 }
    }
    {
      To ~ fetch ~ an ~ action ~ return ~ value ~ you ~ need ~
      to ~ provide ~ a ~ command ~ to ~ be ~ defined ~ to ~
      the ~ returned ~ value
    }
  }
}

\cs_new:Nn \__datatool_cskey_missing_val:n
{
  \PackageError { datatool}
  {
     Invalid ~ cs=key ~ assignment ~ syntax ~ in ~
     `\tl_to_str:n { #1 }' : ~
     missing ~ key ~ or ~ property ~ name
  }
  {
    Column ~ or ~ property ~ assignment ~ lists ~ need ~ to ~
```

```
      have ~ each ~ element ~ in ~ the ~ list ~ in ~ the ~
      form ~ <cs>=<label> ~ where ~ <cs> ~ is ~ a ~ control ~
      sequence ~ (placeholder ~ command) ~ and ~ <label> ~
      is ~ the ~ label ~ identifying ~ the ~ required ~ column ~ or ~ property
    }
  }
```

`\DTLuse` Use the return value. As `\DTLget` but typesets the result. Partially expandable.

```
\newcommand* \DTLuse [ 1 ]
{
  \tl_if_empty:nTF { #1 }
  { \l__datatool_action_return_tl }
  { \@dtl@useresult { #1 } }
}
```

`\@dtl@useresult` Robust.

```
\newrobustcmd* \@dtl@useresult [ 1 ]
{
  \tl_if_blank:nTF { #1 }
  {
    \l__datatool_action_return_tl
  }
  {
    \prop_get:NnN \l__datatool_action_return_prop { #1 }
      \l__datatool_tmpb_tl
    \quark_if_no_value:NTF \l__datatool_tmpb_tl
     {
       \dtlnovalue
     }
     {
       \l__datatool_tmpb_tl
     }
  }
}

\cs_new:Nn \__datatool_action_noop:Nn
{
  \PackageError {datatool}
  {
    Action ~ `#2' ~ may ~ only ~ be ~ used ~ within ~ \token_to_str:N #1
  }
  {}
}

\cs_new:Nn \__datatool_action_error:nnn
{
  \PackageError { #1 }
   {
     Action ~ ` \l__datatool_action_tl ' : ~ #2
   }
```

```
   { #3 }
}
\cs_new:Nn \__datatool_action_error:nn
{
  \__datatool_action_error:nnn { datatool } { #1 } { #2 }
}
\cs_new:Nn \__datatool_action_error:n
{
  \__datatool_action_error:nn { #1 } { }
}
```

Commands provided for checking common syntax problems for actions. The true argument is done if successful. The false argument is done if an error occurs.

```
\cs_new:Nn \__datatool_require_database:TF
{
  \DTLifdbexists { \l__datatool_action_name_tl }
   {
     #1
   }
   {
     \__datatool_action_error:nn
      {
        Database ~ `\l__datatool_action_name_tl' ~ doesn't ~ exist
      }
      {
        Check ~ you ~ have ~ spelt ~ the ~ database ~ name ~
        correctly ~ in ~
        \token_to_str:N \DTLaction [name={ \l__datatool_action_name_tl }]
        { \l__datatool_action_tl } ~ or ~ check ~ the ~ default-
name ~
        option ~ in ~ \token_to_str:N \DTLsetup
      }
      #2
   }
}
\cs_new:Nn \__datatool_require_database:T
{
  \__datatool_require_database:TF { #1 } { }
}
```

For actions where the database name should be in `\dtldbname` not specified by the name key.

```
\cs_new:Nn \__datatool_requires_dbname:T
{
  \tl_if_empty:NTF \dtldbname
  {
    \__datatool_action_error:nn
     {
       no ~ current ~ database ~ selected ~
       or ~ \token_to_str:N \dtldbname \c_space_tl ~
```

```
      has ~ unexpectedly ~ been ~ cleared
    }
    {
      The ~ action ~ `\l__datatool_action_tl' ~
      is ~ for ~ use ~ within ~ certain ~ types ~
      of ~ loops ~ or ~ after ~ a ~ row ~ has ~
      been ~ identified ~ as ~ the ~ current ~ row ~
      or ~ other ~ type ~ of ~ command ~ that ~ sets ~
      the ~ placeholder ~ \token_to_str:N \dtldbname
    }
  }
  {
    \tl_if_eq:NNF
      \l__datatool_action_name_tl
      \l__datatool_default_dbname_tl
    {
      \tl_if_eq:NNF
        \l__datatool_action_name_tl
        \dtldbname
      {
        \PackageWarning { datatool }
        {
          Ignoring ~ name = { \l__datatool_action_name_tl } ~
          for ~ action ~ ` \l__datatool_action_tl '
        }
      }
    }
    \tl_set_eq:NN
     \l__datatool_action_name_tl
     \dtldbname
    \DTLifdbexists { \dtldbname }
    { #1 }
    {
      \__datatool_action_error:nn
       {
         No ~ current ~ row ~ selected ~ or ~
         \token_to_str:N \dtldbname \c_space_tl ~
         has ~ been ~ changed
       }
       { }
    }
  }
}
```

Gathers placeholder=colkey assignment and sets temporary sequence with each element in the form {⟨*cs*⟩}{⟨*col-key*⟩}{⟨*col-idx*⟩}{⟨*col-type*⟩}

```
\cs_new:Nn \__datatool_get_placeholder_assign:
{
  \seq_clear:N \l__datatool_action_tmp_data_seq
  \keyval_parse:nnV
```

```
    { \__datatool_cskey_missing_val:n }
    { \__datatool_action_placeholder_cskey:nn }
    \l__datatool_action_assign_clist
  }
  \cs_new:Nn \__datatool_action_placeholder_cskey:nn
  {
    \tl_if_single_token:nTF { #1 }
     {
       \tl_set:Nn #1 { \c_datatool_nullvalue_tl }
```

Get the column index:

```
       \tl_if_exist:cTF { dtl@ci@ \l__datatool_action_name_tl @ #2 }
        {
          \__datatool_get_col_type:vv
            { dtlkeys@  \l__datatool_action_name_tl }
            { dtl@ci@ \l__datatool_action_name_tl @ #2 }
          \seq_put_right:Ne \l__datatool_action_tmp_data_seq
           {
             { \exp_not:N #1 } { #2 }
             { \tl_use:c { dtl@ci@ \l__datatool_action_name_tl @ #2 } }
             { \int_use:N \l__datatool_item_type_int }
           }
        }
        {
          \__datatool_action_error:nn
           {
             Unknown ~ column ~ ` #2 '
           }
           {
             Database ~ ` \l__datatool_action_name_tl ' ~ does ~ not ~
             have ~ a ~ column ~ labelled ~ ` #2 '
           }
        }
     }
     {
       \__datatool_action_error:nn
        {
          Single ~ placeholder ~ command ~ required ~ in ~ assignment, ~ found ~
          ` \tl_to_str:n { #1 } '
        }
        {
          The ~ `assign' ~ option ~ requires ~ a ~ list ~ with ~
          elements ~ in ~ the ~ form ~ <tl-var>=<column-key>
        }
     }
  }
```

Perform assignments on given row specs

```
  \cs_new:Nn \__datatool_do_action_assignments:n
  {
```

```
  \seq_map_inline:Nn \l__datatool_action_tmp_data_seq
   {
```
Each element in the form: `{`⟨*cs*⟩`}{`⟨*col-key*⟩`}{`⟨*col-idx*⟩`}{`⟨*col-type*⟩`}`
```
      \__datatool_do_action_row_assign:nNnnn { #1 } ##1
   }
}
```
Perform assignment.
```
\cs_new:Nn \__datatool_do_action_row_assign:nNnnn
{
  \__datatool_get_entry_from_row:Nnn #2 { #4 } { #1 }
  \tl_if_eq:NnT #2 { \c_datatool_nullvalue_tl }
   {
     \datatool_if_numeric_datum_type:nTF { #5 }
      {
        \tl_set_eq:NN #2 \DTLnumbernull
      }
      {
        \tl_set_eq:NN #2 \DTLstringnull
      }
   }
}
```
Generate error if both key and column found with given help message on error.
```
\cs_new:Nn \__datatool_forbid_key_and_column:n
{
 \int_if_zero:nF { \l__datatool_action_column_int }
  {
    \__datatool_action_error:nn
     {
       `column' ~ not ~ permitted
     }
     { #1 }
  }
 \tl_if_empty:NF \l__datatool_action_key_tl
  {
    \__datatool_action_error:nn
     {
       `key' ~ not ~ permitted
     }
     { #1 }
  }
}
```
Require ‘value’.
```
\cs_new:Nn \__datatool_require_value:TF
{
  \quark_if_no_value:NTF \l__datatool_action_value_tl
   { #2 }
   { #1 }
```

```
}
\cs_new:Nn \__datatool_require_value:T
{
  \__datatool_require_value:TF
   { #1 }
   {
     \__datatool_action_error:nn
      { missing ~ value }
      {
        You ~ need ~ to ~ set ~ `value' ~ in ~ the ~ optional ~ argument ~
        of ~ \token_to_str:N \DTLaction[...]
        { \l__datatool_action_tl }. ~
        If ~ an ~ empty ~ value ~ is ~ required, ~ do ~
        value={}
      }
   }
}
```

Arguments: ⟨*row int var*⟩`{`⟨*not set*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

```
\cs_new:Nn \__datatool_optional_row:NnTF
{
  \int_if_zero:nTF { #1 }
   { #2 }
   {
    \bool_lazy_or:nnTF
       {
        \int_compare_p:nNn { #1 } < { \c_zero_int }
       }
       {
        \int_compare_p:nNn
         { #1 }
          >
         { \DTLrowcount { \l__datatool_action_name_tl } }
       }
     {
      \__datatool_action_error:nn
       {
         Invalid ~ row ~ index ~ \int_use:N #1
       }
       {
         Database ~ `\l__datatool_action_name_tl' ~ row ~ index ~
         must ~ be ~ in ~ the ~ range ~
         [ 1 , ~  \DTLrowcount { \l__datatool_action_name_tl } ]
       }
       #4
     }
     { #3 }
   }
}
\cs_new:Nn \__datatool_optional_row:NTF
```

```
{
  \__datatool_optional_row:NnTF #1 { #3 } { #2 } { #3 }
}
\cs_new:Nn \__datatool_optional_row:NF
{
  \__datatool_optional_row:NTF #1 { } { #2 }
}
```

Arguments: {⟨*key setting name*⟩}⟨*key str var*⟩{⟨*column setting name*⟩}⟨*col int var*⟩{⟨*not set*⟩}{⟨*true*⟩}{⟨*false*⟩}

One column may be identified: 'key' or 'column' allowed but both not permitted. A negative column index is an error and will reset the index to zero. The ⟨*no set*⟩ argument is done if neither set.

```
\cs_new:Nn \__datatool_optional_key_xor_column:nNnNnTF
{
```

Check the column index hasn't been set to a negative number.

```
  \int_compare:nNnTF
    { #4 }
     <
    { \c_zero_int }
  {
   \__datatool_action_error:nn
    {
      Negative ~ `#3' ~ value ~
      \int_use:N #4 \c_space_tl
      not ~ permitted
    }
    {
      You ~ need ~ to ~ correct ~ the ~ value ~ in ~ the ~
      `#3' ~ setting ~ in ~
      \token_to_str:N \DTLaction
      [#3=\int_use:N #4 \c_space_tl,...]
      { \l__datatool_action_tl } ~
      (or ~ use ~ `#1' ~ instead)
    }
   \int_zero:N #4
```

Negative column index is a failure (do false).

```
    #7
  }
  {
    \tl_if_empty:NTF #2
    {
```

No key has been supplied. Has a column index been provided?

```
       \int_if_zero:nTF #4
        {
```

Neither provided. Do ⟨*no set*⟩ argument.

```
          #5
```

```
        }
        {
```
A column index has been provided. Success (but no guarantee the column is defined). Do true argument.
```
          #6
        }
      }
    {
```
A key has been supplied. Has a column index been provided?
```
      \int_if_zero:nTF #4
       {
```
A key has been supplied but no column index. Success (but no guarantee the key is valid). Do true argument.
```
          #6
        }
        {
```
A column index has been provided as well. Failure. Do false argument.
```
         \__datatool_action_error:nn
         {
           can't ~ have ~ both ~ `#1' ~ and ~ `#3' ~ set
         }
         {
          either ~ have ~ `#1' ~ or `#3' ~ in ~ the ~ optional ~
          argument ~ of ~ \token_to_str:N \DTLaction [...]
          { \l__datatool_action_tl } ~ but ~ not ~ both
         }
         #7
        }
    }
  }
}
```
Arguments: {⟨*not set*⟩}{⟨*true*⟩}{⟨*false*⟩}

One column may be identified: ‘key’ or ‘column’ allowed but both not permitted. A negative column index is an error and will reset the index to zero. The first argument is done if neither set.
```
\cs_new:Nn \__datatool_optional_key_xor_column:nTF
{
  \__datatool_optional_key_xor_column:nNnNnTF
   { key } \l__datatool_action_key_tl
   { column } \l__datatool_action_column_int
   { #1 } { #2 } { #3 }
}
```
Second column may be identified: ‘key2’ or ‘column2’ allowed but both not permitted. A negative column index is an error and will reset the index to zero. The first argument is done if neither set.

```
\cs_new:Nn \__datatool_optional_key_ii_xor_column_ii:nTF
{
  \__datatool_optional_key_xor_column:nNnNnTF
   { key2 } \l__datatool_action_key_ii_tl
   { column2 } \l__datatool_action_column_ii_int
   { #1 } { #2 } { #3 }
}
```

Arguments: {⟨*key setting name*⟩}⟨*key str var*⟩{⟨*column setting name*⟩}⟨*col int var*⟩{⟨*not set*⟩}{⟨*key undef*⟩}{⟨*true*⟩}{⟨*false*⟩}

One column may be identified: 'key' or 'column' optional but both not permitted. If the key is provided, obtain the index. The ⟨*not set*⟩ argument is done if no key or column index provided. The ⟨*key undef*⟩ argument is done if no column matches the given key.

```
\cs_new:Nn \__datatool_optional_key_xor_column_get_column:nNnNnnTF
{
  \__datatool_optional_key_xor_column:nNnNnTF
   { #1 } { #2 } { #3 } { #4 }
   {
```

No key or column index provided. Do ⟨*not set*⟩ argument.

```
    #5
   }
   {
```

A column has been identified either by key or column index.

```
     \int_if_zero:nTF #4
      {
```

No column index so it must have been identified by key. Get the corresponding index.

```
        \tl_if_exist:cTF
           {
             dtl @ ci
             @ \l__datatool_action_name_tl
             @ #2
           }
         {
```

Column index found. Success. Do true argument.

```
           \exp_args:NNv \int_set:Nn
             #4
            {
              dtl @ ci
              @ \l__datatool_action_name_tl
              @ #2
            }
           #7
         }
         {
```

No match on key. Do ⟨*key undef*⟩ argument.

```
           #6
```

```
      }
    }
    {
```

Column index was provided (but no guarantee it's within range). Success. Do true argument.

```
      #7
    }
  }
  {
```

Invalid key or column setting (negative index or both set). Failure. Do false argument.

```
    #8
  }
}
```

Arguments: {⟨*not set*⟩}{⟨*key undef*⟩}{⟨*true*⟩}{⟨*false*⟩}

```
\cs_new:Nn \__datatool_optional_key_xor_column_get_column:nnTF
{
  \__datatool_optional_key_xor_column_get_column:nNnNnnTF
   { key } \l__datatool_action_key_tl
   { column } \l__datatool_action_column_int
   { #1 } { #2 } { #3 } { #4 }
}
\cs_new:Nn \__datatool_optional_key_xor_column:TF
{
  \__datatool_optional_key_xor_column:nTF { #1 } { #1 } { #2 }
}
```

Second column may be identified:

```
\cs_new:Nn \__datatool_optional_key_ii_xor_column_ii_get_column:nnTF
{
  \__datatool_optional_key_xor_column_get_column:nNnNnnTF
   { key2 } \l__datatool_action_key_ii_tl
   { column2 } \l__datatool_action_column_ii_int
   { #1 } { #2 } { #3 } { #4 }
}
\cs_new:Nn \__datatool_optional_key_ii_xor_column_ii:TF
{
  \__datatool_optional_key_ii_xor_column_ii:nTF { #1 } { #1 } { #2 }
}
```

Arguments: {⟨*key setting name*⟩}⟨*key str var*⟩{⟨*column setting name*⟩}⟨*col int var*⟩{⟨*not set*⟩}{⟨*true*⟩}{⟨*false*⟩}

'key' or 'column' optional but both not permitted. If the key is provided, obtain the index. The ⟨*not set*⟩ argument is done if no key or column index provided. Require column existence if key or column provided.

```
\cs_new:Nn \__datatool_optional_key_xor_column_get_column:nNnNnTF
{
  \__datatool_optional_key_xor_column_get_column:nNnNnnTF
```

```
{ #1 } { #2 } { #3 } { #4 }
{ #5 }
{
```

Column key not valid. Failure. Do false.

```
  \__datatool_action_error:nn
   {
     no ~ column ~ found ~ with ~ key ~
     ` #2 ' ~
     in ~ database ~ ` \l__datatool_action_name_tl '
   }
   {
     Check ~ that ~ you ~ have ~ correctly ~ spelt ~
     the ~ column ~ key  ~ in ~ the ~ optional ~
     argument ~ of ~ \token_to_str:N \DTLaction [...]
     { \l__datatool_action_tl } ~ and ~ that ~
     the ~ database ~ name ~ is ~ correct
   }
  #7
}
{
```

Positive column index provided or obtained from the key. Check that column index is less than the total number of columns for the database.

```
  \int_compare:nNnTF
   { #4 }
    >
   {
     \int_use:c
       { dtlcols@ \l__datatool_action_name_tl }
   }
  {
```

Out of range. Failure. Do false. (If the key was originally provided, then the column index must be valid unless the internals are incorrect.)

```
    \__datatool_action_error:nn
     {
       `#3' ~ value ~
       \int_use:N #4 \c_space_tl ~
       out ~ of ~ range ~ for ~ database ~
      ` \l__datatool_action_name_tl '
     }
     {
       The ~ database ~ ` \l__datatool_action_name_tl ' ~
       only ~ has ~
       \int_use:c { dtlcols@ \l__datatool_action_name_tl }
       \c_space_tl ~ column(s)
     }
    #7
  }
  {
```

Valid. Success. Do true.

```
        #6
      }
    }
    { #7 }
}
\cs_new:Nn \__datatool_optional_key_xor_column_get_column:nTF
{
  \__datatool_optional_key_xor_column_get_column:nNnNnTF
   { key } \l__datatool_action_key_tl
   { column } \l__datatool_action_column_int
   { #1 } { #2 } { #3 }
}
\cs_new:Nn \__datatool_optional_key_ii_xor_column_ii_get_column:nTF
{
  \__datatool_optional_key_xor_column_get_column:nNnNnTF
   { key2 } \l__datatool_action_key_ii_tl
   { column2 } \l__datatool_action_column_ii_int
   { #1 } { #2 } { #3 }
}
```

Arguments: {⟨*key setting name*⟩}⟨*key str var*⟩{⟨*column setting name*⟩}⟨*col int var*⟩{⟨*not set*⟩}{⟨*true*⟩}{⟨*false*⟩}

One column may be identified: 'key' or 'column' optional but both not permitted. If the column index is provided, obtain the key. The ⟨*not set*⟩ argument is done if no key or column index provided. An error will be triggered if the index exceeds the column count.

```
\cs_new:Nn
 \__datatool_optional_key_xor_column_get_key:nNnNnTF
{
  \__datatool_optional_key_xor_column:nNnNnTF
   { #1 } { #2 } { #3 } { #4 }
   {
```

No key or column index provided. Do ⟨*not set*⟩ argument.

```
    #5
   }
   {
```

A column has been identified either by key or column index.

```
     \tl_if_empty:NTF #2
      {
```

No key so column must've been identified by its index. Check if index doesn't exceed the total number of columns.

```
       \int_compare:nNnTF
        { #4 }
         >
        { \DTLcolumncount { \l__datatool_action_name_tl } }
        {
```

Index too large. Failure. Do false.

```
            \__datatool_action_error:nn
             {
               `#3' ~ index ~
               \int_use:N #4 \c_space_tl ~
               out ~ of ~ range ~ for ~ database ~
               ` \l__datatool_action_name_tl '
             }
             {
               The ~ database ~ ` \l__datatool_action_name_tl ' ~
               only ~ has ~
               \int_use:c { dtlcols@ \l__datatool_action_name_tl }
               \c_space_tl ~ column(s)
             }
             #7
          }
          {
```

Index in range. Find the key.

```
            \__datatool_get_col_key:vV
              { dtlkeys @ \l__datatool_action_name_tl }
              \l__datatool_action_column_int
            \quark_if_no_value:NTF \l__datatool_item_key_tl
            {
```

No key was found. Unlikely to happen given that the column index is in range. Has a column been removed from the column data without adjusting the other indexes?

```
              \__datatool_action_error:nn
               {
                 No ~ key ~ can ~ be ~ found ~ for ~
                 column ~ index ~
                 \int_use:N #4 \c_space_tl ~ in ~
                 ` \l__datatool_action_name_tl '
               }
               {
                 Something ~ seems ~ to ~ have ~ gone ~
                 wrong ~ with ~ the ~ column ~ metadata
               }
               #7
            }
            {
```

Key found. Success. Do true.

```
              \tl_set_eq:NN #2 \l__datatool_item_key_tl
              #6
            }
          }
        }
        {
```

Key provided. Success. Do true.

```
        #6
      }
    }
    {
```

Invalid key or column setting (negative index or both set). Failure. Do false argument.

```
      #7
    }
}
```

Arguments: {⟨*not set*⟩}{⟨*true*⟩}{⟨*false*⟩}

```
\cs_new:Nn
 \__datatool_optional_key_xor_column_get_key:nTF
{
  \__datatool_optional_key_xor_column_get_key:nNnNnTF
   { key } \l__datatool_action_key_tl
   { column } \l__datatool_action_column_int
   { #1 } { #2 } { #3 }
}
```

If second column allowed:

```
\cs_new:Nn
 \__datatool_optional_key_ii_xor_column_ii_get_key:nTF
{
  \__datatool_optional_key_xor_column_get_key:nNnNnTF
   { key2 } \l__datatool_action_key_ii_tl
   { column2 } \l__datatool_action_column_ii_int
   { #1 } { #2 } { #3 }
}
```

Arguments: {⟨*key setting name*⟩}⟨*key str var*⟩{⟨*column setting name*⟩}⟨*col int var*⟩{⟨*true*⟩}{⟨*false*⟩}

One column required: 'key' or 'column' required but both not permitted. A negative column index will reset it to zero. No check if the key is valid or if the column index is less than the total number of columns.

```
\cs_new:Nn
  \__datatool_requires_key_xor_column:nNnNTF
{
  \__datatool_optional_key_xor_column:nNnNnTF
   { #1 } #2 { #3 } #4
   {
```

No key or column provided. Failure. Do false.

```
      \__datatool_action_error:nn
       { missing ~ `#1' ~ or ~ `#3' }
       {
        You ~ need ~ to ~ set ~ `#1' ~ or ~ `#3' ~ in ~ the ~ optional ~
        argument ~ of ~ \token_to_str:N \DTLaction [...]
        { \l__datatool_action_tl } ~ to ~
        a ~ non-empty ~ value ~ (#1) ~ or ~ positive ~ integer ~ (#3) ~
        identifying ~ the ~ required ~ column
```

```
      }
      #6
    }
    {
```

Key or column provided. Success (but no guarantee that column index isn't greater than the total number of columns). Do true.

```
     #5
    }
    {
```

Invalid syntax (negative column index or key and column both provided). Failure. Do false.

```
      #6
    }
 }
```

First column required:

```
 \cs_new:Nn \__datatool_requires_key_xor_column:TF
 {
   \__datatool_requires_key_xor_column:nNnNTF
    { key } \l__datatool_action_key_tl
    { column } \l__datatool_action_column_int
    { #1 } { #2 }
 }
```

Do nothing if false (error messages will be triggered).

```
 \cs_new:Nn \__datatool_requires_key_xor_column:T
 {
   \__datatool_requires_key_xor_column:TF { #1 } { }
 }
```

Second column required:

```
 \cs_new:Nn \__datatool_requires_key_ii_xor_column_ii:TF
 {
   \__datatool_requires_key_xor_column:nNnNTF
    { key2 } \l__datatool_action_key_ii_tl
    { column2 } \l__datatool_action_column_ii_int
    { #1 } { #2 }
 }
```

Do nothing if false (error messages will be triggered).

```
 \cs_new:Nn \__datatool_requires_key_ii_xor_column_ii:T
 {
   \__datatool_requires_key_ii_xor_column_ii:TF { #1 } { }
 }
```

Arguments: {⟨*key setting name*⟩}⟨*key str var*⟩{⟨*column setting name*⟩}⟨*col int var*⟩{⟨*not set*⟩}{⟨*true*⟩}{⟨*false*⟩}

One column required: 'key' or 'column' required but both not permitted. The column index is required so if the key is provided, obtain the index. The ⟨*not set*⟩ argument is done if no column matches the given key.

```
\cs_new:Nn
   \__datatool_requires_key_xor_column_get_column:nNnNnTF
{
  \__datatool_requires_key_xor_column:TF
   {
     \int_if_zero:nTF #4
      {
```

No column index so it must have been identified by key. Get the corresponding index.

```
        \tl_if_exist:cTF
           {
             dtl @ ci
             @ \l__datatool_action_name_tl
             @ #2
           }
         {
           \exp_args:NNv \int_set:Nn
             #4
            {
              dtl @ ci
              @ \l__datatool_action_name_tl
              @ #2
            }
```

Column index found. Success. Do true argument.

```
           #6
         }
         {
```

No match on key. Do ⟨*not set*⟩ argument.

```
           #5
         }
       }
       {
```

Column index was provided (but no guarantee it's within range). Success. Do true argument.

```
        #6
       }
    }
    {
```

Invalid key or column setting (negative index or both set). Failure. Do false argument.

```
      #7
    }
}
```

First column required.

```
\cs_new:Nn
   \__datatool_requires_key_xor_column_get_column:nTF
{
  \__datatool_requires_key_xor_column_get_column:nNnNnTF
```

```
    { key } \l__datatool_action_key_tl
    { column } \l__datatool_action_column_int
    { #1 } { #2 } { #3 }
 }
```

Second column required.

```
 \cs_new:Nn
    \__datatool_requires_key_ii_xor_column_ii_get_column:nTF
 {
   \__datatool_requires_key_ii_xor_column_ii_get_column:nNnNnTF
    { key2 } \l__datatool_action_key_ii_tl
    { column2 } \l__datatool_action_column_ii_int
    { #1 } { #2 } { #3 }
 }
```

Arguments: {⟨*key setting name*⟩}⟨*key str var*⟩{⟨*column setting name*⟩}⟨*col int var*⟩{⟨*true*⟩}{⟨*false*⟩}

One column required: ‘key’ or ‘column’ required but both not permitted. The column index is required so if the key is provided, obtain the index. Require column existence if key or column provided.

```
 \cs_new:Nn
   \__datatool_requires_key_xor_column_get_column:nNnNTF
 {
   \__datatool_requires_key_xor_column_get_column:nTF
    {
```

Column key not valid. Failure.

```
      \__datatool_action_error:nn
       {
         invalid ~ `#1' ~ value. ~
         No ~ column ~ found ~ with ~ key ~ `#2' ~
         in ~ database ~ ` \l__datatool_action_name_tl '
       }
       {
         Check ~ that ~ you ~ have ~ correctly ~ spelt ~
         the ~ value ~ of ~ `#1' ~ in ~ the ~ optional ~
         argument ~ of ~ \token_to_str:N \DTLaction [...]
         { \l__datatool_action_tl } ~ and ~ that ~
         the ~ database ~ name ~ is ~ correct
       }
      #6
    }
    {
```

Positive column index provided or obtained from the key. Check that column index is less than the total number of columns for the database.

```
      \int_compare:nNnTF
       { #4 }
        >
       {
         \int_use:c
```

```
            { dtlcols@ \l__datatool_action_name_tl }
        }
       {
```

Out of range. Failure. Do false. (If the key was originally provided, then the column index must be valid unless the internals are incorrect.)

```
         \__datatool_action_error:nn
          {
            invalid ~ `#3' ~ value.
            Column ~ index ~
            \int_use:N #4 \c_space_tl ~
            out ~ of ~ range ~ for ~ database ~
           ` \l__datatool_action_name_tl '
          }
          {
            The ~ database ~ ` \l__datatool_action_name_tl ' ~
            only ~ has ~
            \int_use:c { dtlcols@ \l__datatool_action_name_tl }
            \c_space_tl ~ columns.
            Check ~ that ~ you ~ have ~ correctly ~ spelt ~
            the ~ value ~ of ~ `#3' ~ in ~ the ~ optional ~
            argument ~ of ~ \token_to_str:N \DTLaction [...]
            { \l__datatool_action_tl } ~ and ~ that ~
            the ~ database ~ name ~ is ~ correct
          }
         #6
       }
       {
```

Valid. Success. Do true.

```
         #5
       }
    }
    { #6 }
}
```

First column required.

```
\cs_new:Nn
   \__datatool_requires_key_xor_column_get_column:TF
{
   \__datatool_requires_key_xor_column_get_column:nNnNTF
   { key } \l__datatool_action_key_tl
   { column } \l__datatool_action_column_int
   { #1 } { #2 }
}
\cs_new:Nn \__datatool_requires_key_xor_column_get_column:T
{
  \__datatool_requires_key_xor_column_get_column:TF { #1 } { }
}
```

Second column required.

```
\cs_new:Nn
  \__datatool_requires_key_ii_xor_column_ii_get_column:TF
{
   \__datatool_requires_key_xor_column_get_column:nNnNTF
   { key2 } \l__datatool_action_key_ii_tl
   { column2 } \l__datatool_action_column_ii_int
   { #1 } { #2 }
}
\cs_new:Nn
  \__datatool_requires_key_ii_xor_column_ii_get_column:T
{
  \__datatool_requires_key_ii_xor_column_ii_get_column:TF
    { #1 } { }
}
```

Arguments: {⟨*col1 not set*⟩}{⟨*col2 not set*⟩}{⟨*true*⟩}{⟨*false*⟩}

Two columns may be identified: 'key' or 'column' allowed for first column but both not permitted; 'key2' or 'column2' allowed for second column but both not permitted. If an error occurs while attempting to get the first column, the check for the second will be omitted. If the first column hasn't been identified (with key or column) ⟨*col1 not set*⟩ will be done. If the second column hasn't been identified (with key2 or column2) ⟨*col2 not set*⟩ will be done.

```
\cs_new:Nn
  \__datatool_optional_ii_key_xor_column:nnTF
{
```

Get the first column:

```
  \__datatool_optional_key_xor_column_get_column:nTF
   { #1 }
   {
```

Get the second column:

```
     \__datatool_optional_key_ii_xor_column_ii_get_column:nTF
      { #2 } { #3 } { #4 }
   }
   { #4 }
}
\cs_new:Nn \__datatool_optional_ii_key_xor_column:TF
{
  \__datatool_optional_ii_key_xor_column:nnTF
  { } { } { #1 } { #2 }
}
\cs_new:Nn \__datatool_optional_ii_key_xor_column:
{
  \__datatool_optional_ii_key_xor_column:nnTF
  { } { } { } { }
}
```

Arguments: {⟨*col2 not set*⟩}{⟨*true*⟩}{⟨*false*⟩}

Similar but the first column is required and the second optional.

```
\cs_new:Nn \__datatool_requires_i_optional_ii_key_xor_column:nTF
```

```
 {
```
Get the first column:
```
   \__datatool_requires_key_xor_column_get_column:TF
    {
```
Get the second column:
```
      \__datatool_optional_key_ii_xor_column_ii_get_column:nTF
       { #1 } { #2 } { #3 }
    }
    { #3 }
 }
```
For actions that allow an arbitrary number of columns (up to the total number in the database). These may be specified by ‘keys’ and/or ‘columns’ options. The resulting list will be save in the sequence variable `\l__datatool_action_columns_seq`. The ‘columns’ list may include ranges. Any keys will need to be defined. Note that the sequence won’t be ordered (unless the user has supplied the order). If an ordered list is required, it will need to be sorted.
```
 \cs_new:Nn \__datatool_optional_columns:
 {
```
First check the ‘columns’ setting.
```
   \clist_map_inline:Nn
    \l__datatool_action_columns_clist
   {
     \__datatool_optional_columns:w ##1 - \q_nil \q_stop
   }
```
Next check the ‘keys’ setting.
```
   \clist_map_inline:Nn
    \l__datatool_action_keys_clist
   {
     \__datatool_optional_keys:w ##1 - \q_nil \q_stop
   }
```
Make sure that the user hasn’t accidentally used key.
```
   \tl_if_empty:NF \l__datatool_action_key_tl
    {
     \clist_if_empty:NTF \l__datatool_action_keys_clist
      {
        \PackageWarning { datatool }
        {
           Ignoring ~ key={\l__datatool_action_key_tl} ~
           for ~ action ~ `\l__datatool_action_tl'. ~
           Did ~ you ~ mean ~
           keys = { \l__datatool_action_key_tl } ?
        }
      }
      {
        \PackageWarning { datatool }
        {
```

```
          Ignoring ~ key={\l__datatool_action_key_tl} ~
          for ~ action ~ `\l__datatool_action_tl'. ~
          (Append ~ to ~ the ~ `keys' ~ list ~ if ~
           required.)
        }
      }
    }
```

Make sure that the user hasn't accidentally used key.

```
  \tl_if_empty:NF \l__datatool_action_key_ii_tl
   {
    \clist_if_empty:NTF \l__datatool_action_keys_clist
     {
       \PackageWarning { datatool }
       {
          Ignoring ~ key2={\l__datatool_action_key_ii_tl} ~
          for ~ action ~ `\l__datatool_action_tl'. ~
          Did ~ you ~ mean ~
          keys = { \l__datatool_action_key_ii_tl } ?
       }
     }
     {
       \PackageWarning { datatool }
       {
          Ignoring ~ key2={\l__datatool_action_key_ii_tl} ~
          for ~ action ~ `\l__datatool_action_tl'. ~
          (Append ~ to ~ the ~ `keys' ~ list ~ if ~
           required.)
       }
     }
   }
```

Make sure that the user hasn't accidentally used column.

```
  \int_if_zero:nF \l__datatool_action_column_int
   {
    \clist_if_empty:NTF \l__datatool_action_columns_clist
     {
       \PackageWarning { datatool }
       {
          Ignoring ~ column=
          { \int_use:N \l__datatool_action_column_int } ~
          for ~ action ~ `\l__datatool_action_tl'. ~
          Did ~ you ~ mean ~
          columns =
          { \int_use:N \l__datatool_action_column_int } ?
       }
     }
     {
       \PackageWarning { datatool }
       {
          Ignoring ~ column=
```

```
          { \int_use:N \l__datatool_action_column_int } ~
          for ~ action ~ `\l__datatool_action_tl'. ~
          (Append ~ to ~ the ~ `columns' ~ list ~ if ~
           required.)
        }
      }
    }
```

Make sure that the user hasn't accidentally used column2.

```
  \int_if_zero:nF \l__datatool_action_column_ii_int
   {
    \clist_if_empty:NTF \l__datatool_action_columns_clist
     {
       \PackageWarning { datatool }
       {
          Ignoring ~ column2=
          { \int_use:N \l__datatool_action_column_ii_int } ~
          for ~ action ~ `\l__datatool_action_tl'. ~
          Did ~ you ~ mean ~
          columns =
          { \int_use:N \l__datatool_action_column_ii_int } ?
       }
     }
     {
       \PackageWarning { datatool }
       {
          Ignoring ~ column2=
          { \int_use:N \l__datatool_action_column_ii_int } ~
          for ~ action ~ `\l__datatool_action_tl'. ~
          (Append ~ to ~ the ~ `columns' ~ list ~ if ~
           required.)
       }
     }
   }
}
```

Parse a column index range.

```
\cs_new:Npn
  \__datatool_optional_columns:w #1 - #2 \q_stop
{
  \quark_if_nil:nTF { #2 }
   {
     \__datatool_if_column_index_valid:nT { #1 }
      {
        \__datatool_optional_columns_add:n { #1 }
      }
   }
   {
     \__datatool_optional_columns_add_range:w #1 - #2 \q_stop
   }
}
```

```
\cs_new:Npn
  \__datatool_optional_columns_add_range:w
   #1 - #2 - \q_nil \q_stop
{
  \tl_if_empty:nTF { #1 }
  {
```

Start at 1.

```
    \tl_if_empty:nTF { #2 }
    {
```

End at the total number of columns.

```
      \__datatool_optional_columns_add_range:nn
       { 1 }
       { \DTLcolumncount { \l__datatool_action_name_tl } }
    }
    {
      \__datatool_if_column_index_valid:nT { #2 }
       { #2 }
       {
        \__datatool_optional_columns_add_range:nn
         { 1 } { #2 }
       }
    }
  }
  {
    \tl_if_empty:nTF { #2 }
    {
```

End at the total number of columns.

```
      \__datatool_if_column_index_valid:nT { #1 }
       {
        \__datatool_optional_columns_add_range:nn
         { #1 }
         { \DTLcolumncount { \l__datatool_action_name_tl } }
       }
    }
    {
      \__datatool_if_column_index_valid:nT { #1 }
       {
         \__datatool_if_column_index_valid:nT { #2 }
          {
            \int_compare:nNnTF
             { #2 } > { #1 }
             {
               \__datatool_action_error:nn
                {
                  Invalid ~ column ~ range ~ `#1-#2'. ~
                  Start ~ of ~ range ~ can't ~ be ~
                  greater ~ than ~ end ~ of ~ range ~
                }
```

```
                {
                  Check ~ the ~ syntax ~ of ~ the ~ `columns' ~
                  setting ~ in ~ the ~ optional ~
                  argument ~ of ~ \token_to_str:N \DTLaction [...]
                  { \l__datatool_action_tl }
                }
              }
              {
               \__datatool_optional_columns_add_range:nn
                { #1 } { #2 }
              }
          }
        }
    }
  }
}
```

Add a column range.

```
\cs_new:Nn
  \__datatool_optional_columns_add_range:nn
{
  \int_step_inline:nnn
   { #1 } { #2 }
   {
     \__datatool_optional_columns_add:n { ##1 }
   }
}
\cs_generate_variant:Nn
  \__datatool_optional_columns_add_range:nn
  { vn, nv, vv }
```

Parse a column index range.

```
\cs_new:Nn \__datatool_if_column_index_valid:nT
{
 \int_compare:nNnTF
  { #1 } < { \c_one_int }
 {
   \__datatool_action_error:nn
    {
      Invalid ~ column ~ index ~ `#1'
    }
    {
      Check ~ the ~ syntax ~ of ~ the ~ `columns' ~
      setting ~ in ~ the ~ optional ~
      argument ~ of ~ \token_to_str:N \DTLaction [...]
      { \l__datatool_action_tl }
    }
 }
 {
   \int_compare:nNnTF
     { #1 }
```

```
      >
     { \DTLcolumncount { \l__datatool_action_name_tl } }
    {
      \__datatool_action_error:nn
       {
        Invalid ~ column ~ index ~ `#1' ~
        (exceeds ~ column ~ count ~ of ~ database ~
        ` \l__datatool_action_name_tl ')
       }
     {
       Check ~ the ~ syntax ~ of ~ the ~ `columns' ~
       setting ~ in ~ the ~ optional ~
       argument ~ of ~ \token_to_str:N \DTLaction [...]
       { \l__datatool_action_tl } ~ and ~ that ~
       the ~ database ~ name ~ is ~ correct
     }
    }
    { #2 }
  }
}
```

Add a column index to the sequence if not already there.

```
\cs_new:Nn \__datatool_optional_columns_add:n
 {
   \seq_if_in:NnF
    \l__datatool_action_columns_seq { #1 }
   {
     \seq_put_right:Nn
      \l__datatool_action_columns_seq { #1 }
   }
 }
\cs_generate_variant:Nn
  \__datatool_optional_columns_add:n { x , v }
```

Parse a column key range.

```
\cs_new:Npn
  \__datatool_optional_keys:w #1 - #2 \q_stop
{
  \quark_if_nil:nTF { #2 }
   {
     \__datatool_if_column_key_valid:nT { #1 }
      {
        \__datatool_optional_columns_add:v
         {
           dtl@ci
           @ \l__datatool_action_name_tl
           @ #1
         }
      }
   }
   {
```

```
      \__datatool_optional_keys_add_range:w #1 - #2 \q_stop
    }
}
\cs_new:Npn
  \__datatool_optional_keys_add_range:w
   #1 - #2 - \q_nil \q_stop
{
  \tl_if_empty:nTF { #1 }
  {
```

Start at 1.

```
    \tl_if_empty:nTF { #2 }
    {
```

End at the total number of columns.

```
      \__datatool_optional_columns_add_range:nn
       { 1 }
       { \DTLcolumncount { \l__datatool_action_name_tl } }
    }
    {
      \__datatool_if_column_key_valid:nT { #2 }
       {
        \__datatool_optional_columns_add_range:nv
         { 1 }
         {
           dtl@ci
           @ \l__datatool_action_name_tl
           @ #2
         }
       }
    }
  }
  {
    \tl_if_empty:nTF { #2 }
    {
```

End at the total number of columns.

```
      \__datatool_if_column_key_valid:nT { #1 }
       {
        \__datatool_optional_columns_add_range:vn
         {
           dtl@ci
           @ \l__datatool_action_name_tl
           @ #1
         }
         { \DTLcolumncount { \l__datatool_action_name_tl } }
       }
    }
    {
      \__datatool_if_column_key_valid:nT { #1 }
       {
```

```
\__datatool_if_column_key_valid:nT { #2 }
 {
```

Unlike the column index ranges, with a key range, if the start key has a smaller index than the end key, switch them round.

```
    \int_compare:nNnTF
       {
         \tl_use:c
          {
            dtl@ci
            @ \l__datatool_action_name_tl
            @ #2
          }
       }
         >
       {
         \tl_use:c
          {
            dtl@ci
            @ \l__datatool_action_name_tl
            @ #1
          }
       }
      {
       \__datatool_optional_columns_add_range:vv
         {
           dtl@ci
           @ \l__datatool_action_name_tl
           @ #1
         }
         {
           dtl@ci
           @ \l__datatool_action_name_tl
           @ #2
         }
      }
      {
       \__datatool_optional_columns_add_range:vv
         {
           dtl@ci
           @ \l__datatool_action_name_tl
           @ #2
         }
         {
           dtl@ci
           @ \l__datatool_action_name_tl
           @ #1
         }
      }
  }
```

```
      }
    }
  }
}
```

Check the column key is valid.

```
\cs_new:Nn \__datatool_if_column_key_valid:nT
{
  \tl_if_exist:cTF
   {
     dtl@ci
     @ \l__datatool_action_name_tl
     @ #1
   }
   { #2 }
 {
   \__datatool_action_error:nn
    {
      Unknown ~ column ~ key ~ `#1' ~
      in ~ database ~ `\l__datatool_action_name_tl'
    }
    {
      Check ~ the ~ `keys' ~
      setting ~ in ~ the ~ optional ~
      argument ~ of ~ \token_to_str:N \DTLaction [...]
      { \l__datatool_action_tl }
    }
 }
}
```

Define actions. New database (action=new):

```
\cs_new:cn { __datatool_action_new: }
{
  \DTLifdbexists
    { \l__datatool_action_name_tl }
  {
    \__datatool_action_error:nn
     {
       Database ~ `\l__datatool_action_name_tl' ~
       already ~ exists
     }
     {
       Did ~ you ~ forget ~ to ~ specify ~ the ~
       database ~ name ~ in ~ the ~ options ~ for ~
       \token_to_str:N \DTLaction [ ... ]
       { \l__datatool_action_tl } ~
       or ~ change ~ the ~ default ~ name ~ in ~
       \token_to_str:N \DTLsetup?
     }
  }
  {
```

```
    \__datatool_new_db:n { \l__datatool_action_name_tl }
```
Set the return value:
```
    \tl_set_eq:NN \l__datatool_action_return_tl \l__datatool_action_name_tl
```
Duplicate this as a secondary return value.
```
    \__datatool_put_return_action_string:nV
      { name } \l__datatool_action_name_tl
  }
}
```
Delete database (action=delete):
```
\cs_new:cn { __datatool_action_delete: }
{
  \__datatool_require_database:T
  {
```
Set the column and row return values first.
```
    \int_if_exist:cT { dtlrows@ \l__datatool_action_name_tl }
    {
      \__datatool_put_return_action_int:nv
       { rows }
       { dtlrows@ \l__datatool_action_name_tl }
    }
    \int_if_exist:cT { dtlcols@ \l__datatool_action_name_tl }
    {
      \__datatool_put_return_action_int:nv
       { columns }
       { dtlcols@ \l__datatool_action_name_tl }
    }
```
Delete the database:
```
    \DTLdeletedb { \l__datatool_action_name_tl }
```
Set the primary return value:
```
    \tl_set_eq:NN
      \l__datatool_action_return_tl
      \l__datatool_action_name_tl
    \__datatool_put_return_action_string:nV
     { name } \l__datatool_action_name_tl
  }
```
For consistency with the other actions, also set the name property.
```
}
```
Clear database (action=clear):
```
\cs_new:cn { __datatool_action_clear: }
{
  \__datatool_require_database:T
  {
```
Set the column and row return values first.
```
    \int_if_exist:cT { dtlrows@ \l__datatool_action_name_tl }
```

```
    {
      \__datatool_put_return_action_int:nv
       { rows }
       { dtlrows@ \l__datatool_action_name_tl }
    }
    \int_if_exist:cT { dtlcols@ \l__datatool_action_name_tl }
    {
      \__datatool_put_return_action_int:nv
       { columns }
       { dtlcols@ \l__datatool_action_name_tl }
    }
```

Clear the database:

```
    \DTLcleardb { \l__datatool_action_name_tl }
```

Set the primary return value:

```
    \tl_set_eq:NN
      \l__datatool_action_return_tl
      \l__datatool_action_name_tl
  }
```

For consistency with the other actions, also set the name property.

```
  \__datatool_put_return_action_string:nV
   { name } \l__datatool_action_name_tl
}
```

New row (action=new row):

```
\cs_new:cn { __datatool_action_new ~ row: }
{
 \__datatool_require_database:T
  {
    \tl_clear:N \l__datatool_action_value_tl
```

Check for column and key in case user has confused this action with ‘new entry’.

```
    \__datatool_forbid_key_and_column:n
     {
       The ~ action ~ ` \l__datatool_action_tl ' ~ should ~ have ~
       column-key = value ~ comma ~ separated ~ list ~ in ~ the ~
      `assign' ~ setting ~ to ~ add ~ entries ~ to ~ the ~
       new ~ row
     }
```

Check for ‘options’ in case user has confused it for ‘assign’.

```
    \clist_if_empty:NF  \l__datatool_action_options_clist
    {
      \PackageWarning { datatool }
       {
         `options' ~ setting ~ ignored ~ in ~ action ~
          ` \l__datatool_action_tl ' ~
        (did ~ you ~ mean ~ `assign'?)
       }
    }
```

```
    \keyval_parse:NNV
      \__datatool_new_entry:n
      \__datatool_new_entry:nn
      \l__datatool_action_assign_clist
    \bool_if:NTF \l__datatool_db_global_bool
     {
```

Increment row count.

```
      \int_gincr:c { dtlrows@ \l__datatool_action_name_tl }
     }
     {
```

Increment row count.

```
      \int_incr:c { dtlrows@ \l__datatool_action_name_tl }
     }
```

Append the new row to the database

```
    \__datatool_dtldb_put_right:Vx
      \l__datatool_action_name_tl
     {
       \__datatool_row_markup:vV
         { dtlrows@ \l__datatool_action_name_tl }
         \l__datatool_action_value_tl
     }
```

The primary return value is the row count:

```
    \tl_set:Nv \l__datatool_action_return_tl
      { dtlrows@ \l__datatool_action_name_tl }
  }
```

Set the return value:

```
  \__datatool_put_return_action_string:nV
   { name } \l__datatool_action_name_tl
  \__datatool_put_return_action_int:nx
   { row }
   { \DTLrowcount { \l__datatool_action_name_tl } }
}
```

`\__datatool_new_entry:nn{`⟨*key*⟩`}{`⟨*value*⟩`}` adds value to `\l__datatool_action_value_tl` using database column markup

```
\cs_new:Nn \__datatool_new_entry:nn
{
  \tl_set:Nx \l__datatool_item_key_tl { #1 }
```

This bit is as `\@dtl@storeandupdate`. Store the value of this entry in `\l__datatool_item_value_tl` and also the older `\@dtl@toks` taking the expansion setting into account. This will also parse and set the data type in `\@dtl@datatype`.

```
  \__datatool_process_new_value:n { #2 }
```

Is there already a column with the given key?

```
  \tl_if_exist:cTF
   {
     dtl@ci@
```

```
      \l__datatool_action_name_tl
      @ \l__datatool_item_key_tl
    }
  {
```

This column already exists. Get the column index from the key.

```
    \exp_args:NNc \int_set:Nn \l__datatool_action_column_int
      {
         dtl@ci@
          \l__datatool_action_name_tl
        @ \l__datatool_item_key_tl
      }
```

Does the meta data need updating?

```
    \__datatool_get_col_type:vV
      { dtlkeys @ \l__datatool_action_name_tl }
      \l__datatool_action_column_int
    \int_compare:nNnT
       { \@dtl@datatype } > { \l__datatool_item_type_int }
      {
        \int_set_eq:NN
          \l__datatool_item_type_int
          \@dtl@datatype
```

Column type needs updating. Get current properties.

```
        \__datatool_get_props:NNNNNvV
          \l__datatool_item_key_tl
          \l__datatool_item_type_tl
          \l__datatool_item_head_tl
          \l__datatool_before_tl
          \l__datatool_after_tl
          { dtlkeys@ \l__datatool_action_name_tl }
          \l__datatool_action_column_int
```

Update underlying token register.

```
        \__datatool_dtlkeys_set:Vx
          \l__datatool_action_name_tl
          {
            \exp_not:V \l__datatool_before_tl
            \__datatool_column_markup:VVVV
              \l__datatool_action_column_int
              \l__datatool_item_key_tl
              \l__datatool_item_type_int
              \l__datatool_item_head_tl
            \exp_not:V \l__datatool_after_tl
          }
      }
  }
  {
```

Column doesn’t exist so create it.

```
    \int_set:Nn \l__datatool_action_column_int
```

```
      { \DTLcolumncount { \l__datatool_action_name_tl } + 1 }
```
Append to column metadata list.
```
     \__datatool_dtlkeys_put_right:Vx
       \l__datatool_action_name_tl
      {
        \__datatool_column_markup:VVVV
          \l__datatool_action_column_int
          \l__datatool_item_key_tl
          \@dtl@datatype
          \l__datatool_item_key_tl
      }
     \bool_if:NTF \l__datatool_db_global_bool
      {
```
Assign key to column index mapping.
```
        \tl_gclear_new:c
          {
           dtl@ci@
           \l__datatool_action_name_tl
           @ \l__datatool_item_key_tl
          }
        \tl_gset:cV
          {
           dtl@ci@
           \l__datatool_action_name_tl
           @ \l__datatool_item_key_tl
          }
         \l__datatool_action_column_int
```
Update column count.
```
         \int_gset_eq:cN
           { dtlcols@ \l__datatool_action_name_tl }
          \l__datatool_action_column_int
      }
      {
```
Assign key to column index mapping (local).
```
        \tl_clear_new:c
          {
           dtl@ci@
           \l__datatool_action_name_tl
           @ \l__datatool_item_key_tl
          }
        \tl_set:cV
          {
           dtl@ci@
           \l__datatool_action_name_tl
           @ \l__datatool_item_key_tl
          }
         \l__datatool_action_column_int
```

Update column count.

```
        \int_set_eq:cN
         { dtlcols@ \l__datatool_action_name_tl }
         \l__datatool_action_column_int
     }
  }
```

Append to token list variable used to store the current row.

```
  \tl_put_right:Nx \l__datatool_action_value_tl
   {
      \__datatool_row_element_markup:VV
        \l__datatool_action_column_int
        \l__datatool_item_value_tl
   }
}
```

Value not provided:

```
\cs_new:Nn \__datatool_new_entry:n
{
 \__datatool_action_error:nn
  {
    Invalid ~ option ~ setting ~ ` \tl_to_str:n { #1 } '
  }
  {
    The ~ action ~ ` \l__datatool_action_tl ' ~ should ~ have ~
    column-key = {value} ~ comma ~ separated ~ list ~ in ~ the ~
    `assign' ~ setting ~ to ~ add ~ entries ~ to ~ the ~ new ~
    row. ~ If ~ you ~ want ~ an ~
    empty ~ value ~ do ~ \token_to_str:N \DTLaction
     [ assign = { column-key = { ... }, ... } ] { \l__datatool_action_tl }
  }
}
```

New entry (action=new entry):

```
\cs_new:cn { __datatool_action_new ~ entry: }
{
 \__datatool_require_database:T
 {
   \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int
   \__datatool_require_value:T
    {
      \__datatool_requires_key_xor_column:T
      {
        \tl_if_empty:NTF \l__datatool_action_key_tl
        {
```

No key so the column index was provided instead. First check if there is a corresponding column.

```
          \int_set_eq:NN
            \dtlcolumnnum
            \l__datatool_action_column_int
```

This bit is as `\@dtl@storeandupdate`. Store the value of this entry in `\@dtl@toks` taking the expansion setting into account. This will also set the `\l__datatool_item_value_tl` and `\@dtl@datatype` variables.

```
\__datatool_process_new_value:V
  \l__datatool_action_value_tl
```

Is there a key associated with this column index? (This also gets the data type.)

```
\__datatool_get_col_type_key:vV
  { dtlkeys @ \l__datatool_action_name_tl }
  \l__datatool_action_column_int
\quark_if_no_value:NTF \l__datatool_item_key_tl
{
```

No key, so this must be a new column. Create a default key.

```
  \tl_set:Nx \l__datatool_item_key_tl
    {
      \dtldefaultkey
      \int_use:N \l__datatool_action_column_int
    }
```

Append to property list

```
  \__datatool_dtlkeys_put_right:Vx
   \l__datatool_action_name_tl
    {
      \__datatool_column_markup:VVVV
        \l__datatool_action_column_int
        \l__datatool_item_key_tl
        \@dtl@datatype
        \l__datatool_item_key_tl
    }
  \bool_if:NTF \l__datatool_db_global_bool
  {
    \tl_gclear_new:c
      {
         dtl@ci
         @ \l__datatool_action_name_tl
         @ \l__datatool_item_key_tl
      }
    \tl_gset:cV
      {
         dtl@ci
         @ \l__datatool_action_name_tl
         @ \l__datatool_item_key_tl
      }
      \l__datatool_action_column_int
```

Is the column index greater than the current size?

```
  \int_compare:nNnT
     { \l__datatool_action_column_int }
      >
     { \int_use:c { dtlcols@ \l__datatool_action_name_tl } }
```

```
      {
        \int_gset_eq:cN
          { dtlcols@ \l__datatool_action_name_tl }
          \l__datatool_action_column_int
      }
    }
    {
      \tl_clear_new:c
        {
          dtl@ci
          @ \l__datatool_action_name_tl
          @ \l__datatool_item_key_tl
        }
      \tl_set:cV
        {
          dtl@ci
          @ \l__datatool_action_name_tl
          @ \l__datatool_item_key_tl
        }
        \l__datatool_action_column_int
```

Is the column index greater than the current size?

```
      \int_compare:nNnT
         { \l__datatool_action_column_int }
          >
         {
           \int_use:c
             { dtlcols@ \l__datatool_action_name_tl }
         }
      {
        \int_set_eq:cN
          { dtlcols@ \l__datatool_action_name_tl }
          \l__datatool_action_column_int
      }
    }
  }
  {
```

This column already exists. Does the meta data need updating?

```
    \int_set_eq:NN \l__datatool_item_type_int
       \c_datatool_unknown_int
    \tl_if_empty:NTF \l__datatool_item_type_tl
    {
      \int_compare:nNnF
        { \@dtl@datatype } = { \c_datatool_unknown_int }
      {
        \int_set_eq:NN
          \l__datatool_item_type_int
          \@dtl@datatype
      }
    }
```

```
    {
      \int_compare:nNnT
       { \@dtl@datatype } > { \l__datatool_item_type_tl}
      {
        \int_set_eq:NN
          \l__datatool_item_type_int
          \@dtl@datatype
      }
    }
    \int_compare:nNnT
      { \l__datatool_item_type_int } > { \c_datatool_unknown_int }
    {
```

Column type needs updating.

```
      \__datatool_get_props:NNNNNvV
        \l__datatool_item_key_tl
        \l__datatool_item_type_tl
        \l__datatool_item_head_tl
        \l__datatool_before_tl
        \l__datatool_after_tl
        { dtlkeys@ \l__datatool_action_name_tl }
        \dtlcolumnnum
```

Update underlying token register.

```
      \__datatool_dtlkeys_set:Vx
         \l__datatool_action_name_tl
        {
          \exp_not:V \l__datatool_before_tl
          \__datatool_column_markup:VVVV
            \dtlcolumnnum
            \l__datatool_item_key_tl
            \l__datatool_item_type_int
            \l__datatool_item_head_tl
          \exp_not:V \l__datatool_after_tl
        }
    }
  }
```

Now split off the last row and insert the new entry. This expects the value to be in `\@dtl@toks`:

```
    \@s@DTLnewdbentry \l__datatool_action_name_tl
  }
  {
```

Key provided, use `\DTLnewdbentry`:

```
    \int_zero:N \dtlcolumnnum
    \exp_args:NNVV \DTLnewdbentry
        \l__datatool_action_name_tl
        \l__datatool_action_key_tl
        \l__datatool_action_value_tl
```

Get the column index for the return value.

```
          \int_set_eq:NN \l__datatool_action_column_int \dtlcolumnnum
        }
```

The primary return value is only set if successful.

```
        \tl_set:NV \l__datatool_action_return_tl
           \l__datatool_action_column_int
```

Set the return values to indicate success.

```
       \__datatool_put_return_action_int:nV
        { column } \l__datatool_action_column_int
       \__datatool_put_return_action_string:nV
        { key } \l__datatool_action_key_tl
       \__datatool_put_return_action_int:nV
        { type } \@dtl@datatype
      }
    }
```

The row property is just the row count:

```
    \__datatool_put_return_action_int:nv
     { row }
     { dtlrows@ \l__datatool_action_name_tl }
  }
```

Always set the name as a secondary return value to help debugging:

```
  \__datatool_put_return_action_string:nV
   { name } \l__datatool_action_name_tl
}
```

Get column index (action=column index):

```
\cs_new:cn { __datatool_action_column ~ index: }
{
```

Column must be identified by its label. (No point identifying it by the column index as that's what's being queried.)

```
  \tl_if_empty:NTF \l__datatool_action_key_tl
  {
```

No key has been supplied.

```
    \__datatool_action_error:nn
     { Missing ~ `key' }
     {
        \token_to_str:N \DTLaction [...]{ \l__datatool_action_tl } ~
        needs ~ the ~ `key' ~ setting ~ in ~ the ~ optional ~
        argument ~ to ~ identify ~ the ~ required ~ column
     }
  }
  {
    \__datatool_require_database:T
    {
      \exp_args:NVV \@sDTLifhaskey
        \l__datatool_action_name_tl
        \l__datatool_action_key_tl
      {
```

Primary return value is the column index:

```
          \@sdtl@getcolumnindex
            { \l__datatool_action_return_tl }
            { \l__datatool_action_name_tl }
            { \l__datatool_action_key_tl }
```

Duplicate as a secondary return value:

```
          \__datatool_put_return_action_int:nV
           { column } \l__datatool_action_return_tl
        }
        { }
      }
    }
```

Set the secondary return values:

```
    \__datatool_put_return_action_string:nV
     { name } \l__datatool_action_name_tl
    \__datatool_put_return_action_string:nV
     { key } \l__datatool_action_key_tl
}
```

Get column metadata (action=column data):

```
\cs_new:cn { __datatool_action_column ~ data: }
{
  \__datatool_require_database:T
   {
```

Initialise:

```
     \tl_set:Nn \l__datatool_item_head_tl
       { \q_no_value }
     \int_set_eq:NN
      \l__datatool_item_type_int
      \c_datatool_unknown_int
     \__datatool_requires_key_xor_column_get_column:T
      {
        \__datatool_get_col_data:vV
          { dtlkeys@ \l__datatool_action_name_tl }
          \l__datatool_action_column_int
```

Set the return values:

```
        \quark_if_no_value:NF \l__datatool_item_key_tl
         {
           \__datatool_put_return_action_string:nV
            { key } \l__datatool_item_key_tl
```

Primary return value is the key.

```
            \tl_set_eq:NN
              \l__datatool_action_return_tl
              \l__datatool_item_key_tl
```

Duplicate as secondary return value.

```
            \__datatool_put_return_action_string:nV
```

```
            { key } \l__datatool_item_key_tl
          }
```
Set the secondary return values:
```
        \quark_if_no_value:NF \l__datatool_item_head_tl
         {
           \__datatool_put_return_action_string:nV
           { header } \l__datatool_item_head_tl
         }
       \__datatool_put_return_action_int:nV
        { type } \l__datatool_item_type_int
       \int_if_zero:nF \l__datatool_action_column_int
        {
          \__datatool_put_return_action_int:nV
          { column } \l__datatool_action_column_int
        }
       }
     \__datatool_put_return_action_string:nV
      { name } \l__datatool_action_name_tl
  }
}
```
Display data (action=display):
```
\cs_new:cn { __datatool_action_display: }
{
  \__datatool_require_database:T
   {
    \group_begin:
      \clist_if_empty:NF \l__datatool_action_options_clist
      {
        \keys_set_filter:nnVN
          { datatool/display } { longtable }
          \l__datatool_action_options_clist
          \l__datatool_action_tmpb_tl
        \tl_if_empty:NF \l__datatool_action_tmpb_tl
        {
          \PackageWarning { datatool }
          {
            Ignoring ~ unsupported ~
            \token_to_str:N \DTLaction
            { \l__datatool_action_tl} ~
            option(s): ~ \exp_not:o { \l__datatool_action_tmpb_tl }
          }
        }
      }
      \tl_set_eq:NN \dtldbname \l__datatool_action_name_tl
      \__datatool_display_db:
```
Need to set return properties after the group ends:
```
      \tl_set:Nx \l__datatool_action_tmpb_tl
        {
```

```
          \exp_not:N \group_end:
          \exp_not:N \__datatool_put_return_action_int:nn
            { columns }
            { \int_use:N \dtlcolumnnum }
          \exp_not:N \__datatool_put_return_action_int:nn
            { rows }
            { \int_use:N \dtlrownum }
        }
     \l__datatool_action_tmpb_tl
    }
```

Set the secondary return values:

```
   \__datatool_put_return_action_string:nV
    { name } \l__datatool_action_name_tl
 }
```

Display data in a longtable (action=display long):

```
 \cs_new:cn { __datatool_action_display ~ long: }
 {
   \__datatool_require_database:T
    {
     \group_begin:
       \clist_if_empty:NF \l__datatool_action_options_clist
       {
         \keys_set_filter:nnVN
           { datatool/display } { tabular }
           \l__datatool_action_options_clist
           \l__datatool_action_tmpb_tl
         \tl_if_empty:NF \l__datatool_action_tmpb_tl
         {
           \PackageWarning { datatool }
           {
             Ignoring ~ unsupported ~
             \token_to_str:N \DTLaction
             { \l__datatool_action_tl} ~
             option(s): ~ \exp_not:o { \l__datatool_action_tmpb_tl }
           }
         }
       }
       \tl_set_eq:NN \dtldbname \l__datatool_action_name_tl
       \__datatool_display_long_db:
```

Need to set return properties after the group ends:

```
       \tl_set:Nx \l__datatool_action_tmpb_tl
         {
           \exp_not:N \group_end:
           \exp_not:N \__datatool_put_return_action_int:nn
             { columns }
             { \int_use:N \dtlcolumnnum }
           \exp_not:N \__datatool_put_return_action_int:nn
             { rows }
```

```
            { \int_use:N \dtlrownum }
        }
     \l__datatool_action_tmpb_tl
    }
```

Set the secondary return values:

```
   \__datatool_put_return_action_string:nV
    { name } \l__datatool_action_name_tl
 }
```

Add a new column (action=add column):

```
 \cs_new:cn { __datatool_action_add ~ column: }
 {
  \__datatool_require_database:T
   {
     \int_if_zero:nF { \l__datatool_action_column_int }
     {
      \__datatool_action_error:nn
       {
        `column' ~ setting ~ not ~ supported. ~ Ignoring
       }
       {
         You ~ need ~ to ~ use ~ `key' ~ not ~ `column' ~ to ~
         reference ~ a ~ column ~ with ~ action ~ ` \l__datatool_action_tl '
       }
     }
```

The column index will be one more than the current number of columns.

```
     \int_set:Nn \l__datatool_action_column_int
       { \DTLcolumncount \l__datatool_action_name_tl + 1 }
```

If no key has been set, assign the default.

```
     \tl_if_empty:NT \l__datatool_action_key_tl
      {
        \tl_set:Nx \l__datatool_action_key_tl
         {
           \dtldefaultkey
           \int_use:N \l__datatool_action_column_int
         }
      }
```

If no value has been set, assume the header is the same as the key.

```
     \quark_if_no_value:NTF \l__datatool_action_value_tl
      {
        \tl_set_eq:NN
          \l__datatool_action_value_tl
          \l__datatool_action_key_tl
      }
     \exp_args:NVV \@sDTLifhaskey
        \l__datatool_action_name_tl
        \l__datatool_action_key_tl
     {
```

```
      \__datatool_action_error:nn
       {
         Can't ~ add ~ new ~ column ~ ` \l__datatool_action_key_tl ' ~
         to ~ database ~ ` \l__datatool_action_name_tl ': ~
         column ~ with ~ that ~ key ~ already ~ exists
       }
       {
         Check ~ the ~ key ~ setting ~ in ~ the ~ optional ~
         argument ~ of ~ \token_to_str:N \DTLaction [...]
         { \l__datatool_action_tl }
       }
     }
     {
       \__datatool_add_column_with_header:VVVV
         \l__datatool_action_name_tl
         \l__datatool_action_key_tl
         \l__datatool_action_type_int
         \l__datatool_action_value_tl
```

The primary return value is only set if successful.

```
        \tl_set:NV \l__datatool_action_return_tl
          \l__datatool_action_column_int
```

Duplicate as a secondary return value.

```
       \__datatool_put_return_action_int:nV
        { column } \l__datatool_action_column_int
     }
```

Set the return values:

```
     \__datatool_put_return_action_string:nV
      { key } \l__datatool_action_key_tl
     \__datatool_put_return_action_int:nV
      { type } \l__datatool_action_type_int
     \quark_if_no_value:NF \l__datatool_action_value_tl
     {
       \__datatool_put_return_action_string:nV
        { header } \l__datatool_action_value_tl
     }
   }
  \__datatool_put_return_action_string:nV
    { name } \l__datatool_action_name_tl
}
```

Get column aggregates (action=aggregate):

```
\cs_new:cn { __datatool_action_aggregate: }
{
```

Check if database exists:

```
   \__datatool_require_database:T
   {
```

Initialise:

```
\fp_set_eq:NN \l__datatool_min_fp \c_inf_fp
\fp_set_eq:NN \l__datatool_min_ii_fp \c_inf_fp
\fp_set_eq:NN \l__datatool_max_fp \c_minus_inf_fp
\fp_set_eq:NN \l__datatool_max_ii_fp \c_minus_inf_fp
\fp_zero:N \l__datatool_total_fp
\fp_zero:N \l__datatool_total_ii_fp
\tl_clear:N \l__datatool_item_currency_tl
\tl_clear:N \l__datatool_item_currency_ii_tl
```

List of aggregate types required:

```
\seq_clear:N \l__datatool_tmp_seq
\clist_if_empty:NF \l__datatool_action_options_clist
{
  \clist_map_inline:Nn
   \l__datatool_action_options_clist
  {
    \clist_if_in:nnTF
      { sum , mean , variance, sd , min , max } { ##1 }
     {
       \seq_put_right:Nn \l__datatool_tmp_seq { ##1 }
     }
     {
       \PackageWarning { datatool }
        {
          Unknown ~ aggregate ~ function ~ `##1' ~ in ~
          \token_to_str:N \DTLaction
           [ options={\l__datatool_action_options_clist}, ... ]
           {` \l__datatool_action_tl ' }
        }
     }
  }
}
```

Add dependent types if omitted:

```
\seq_if_in:NnT
  \l__datatool_tmp_seq { sd }
 {
   \seq_if_in:NnF
     \l__datatool_tmp_seq { variance }
    {
      \seq_put_right:Nn \l__datatool_tmp_seq { variance }
    }
 }
\seq_if_in:NnT
  \l__datatool_tmp_seq { variance }
 {
   \seq_if_in:NnF
     \l__datatool_tmp_seq { mean }
    {
      \seq_put_right:Nn \l__datatool_tmp_seq { mean }
    }
```

```
 }
\seq_if_in:NnT
  \l__datatool_tmp_seq { mean }
 {
   \seq_if_in:NnF
     \l__datatool_tmp_seq { sum }
    {
      \seq_put_right:Nn \l__datatool_tmp_seq { sum }
    }
 }
```

List of numeric values:

```
\seq_clear:N \l__datatool_tmpa_seq
\seq_clear:N \l__datatool_tmpb_seq
```

Check the key and column settings, which will set the index for column 1 and column 2:

```
\__datatool_optional_ii_key_xor_column:
```

Iterate and compute required aggregates:

```
\DTLmapdata [ name = { \l__datatool_action_name_tl }, read-
only ]
{
  \exp_args:NNVV
     \dtl@getentryfromrow
     \l__datatool_item_value_tl
     \l__datatool_action_column_int
     \l__datatool_map_data_row_tl
  \datatool_if_null:NF \l__datatool_item_value_tl
   {
     \__datatool_parse:N \l__datatool_item_value_tl
```

Check that the value is numerical.

```
      \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
      {
        \int_compare:nNnT
          { \@dtl@datatype } = { \c_datatool_currency_int }
        {
          \tl_if_empty:NT \l__datatool_item_currency_tl
           {
             \tl_set_eq:NN
               \l__datatool_item_currency_tl
               \l__datatool_datum_currency_tl
           }
        }
        \fp_set:Nn \l__datatool_tmpa_fp
         { \DTLdatumvalue { \l__datatool_item_value_tl } }
        \seq_put_right:Nx \l__datatool_tmpa_seq
         { \fp_to_decimal:N \l__datatool_tmpa_fp }
        \seq_if_in:NnT \l__datatool_tmp_seq { sum }
        {
          \fp_add:Nn \l__datatool_total_fp
```

```
            \l__datatool_tmpa_fp
          }
          \seq_if_in:NnT \l__datatool_tmp_seq { min }
          {
            \fp_compare:nNnT
              { \l__datatool_min_fp }
               >
              { \l__datatool_tmpa_fp }
            {
              \fp_set_eq:NN
                \l__datatool_min_fp
                \l__datatool_tmpa_fp
            }
          }
          \seq_if_in:NnT \l__datatool_tmp_seq { max }
          {
            \fp_compare:nNnT
              { \l__datatool_max_fp }
               <
              { \l__datatool_tmpa_fp }
            {
              \fp_set_eq:NN
                \l__datatool_max_fp
                \l__datatool_tmpa_fp
            }
          }
        }
      }
```

Second column, if set:

```
    \exp_args:NNVV
       \dtl@getentryfromrow
       \l__datatool_item_value_tl
       \l__datatool_action_column_ii_int
       \l__datatool_map_data_row_tl
    \datatool_if_null:NF \l__datatool_item_value_tl
     {
       \__datatool_parse:N
         \l__datatool_item_value_tl
```

Check that the value is numerical.

```
        \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
        {
          \int_compare:nNnT
            { \@dtl@datatype } = { \c_datatool_currency_int }
          {
            \tl_if_empty:NT \l__datatool_item_currency_ii_tl
             {
               \tl_set_eq:NN
                 \l__datatool_item_currency_ii_tl
                 \l__datatool_datum_currency_tl
```

```
            }
          }
          \fp_set:Nn \l__datatool_tmpb_fp
           { \DTLdatumvalue { \l__datatool_item_value_tl } }
          \seq_put_right:Nx \l__datatool_tmpb_seq
           { \fp_to_decimal:N \l__datatool_tmpb_fp }
          \seq_if_in:NnT \l__datatool_tmp_seq { sum }
          {
            \fp_add:Nn \l__datatool_total_ii_fp
              \l__datatool_tmpb_fp
          }
          \seq_if_in:NnT \l__datatool_tmp_seq { min }
          {
            \fp_compare:nNnT
              { \l__datatool_min_ii_fp }
               >
              { \l__datatool_tmpb_fp }
            {
              \fp_set_eq:NN
                \l__datatool_min_ii_fp
                \l__datatool_tmpb_fp
            }
          }
          \seq_if_in:NnT \l__datatool_tmp_seq { max }
          {
            \fp_compare:nNnT
              { \l__datatool_max_ii_fp }
               <
              { \l__datatool_tmpb_fp }
            {
              \fp_set_eq:NN
                \l__datatool_max_ii_fp
                \l__datatool_tmpb_fp
            }
          }
        }
      }
    }
```

Primary return value is the number of items aggregated for the first column. This should be non-zero if no problems have occurred.

```
\int_set:Nn
  \l__datatool_count_int
  { \seq_count:N \l__datatool_tmpa_seq }
\tl_set:NV
  \l__datatool_action_return_tl
  \l__datatool_count_int
```

Set the secondary return values.

```
\__datatool_put_return_action_int:nV
 { count } \l__datatool_count_int
```

```
\__datatool_put_return_action_int:nV
 { column } \l__datatool_action_column_int
\prop_put:NnV \l__datatool_action_return_prop
 { seq } \l__datatool_tmpa_seq
\int_if_zero:nF { \l__datatool_count_int }
 {
  \seq_if_in:NnT \l__datatool_tmp_seq { min }
   {
     \fp_compare:nNnT
        { \l__datatool_min_fp }
        <
        { \c_inf_fp }
      {
        \__datatool_put_return_action_decimal:nx
         { min }
         { \fp_to_decimal:N \l__datatool_min_fp }
      }
   }
  \seq_if_in:NnT \l__datatool_tmp_seq { max }
   {
     \fp_compare:nNnT
        { \l__datatool_max_fp }
        >
        { \c_minus_inf_fp }
      {
        \__datatool_put_return_action_decimal:nx
         { max }
         { \fp_to_decimal:N \l__datatool_max_fp }
      }
   }
  \seq_if_in:NnT \l__datatool_tmp_seq { sum }
   {
     \__datatool_put_return_action_decimal:nx
      { sum }
      { \fp_to_decimal:N \l__datatool_total_fp }
```

Calculate the mean.

```
     \seq_if_in:NnT \l__datatool_tmp_seq { mean }
     {
       \fp_set:Nn \l__datatool_mean_fp
        { \l__datatool_total_fp / \l__datatool_count_int }
       \__datatool_put_return_action_decimal:nx
        { mean }
        { \fp_to_decimal:N \l__datatool_mean_fp }
```

Calculate the variance.

```
       \seq_if_in:NnT \l__datatool_tmp_seq { variance }
       {
        \fp_zero:N \l__datatool_tmpa_fp
        \seq_map_inline:Nn \l__datatool_tmpa_seq
         {
```

```
                \fp_set:Nn \l__datatool_tmpb_fp
                 {
                   ##1 - \l__datatool_mean_fp
                 }
                \fp_add:Nn \l__datatool_tmpa_fp
                 {
                    \l__datatool_tmpb_fp * \l__datatool_tmpb_fp
                 }
               }
              \fp_set:Nn \l__datatool_tmpa_fp
               { \l__datatool_tmpa_fp / \l__datatool_count_int }
              \__datatool_put_return_action_decimal:nx
               { variance }
               { \fp_to_decimal:N \l__datatool_tmpa_fp }
```

Calculate the standard deviation.

```
               \seq_if_in:NnT \l__datatool_tmp_seq { sd }
               {
                 \fp_set:Nn \l__datatool_tmpa_fp
                  { sqrt ( \l__datatool_tmpa_fp ) }
                 \__datatool_put_return_action_decimal:nx
                  { sd }
                  { \fp_to_decimal:N \l__datatool_tmpa_fp }
               }
             }
           }
         }
       }
     }
    \int_if_zero:nF { \l__datatool_action_column_ii_int }
     {
       \tl_if_empty:NF \l__datatool_item_currency_ii_tl
        {
          \tl_set_eq:NN
            \l__datatool_item_currency_tl
            \l__datatool_item_currency_ii_tl
        }
       \__datatool_put_return_action_int:nV
        { column2 } \l__datatool_action_column_ii_int
```

Number of column2 items counted:

```
       \int_set:Nn
        \l__datatool_count_int
        { \seq_count:N \l__datatool_tmpb_seq }
       \__datatool_put_return_action_int:nV
        { count2 } \l__datatool_count_int
       \prop_put:NnV \l__datatool_action_return_prop
        { seq2 } \l__datatool_tmpb_seq
       \int_if_zero:nF { \l__datatool_count_int }
        {
         \seq_if_in:NnT \l__datatool_tmp_seq { min }
          {
```

```
      \fp_compare:nNnT
        { \l__datatool_min_ii_fp }
        <
        { \c_inf_fp }
       {
        \__datatool_put_return_action_decimal:nx
         { min2 }
         { \fp_to_decimal:N \l__datatool_min_ii_fp }
       }
     }
    \seq_if_in:NnT \l__datatool_tmp_seq { max }
     {
      \fp_compare:nNnT
        { \l__datatool_max_ii_fp }
        >
        { \c_minus_inf_fp }
       {
        \__datatool_put_return_action_decimal:nx
         { max2 }
         { \fp_to_decimal:N \l__datatool_max_ii_fp }
       }
     }
    \seq_if_in:NnT \l__datatool_tmp_seq { sum }
     {
      \__datatool_put_return_action_decimal:nx
       { sum2 }
       { \fp_to_decimal:N \l__datatool_total_ii_fp }
```

Calculate the mean.

```
      \seq_if_in:NnT \l__datatool_tmp_seq { mean }
      {
        \fp_set:Nn \l__datatool_mean_fp
         { \l__datatool_total_fp / \l__datatool_count_int }
        \__datatool_put_return_action_decimal:nx
         { mean2 }
         { \fp_to_decimal:N \l__datatool_mean_fp }
```

Calculate the variance.

```
       \seq_if_in:NnT \l__datatool_tmp_seq { variance }
       {
        \fp_zero:N \l__datatool_tmpa_fp
        \seq_map_inline:Nn \l__datatool_tmpb_seq
         {
           \fp_set:Nn \l__datatool_tmpb_fp
            {
              ##1 - \l__datatool_mean_fp
            }
           \fp_add:Nn \l__datatool_tmpa_fp
            {
               \l__datatool_tmpb_fp * \l__datatool_tmpb_fp
            }
```

```
                }
              \fp_set:Nn \l__datatool_tmpa_fp
                { \l__datatool_tmpa_fp / \l__datatool_count_int }
              \__datatool_put_return_action_decimal:nx
                { variance2 }
                { \fp_to_decimal:N \l__datatool_tmpa_fp }
```

Calculate the standard deviation.

```
                \seq_if_in:NnT \l__datatool_tmp_seq { sd }
                {
                  \fp_set:Nn \l__datatool_tmpa_fp
                    { sqrt ( \l__datatool_tmpa_fp ) }
                  \__datatool_put_return_action_decimal:nx
                    { sd2 }
                    { \fp_to_decimal:N \l__datatool_tmpa_fp }
                }
              }
            }
          }
        }
      }
  }
```

Supplementary secondary return value:

```
  \__datatool_put_return_action_string:nV
   { name } \l__datatool_action_name_tl
}
```

Get row aggregates (action=row aggregate) for use in `\DTLmapdata`:

```
\cs_new:cn { __datatool_action_row ~ aggregate: }
{
  \__datatool_action_noop:Nn
    \DTLmapdata
    { row ~ aggregate: }
}
```

Action within `\DTLmapdata`:

```
\cs_new:Nn \__datatool_action_row_aggregate_op:
{
  \exp_args:NV \__datatool_action_row_aggregate:n
   \l__datatool_map_data_row_tl
}
```

Get row aggregates (action=current row aggregate) for use with `\dtlcurrentrow`:

```
\cs_new:cn { __datatool_action_current ~ row ~ aggregate: }
{
  \exp_args:NV \__datatool_action_row_aggregate:n
    \dtlcurrentrow
}
```

Underlying action of both “row aggregate” and “current row aggregate”:

```
\cs_new:Nn \__datatool_action_row_aggregate:n
```

```
{
 \__datatool_requires_dbname:T
 {
  \tl_if_empty:nT { #1 }
   {
     \PackageWarning { datatool }
      {
         Action ~ ` \l__datatool_action_tl ': ~ empty ~ row
      }
   }
```

Initialise:

```
   \fp_set_eq:NN \l__datatool_min_fp \c_inf_fp
   \fp_set_eq:NN \l__datatool_max_fp \c_minus_inf_fp
   \fp_zero:N \l__datatool_total_fp
   \tl_clear:N \l__datatool_item_currency_tl
```

List of aggregate types required. These will be stored in the options sequence:

```
   \seq_clear:N \l__datatool_action_tmp_options_seq
```

Iterate through the list of options supplied by the user:

```
   \clist_if_empty:NF \l__datatool_action_options_clist
   {
     \clist_map_inline:Nn
      \l__datatool_action_options_clist
     {
       \clist_if_in:nnTF
         { sum , mean , variance, sd , min , max } { ##1 }
        {
          \__datatool_add_action_option:n { ##1 }
        }
        {
          \PackageWarning { datatool }
           {
             Unknown ~ aggregate ~ function ~ `##1' ~ in ~
             \token_to_str:N \DTLaction
              [ options={\l__datatool_action_options_clist}, ... ]
              {` \l__datatool_action_tl ' }
           }
        }
     }
   }
```

Add dependent types if omitted:

```
   \seq_if_in:NnT
     \l__datatool_action_tmp_options_seq { sd }
    {
      \__datatool_add_action_option:n { variance }
    }
   \seq_if_in:NnT
     \l__datatool_action_tmp_options_seq { variance }
```

```
    {
      \__datatool_add_action_option:n { mean }
    }
  \seq_if_in:NnT
    \l__datatool_action_tmp_options_seq { mean }
    {
      \__datatool_add_action_option:n { sum }
    }
```

List of numeric values:

```
  \seq_clear:N \l__datatool_action_tmp_data_seq
```

Populate the column sequence `\l__datatool_action_columns_seq` with all the required columns:

```
  \__datatool_optional_columns:
```

Iterate over all identified columns (or all if no subset provided).

```
  \seq_if_empty:NTF \l__datatool_action_columns_seq
  {
    \int_step_inline:nnn
     { 1 }
     { \DTLcolumncount { \l__datatool_action_name_tl } }
     {
       \__datatool_action_row_aggregate_col:nn { ##1 } { #1 }
     }
  }
  {
   \seq_map_inline:Nn \l__datatool_action_columns_seq
    {
      \__datatool_action_row_aggregate_col:nn { ##1 } { #1 }
    }
  }
```

Primary return value is the number of numeric items. This should be non-zero if no problems have occurred.

```
  \int_set:Nn
    \l__datatool_count_int
    { \seq_count:N \l__datatool_action_tmp_data_seq }
  \tl_set:NV
    \l__datatool_action_return_tl
    \l__datatool_count_int
```

Set the secondary return values.

```
  \__datatool_put_return_action_int:nV
   { count } \l__datatool_count_int
  \__datatool_put_return_action_string:nx
   { columns }
   { \seq_use:Nn \l__datatool_action_columns_seq { , } }
  \prop_put:NnV \l__datatool_action_return_prop
   { seq } \l__datatool_action_tmp_data_seq
  \int_if_zero:nF { \l__datatool_count_int }
   {
```

```
\__datatool_if_action_option:nT { min }
 {
   \fp_compare:nNnT
      { \l__datatool_min_fp }
      <
      { \c_inf_fp }
    {
      \__datatool_put_return_action_decimal:nx
       { min }
       { \fp_to_decimal:N \l__datatool_min_fp }
    }
 }
\__datatool_if_action_option:nT { max }
 {
   \fp_compare:nNnT
      { \l__datatool_max_fp }
      >
      { \c_minus_inf_fp }
    {
      \__datatool_put_return_action_decimal:nx
       { max }
       { \fp_to_decimal:N \l__datatool_max_fp }
    }
 }
\__datatool_if_action_option:nT { sum }
 {
   \__datatool_put_return_action_decimal:nx
    { sum }
    { \fp_to_decimal:N \l__datatool_total_fp }
```

Calculate the mean.

```
\__datatool_if_action_option:nT { mean }
{
  \fp_set:Nn \l__datatool_mean_fp
   { \l__datatool_total_fp / \l__datatool_count_int }
  \__datatool_put_return_action_decimal:nx
   { mean }
   { \fp_to_decimal:N \l__datatool_mean_fp }
```

Calculate the variance.

```
\__datatool_if_action_option:nT { variance }
{
 \fp_zero:N \l__datatool_tmpa_fp
 \seq_map_inline:Nn
  \l__datatool_action_tmp_data_seq
  {
    \fp_set:Nn \l__datatool_tmpb_fp
     {
       ##1 - \l__datatool_mean_fp
     }
    \fp_add:Nn \l__datatool_tmpa_fp
```

```
            {
               \l__datatool_tmpb_fp * \l__datatool_tmpb_fp
            }
          }
         \fp_set:Nn \l__datatool_tmpa_fp
          { \l__datatool_tmpa_fp / \l__datatool_count_int }
         \__datatool_put_return_action_decimal:nx
          { variance }
          { \fp_to_decimal:N \l__datatool_tmpa_fp }
```

Calculate the standard deviation.

```
          \__datatool_if_action_option:nT { sd }
          {
            \fp_set:Nn \l__datatool_tmpa_fp
             { sqrt ( \l__datatool_tmpa_fp ) }
            \__datatool_put_return_action_decimal:nx
             { sd }
             { \fp_to_decimal:N \l__datatool_tmpa_fp }
          }
        }
      }
    }
  }
```

Supplementary secondary return values. Database name:

```
  \__datatool_put_return_action_string:nV
   { name } \dtldbname
```

Row index:

```
  \__datatool_put_return_action_int:nV
   { row } \dtlrownum
 }
}
```

Arguments: ⟨*col-idx*⟩{⟨*row markup*⟩}

Parse given column in the current row.

```
\cs_new:Nn \__datatool_action_row_aggregate_col:nn
{
  \dtl@getentryfromrow
    \l__datatool_item_value_tl { #1 } { #2 }
  \datatool_if_null:NF \l__datatool_item_value_tl
   {
     \__datatool_parse:N \l__datatool_item_value_tl
```

Check that the value is numerical.

```
      \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
      {
        \int_compare:nNnT
          { \@dtl@datatype } = { \c_datatool_currency_int }
        {
          \tl_if_empty:NT \l__datatool_item_currency_tl
           {
```

```
            \tl_set_eq:NN
              \l__datatool_item_currency_tl
              \l__datatool_datum_currency_tl
          }
        }
        \fp_set:Nn \l__datatool_tmpa_fp
         { \DTLdatumvalue { \l__datatool_item_value_tl } }
        \seq_put_right:Nx \l__datatool_action_tmp_data_seq
         { \fp_to_decimal:N \l__datatool_tmpa_fp }
        \__datatool_if_action_option:nT { sum }
        {
          \fp_add:Nn \l__datatool_total_fp
            \l__datatool_tmpa_fp
        }
        \__datatool_if_action_option:nT { min }
        {
          \fp_compare:nNnT
            { \l__datatool_min_fp }
             >
            { \l__datatool_tmpa_fp }
          {
            \fp_set_eq:NN
              \l__datatool_min_fp
              \l__datatool_tmpa_fp
          }
        }
        \__datatool_if_action_option:nT { max }
        {
          \fp_compare:nNnT
            { \l__datatool_max_fp }
             <
            { \l__datatool_tmpa_fp }
          {
            \fp_set_eq:NN
              \l__datatool_max_fp
              \l__datatool_tmpa_fp
          }
        }
      }
    }
}
```

Select a row, which will make it the current row (action=select row):

```
\cs_new:cn { __datatool_action_select ~ row: }
{
  \int_zero:N \dtlrownum
  \int_zero:N \dtlcolumnnum
```

Check if database exists:

```
  \__datatool_require_database:T
  {
```

```
    \int_if_zero:nTF \l__datatool_action_row_int
    {
```
No row index provided, so match on column.
```
     \__datatool_optional_key_xor_column_get_column:nTF
      {
```
No key or column index provided.
```
        \__datatool_action_error:nn
         {
            Missing ~ row ~ or ~ column ~ identifier ~
            ( `row' ~ or ~ `key' ~ or `column' ~ required )
         }
         {
            If ~ the ~ row ~ index ~ isn't ~ provided, ~
            \token_to_str:N \DTLaction [...]{ \l__datatool_action_tl } ~
            needs ~ the ~ `key' ~ or `column' ~ setting ~ in ~ the ~ optional ~
            argument ~ to ~ identify ~ the ~ required ~ column ~
            to ~ match on
         }
      }
      {
```
Column index available. Don't trigger an error if no match.
```
        \__datatool_get_row_for_value:VVVT
         \l__datatool_action_name_tl
         \l__datatool_action_column_int
         \l__datatool_action_value_tl
        {
          \int_set_eq:NN
            \dtlcolumnnum
            \l__datatool_action_column_int
        }
      }
      {
```
No match on column. Do nothing. User can check return value.
```
      }
    }
    {
      \int_set_eq:NN \dtlrownum \l__datatool_action_row_int
```
Row index provided, so check column and key haven't been set.
```
      \tl_if_empty:NF \l__datatool_action_key_tl
      {
       \__datatool_action_error:nn
        {
            can't ~ use ~ both ~ `key' ~ and ~ `row'. ~ Ignoring ~
            key = \l__datatool_action_key_tl
        }
        {
            Either ~ use ~ `key' ~ or ~ `row' ~ but ~ not ~ both ~
```

```
            in the optional argument of \token_to_str:N \DTLaction
            [...] { \l__datatool_action_tl }
        }
      }
      \int_if_zero:nF \l__datatool_action_column_int
      {
       \__datatool_action_error:nn
        {
            can't ~ use ~ both ~ `column' ~ and ~ `row'. ~ Ignoring ~
            column = \int_use:N \l__datatool_action_column_int
        }
        {
            Either ~ use ~ `column' ~ or ~ `row' ~ but ~ not ~ both ~
            in the optional argument of \token_to_str:N \DTLaction
            [...] { \l__datatool_action_tl }
        }
        \int_zero:N \l__datatool_action_column_int
      }
      \exp_args:NNV \dtlgetrow
       \l__datatool_action_name_tl
       \l__datatool_action_row_int
    }
```

Primary return value is the row index.

```
    \int_if_zero:nF \dtlrownum
     {
       \tl_set:NV
         \l__datatool_action_return_tl
         \dtlrownum
```

Set the secondary return values (column keys).

```
        \__datatool_map_current_row:Nn
         \l__datatool_item_value_tl
         {
           \exp_args:NNVV \@dtl@getkeyforcolumn
            \l__datatool_item_key_tl
            \l__datatool_action_name_tl
            \dtlcolumnnum
           \__datatool_put_return_action_parse:VV
             \l__datatool_item_key_tl
             \l__datatool_item_value_tl
         }
         \int_set_eq:NN
           \dtlcolumnnum
           \l__datatool_action_column_int
      }
  }
}
```

Get each entry from the current row (action=current row values):

```
\cs_new:cn { __datatool_action_current ~ row ~ values: }
```

```
{
  \__datatool_requires_dbname:T
  {
    \clist_if_empty:NTF
      \l__datatool_action_options_clist
    {
      \seq_clear:N \l__datatool_column_keys_seq
    }
    {
      \seq_set_from_clist:NN \l__datatool_column_keys_seq
        \l__datatool_action_options_clist
    }
```

Populate the column sequence `\l__datatool_action_columns_seq` with all the required columns:

```
    \__datatool_optional_columns:
    \seq_if_empty:NTF \l__datatool_action_columns_seq
     {
```

No columns supplied so get all column values.

```
       \int_step_inline:nn { \DTLcolumncount { \dtldbname } }
        {
          \dtlgetentryfromcurrentrow
            \l__datatool_item_value_tl
            { ##1 }
          \datatool_if_null:NF \l__datatool_item_value_tl
           {
             \@dtl@getkeyforcolumn
               \l__datatool_item_key_tl
               \dtldbname
               { ##1 }
             \__datatool_put_return_action_parse:VV
               \l__datatool_item_key_tl
               \l__datatool_item_value_tl
           }
        }
    }
    {
```

Iterate over subset

```
       \seq_map_inline:Nn \l__datatool_action_columns_seq
        {
          \dtlgetentryfromcurrentrow
            \l__datatool_item_value_tl
            { ##1 }
          \datatool_if_null:NF
            \l__datatool_item_value_tl
           {
             \@dtl@getkeyforcolumn
               \l__datatool_item_key_tl
               \dtldbname
```

```
            { ##1 }
          \__datatool_put_return_action_parse:VV
            \l__datatool_item_key_tl
            \l__datatool_item_value_tl
        }
      }
    }
  }
```

Primary return value is the total number of entries found in the current row.

```
  \tl_set:Nx
    \l__datatool_action_return_tl
    { \prop_count:N \l__datatool_action_return_prop }
}
```

Options for action=find. Match function:

```
\cs_new:Nn \__datatool_action_find_if:T { #1 }
```

Search direction:

```
\bool_new:N \__datatool_action_find_asc_bool
```

Select row if match found:

```
\bool_new:N \__datatool_action_find_select_bool
\keys_define:nn { datatool / action / find }
{
  select .bool_set:N = \__datatool_action_find_select_bool ,
  direction .choice: ,
  direction / ascending .code:n =
   {
      \bool_set_true:N \__datatool_action_find_asc_bool
   } ,
  direction / asc .code:n =
   {
      \bool_set_true:N \__datatool_action_find_asc_bool
   } ,
  direction / descending .code:n =
   {
      \bool_set_false:N \__datatool_action_find_asc_bool
   } ,
  direction / desc .code:n =
   {
      \bool_set_false:N \__datatool_action_find_asc_bool
   } ,
  direction .value_required:n = true ,
  inline .cs_set:Np = \__datatool_action_find_if:T #1 ,
  inline .value_required:n = true ,
  function .code =
   {
     \cs_set_eq:NN \__datatool_action_find_if:T #1
   },
  function .value_required:n = true ,
```

```
}
```

Find a row matching criteria given in options (action=find):

```
\cs_new:cn { __datatool_action_find: }
{
```

Check if database exists:

```
  \__datatool_require_database:T
  {
    \__datatool_forbid_key_and_column:n
     {
       The ~ action ~ ` \l__datatool_action_tl ' ~ should ~ have ~
       cs = column-key ~ comma ~ separated ~ list ~ in ~ the ~
      `assign' ~ setting ~ to ~ reference ~ column ~ values
     }
    \@DTLifdbempty { \l__datatool_action_name_tl }
     { }
     {
```

The row and row2 settings may be used to limit the search range.

```
      \__datatool_optional_row:NF \l__datatool_action_row_int
       {
         \int_set_eq:NN \l__datatool_action_row_int \c_one_int
       }
      \__datatool_optional_row:NF \l__datatool_action_row_ii_int
       {
         \int_set:Nn \l__datatool_action_row_ii_int
          { \DTLrowcount { \l__datatool_action_name_tl } }
       }
```

Set options. The default is to select the first row. This makes it easier to select a row simply by index.

```
      \cs_set_eq:NN \__datatool_action_find_if:T \use:n
      \bool_set_true:N \__datatool_action_find_asc_bool
      \bool_set_false:N \__datatool_action_find_select_bool
      \keys_set:nV { datatool / action / find }
        \l__datatool_action_options_clist
```

Gather assignment information, if provided.

```
      \__datatool_get_placeholder_assign:
```

If ascending, row should be less than row2 value.

```
      \bool_if:NTF \__datatool_action_find_asc_bool
       {
         \int_compare:nNnT
           { \l__datatool_action_row_ii_int }
            <
           { \l__datatool_action_row_int }
          {
            \datatool_swap_ints:NN
              \l__datatool_action_row_int
              \l__datatool_action_row_ii_int
```

```
          }
```
Iterate over all rows in ascending order.
```
        \int_until_do:nn
          {
            \l__datatool_action_row_int
              >
            \l__datatool_action_row_ii_int
          }
          {
            \__datatool_action_find_loop_body:
            \int_incr:N \l__datatool_action_row_int
          }
        }
        {
```
If descending, row2 should be less than row value.
```
          \int_compare:nNnT
            { \l__datatool_action_row_ii_int }
             >
            { \l__datatool_action_row_int }
            {
              \datatool_swap_ints:NN
                \l__datatool_action_row_int
                \l__datatool_action_row_ii_int
            }
```
Iterate over all rows in descending order.
```
        \int_until_do:nn
          {
            \l__datatool_action_row_int
              <
            \l__datatool_action_row_ii_int
          }
          {
            \__datatool_action_find_loop_body:
            \int_decr:N \l__datatool_action_row_int
          }
        }
      }
    }
  }
```
Loop body for ‘find’ action (but doesn’t update loop row counter):
```
 \cs_new:Nn \__datatool_action_find_loop_body:
 {
```
Get specs for this row.
```
   \__datatool_get_row:vVN
     { dtldb@ \l__datatool_action_name_tl }
     \l__datatool_action_row_int
    \l__datatool_action_tmpa_tl
```

Perform assignments.

```
   \exp_args:NV \__datatool_do_action_assignments:n
    \l__datatool_action_tmpa_tl
```

Perform test.

```
   \__datatool_action_find_if:T
    {
```

The primary return value is the row index, if successful

```
      \tl_set:Nx
        \l__datatool_action_return_tl
        { \int_use:N \l__datatool_action_row_int }
```

Set secondary return values.

```
       \__datatool_set_return_from_row:Vn
        \l__datatool_action_tmpa_tl
        { \l__datatool_action_name_tl }
```

Select the row if applicable.

```
       \bool_if:NT \__datatool_action_find_select_bool
        {
          \exp_args:NNV \dtlgetrow
           \l__datatool_action_name_tl
           \l__datatool_action_row_int
        }
```

Break out of loop.

```
      \int_set_eq:NN
        \l__datatool_action_row_int
        \l__datatool_action_row_ii_int
    }
 }
```

Sort data (action=sort):

```
 \cs_new:cn { __datatool_action_sort: }
 {
   \__datatool_require_database:T
    {
```

Check that the user hasn't accidentally used 'columns' or 'keys' instead of 'assign':

```
      \clist_if_empty:NF
         \l__datatool_action_columns_clist
       {
         \__datatool_action_error:n
          {
            `columns' ~ action ~ setting ~ not ~ available. ~ Ignoring
          }
       }
      \clist_if_empty:NF
         \l__datatool_action_keys_clist
       {
         \__datatool_action_error:n
```

```
          {
            `keys' ~ action ~ setting ~ not ~ available. ~ Ignoring
          }
        }
```

Check that the user has provided ‘assign’:

```
      \clist_if_empty:NTF
        \l__datatool_action_assign_clist
       {
        \__datatool_action_error:nn
         {
           Missing ~ `assign' ~ setting. ~
           Unable ~ to ~ sort ~ without ~ any ~
           sort ~ criteria!
         }
         {
           The ~ `assign' ~ setting ~ corresponds ~ to ~
           the ~ final ~ `criteria' ~ argument ~ of ~
           \token_to_str:N \DTLsortdata
         }
       }
       {
        \seq_clear:N \l__datatool_wordlist_seq
        \__datatool_db_sort:VVV
          \l__datatool_action_options_clist
          \l__datatool_action_name_tl
          \l__datatool_action_assign_clist
```

The primary return value is the element count of word list. This will be empty if the sort failed. If successful, it should equal the number of rows.

```
        \seq_if_empty:NF \l__datatool_wordlist_seq
         {
           \tl_set:Nx \l__datatool_action_return_tl
            { \seq_count:N \l__datatool_wordlist_seq }
```

Set the secondary return values:

```
           \__datatool_put_return_action_int:nx
            { rows }
            { \DTLrowcount { \l__datatool_action_name_tl } }
           \__datatool_put_return_action_int:nx
            { columns }
            { \DTLcolumncount { \l__datatool_action_name_tl } }
         }
       }
    }
```

The name is always set as a secondary return value:

```
   \__datatool_put_return_action_string:nV
    { name } \l__datatool_action_name_tl
 }
```

12.3 Defining New Databases

As from v2.0, the internal structure of the database has changed to make it more efficient.[1] The database is now stored in a token register instead of a macro. Each row is represented as:

\db@row@elt@w \db@row@id@w ⟨*row idx*⟩\db@row@id@end@ ⟨*column data*⟩ \db@row@id@w ⟨*row idx*⟩\db@row@id@end@ \db@row@elt@end@

where ⟨*row idx*⟩ is the row index and ⟨*column data*⟩ is the data for each column in the row. Each column for a given row is stored as:

\db@col@id@w ⟨*column idx*⟩\db@col@id@end@ \db@col@elt@w ⟨*value*⟩\db@col@elt@end@ \db@col@id@w ⟨*column idx*⟩\db@col@id@end@

where ⟨*column idx*⟩ is the column index and ⟨*value*⟩ is the entry for the given column and row.

Note that LaTeX3 syntax isn't used. Old dbtex files still need to be supported.

Each row only has an associated index, but columns have a unique identifying key as well as an associated index. Columns also have an associated data type which may be: 0 (column contains strings), 1 (column contains integers), 2 (column contains real numbers), 3 (column contains currency) or ⟨*empty*⟩ (column contains no data). Since the key sometimes has to be expanded, a header is also available in the event that the user wants to use \DTLdisplaydb or \DTLdisplaylongdb and requires a column header that would cause problems if used as a key. The general column information is stored in a token register where each column has information stored in the form:

\db@plist@elt@w \db@col@id@w ⟨*index*⟩\db@col@id@end@ \db@key@id@w ⟨*key*⟩\db@key@id@end@ \db@type@id@w ⟨*type*⟩\db@type@id@end@ \db@header@id@w ⟨*type*⟩\db@header@id@end@ \db@col@id@w ⟨*index*⟩\db@col@id@end@ \db@plist@elt@end@

The column name (⟨*key*⟩) is mapped to the column index using \dtl@ci@⟨*db*⟩@⟨*key*⟩ where ⟨*db*⟩ is the database name.

__datatool_column_markup:nnnn{⟨*column idx*⟩}{⟨*key*⟩}{⟨*type*⟩}{⟨*header*⟩}

Expands to the column markup for the given column.

```
\cs_new:Nn \__datatool_column_markup:nnnn
{
  \__datatool_column_markup_full:nnnn
   { #1 } { #2 } { #3 } { \exp_not:n { #4 } }
}
```

NB don't include x variant or it won't expand.

```
\cs_generate_variant:Nn
  \__datatool_column_markup:nnnn
  { VVVV, VVVn, VnVn, VnVV, Vnnn, nVVV, nVnV, VnnV }
```

```
\cs_new:Nn \__datatool_column_markup_full:nnnn
{
```

[1] Thanks to Morten Høgholm for the suggestion.

Start of column block.

```
   \exp_not:N \db@plist@elt@w
```

Column index

```
     \exp_not:N \db@col@id@w
       #1
     \exp_not:N \db@col@id@end@
```

Column key

```
     \exp_not:N \db@key@id@w
       #2
     \exp_not:N \db@key@id@end@
```

Column type

```
     \exp_not:N \db@type@id@w
       #3
     \exp_not:N \db@type@id@end@
```

Column header

```
     \exp_not:N \db@header@id@w
       #4
     \exp_not:N \db@header@id@end@
```

Column index

```
     \exp_not:N \db@col@id@w
       #1
     \exp_not:N \db@col@id@end@
```

End of Column block

```
   \exp_not:N \db@plist@elt@end@
}
```

`\__datatool_row_markup:nn{`⟨*row idx*⟩`}{`⟨*content*⟩`}`

Expands to the row markup for the given row.

```
\cs_new:Nn \__datatool_row_markup:nn
{
  \__datatool_row_markup_full:nn
   { #1 } { \exp_not:n { #2 } }
}
```

NB don’t include x variant or it won’t expand.

```
\cs_generate_variant:Nn
  \__datatool_row_markup:nn
  { vn , Vn , vV, VV, vv , nV }
```

Don’t inhibit expansion:

```
\cs_new:Nn \__datatool_row_markup_full:nn
{
```

Start of row block.

```
  \exp_not:N \db@row@elt@w
```

Row ID marker:

```
    \exp_not:N \db@row@id@w
      #1 % row ID
    \exp_not:N \db@row@id@end@
```

Content:

```
        #2
```

Matching end row ID marker:

```
    \exp_not:N \db@row@id@w
      #1  % row ID
    \exp_not:N \db@row@id@end@
```

End of row block

```
  \exp_not:N \db@row@elt@end@
 }
 \cs_generate_variant:Nn
  \__datatool_row_markup_full:nn
  { vn }
```

`\__datatool_row_element_markup:nn{`⟨*col idx*⟩`}`
`{`⟨*content*⟩`}`

Expands to the element markup for the given row but prevents content from expanding.

```
 \cs_new:Nn \__datatool_row_element_markup:nn
 {
   \__datatool_row_element_markup_full:nn
    { #1 } { \exp_not:n { #2 } }
 }
```

NB don't include x variant or it won't expand.

```
 \cs_generate_variant:Nn
   \__datatool_row_element_markup:nn
   { vn, Vn , vV , VV, vv, nV }
```

Don't inhibit expansion of content:

```
 \cs_new:Nn \__datatool_row_element_markup_full:nn
 {
```

Column ID:

```
   \exp_not:N \db@col@id@w
     #1
   \exp_not:N \db@col@id@end@
   \exp_not:N \db@col@elt@w
```

Content:

```
     #2
   \exp_not:N \db@col@elt@end@
```

Matching end of column ID:

```
   \exp_not:N \db@col@id@w
     #1
```

```
    \exp_not:N \db@col@id@end@
  }
```

\DTLnewdb

\DTLnewdb{⟨*db name*⟩}

Initialises a database called ⟨*name*⟩.

```
\NewDocumentCommand \DTLnewdb { m }
{
```

Check if there is already a database with this name.

```
  \DTLifdbexists{#1}
  {
     \PackageError{datatool}{Database ~ `#1' ~ already ~ exists}{}
  }
  {
    \__datatool_new_db:n { #1 }
  }
}
```

New registers are global. So new databases will always be globally defined.

```
\cs_new:Nn \__datatool_new_db:n
{%
```

Define new database. Add information message if in verbose mode.

```
  \dtl@message {Creating ~ database ~ `#1'}
```

Define token register used to store the contents of the database.

```
  \exp_args:Nc \newtoks { dtldb @ #1 }
```

Define token register used to store the column header information.

```
  \exp_args:Nc \newtoks { dtlkeys @ #1 }
```

Define count register used to store the row count.

```
  \int_new:c { dtlrows @ #1 }
```

Define count register used to store the column count.

```
  \int_new:c { dtlcols @ #1 }
}
```

\DTLcleardb

\DTLcleardb{⟨*db name*⟩}

Clears the database. (Makes it empty, but still defined.)

```
\NewDocumentCommand \DTLcleardb { m }
{
  \bool_if:NTF \l__datatool_db_global_bool
  {
    \DTLgcleardb { #1 }
  }
```

```
  {
    \DTLifdbexists { #1 }
    {
      \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
      {
        \cs_undefine:c { dtl@ci@ #1 @ \@dtl@key }
      }
      \__datatool_token_register_set:cn { dtldb@ #1 } { }
      \__datatool_token_register_set:cn { dtlkeys@ #1 } { }
      \int_zero:c { dtlrows@ #1 }
      \int_zero:c { dtlcols@ #1 }
    }
    {
      \PackageError { datatool }
      {
         Can't ~ clear ~ database ~ `#1': ~
         database ~ doesn't ~ exist
      }
      { }
    }
  }
}
```

\DTLdeletedb

\DTLdeletedb{⟨*db name*⟩}

Deletes a database.

```
\NewDocumentCommand \DTLdeletedb { m }
{
  \bool_if:NTF \l__datatool_db_global_bool
  {
    \DTLgdeletedb { #1 }
  }
  {
    \DTLifdbexists {#1}
    {
      \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
      {
        \cs_undefine:c { dtl@ci@#1@\@dtl@key }
      }
      \cs_undefine:c { \csname dtldb@#1 }
      \cs_undefine:c { dtlkeys@#1 }
      \cs_undefine:c { dtlrows@#1 }
      \cs_undefine:c {  dtlcols@#1 }
    }
    {
      \PackageError{datatool}{Can't ~ delete ~ database ~ `#1': ~
         database ~ doesn't ~ exist}{}%
    }
```

```
  }
}
```

\DTLgnewdb

\DTLgnewdb{⟨*db name*⟩}

Since new registers are always globally defined, this is equivalent to \DTLnewdb.

```
\newcommand\DTLgnewdb{\DTLnewdb}
```

\DTLgdeletedb

\DTLgdeletedb{⟨*db name*⟩}

Deletes a database. (Global version.)

```
\NewDocumentCommand \DTLgdeletedb { m } {
  \DTLifdbexists { #1 }
  {
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {
      \cs_undefine:c { dtl@ci@ #1 @ \@dtl@key }
    }
    \cs_undefine:c { dtldb@#1 }
    \cs_undefine:c { dtlkeys@#1 }
    \cs_undefine:c { dtlrows@#1 }
    \cs_undefine:c { dtlcols@#1 }
  }
  {
    \PackageError {datatool}
     {
       Can't ~ delete ~ database ~ `#1': ~
       database ~ doesn't ~ exist
     }
     { }
  }
}
```

\DTLgcleardb

\DTLgcleardb{⟨*db name*⟩}

Clears the database. (Global version.)

```
\NewDocumentCommand \DTLgcleardb { m }
{
  \DTLifdbexists { #1 }
  {
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {
      \cs_undefine:c { dtl@ci@ #1 @ \@dtl@key }
    }
    \__datatool_token_register_gset:cn { dtldb@ #1 } { }
```

```
    \__datatool_token_register_gset:cn { dtlkeys@ #1 } { }
    \int_gzero:c { dtlrows@ #1 }
    \int_gzero:c { dtlcols@ #1 }
  }
  {
    \PackageError {datatool}
    {
      Can't ~ clear ~ database ~ `#1': ~
       database ~ doesn't ~ exist
    }
    { }
  }
}
```

\DTLrowcount

\DTLrowcount{⟨*db name*⟩}

The number of rows in the database called ⟨*db name*⟩. (Doesn't check if database exists.)

```
\newcommand*{\DTLrowcount}[1]{
 \int_use:c { dtlrows@ #1 }
}
```

\DTLcolumncount

\DTLcolumncount{⟨*db name*⟩}

The number of columns in the database called ⟨*db name*⟩. (Doesn't check if database exists.)

```
\newcommand*{\DTLcolumncount}[1]{
  \int_use:c { dtlcols@ #1 }
}
```

Syntax: {⟨*db-name*⟩}{⟨*not empty*⟩}{⟨*empty*⟩}{⟨*no exists*⟩}

```
\cs_new:Nn \datatool_db_state:nnnn
{
  \DTLifdbexists { #1 }
  { \@DTLifdbempty { #1 } { #3 } { #2 } }
  { #4 }
}
```

\DTLifdbempty

\DTLifdbempty{⟨*name*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Check if named database is empty (i.e. no rows have been added).

```
\NewDocumentCommand \DTLifdbempty { m +m +m }
{
  \DTLifdbexists { #1 }
```

```
  { \@DTLifdbempty { #1 } { #2 } { #3 } }
  {
    \PackageError {datatool}
     {
       \token_to_str:N \DTLifdbempty : ~
       Can't ~ check ~ if ~ database ~ `#1' ~ is ~ empty: ~
       database ~ doesn't ~ exist
     }
     { }
  }
}
```

\@DTLifdbempty

\@sDTLifdbempty{⟨*name*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Check if named existing database is empty. (No check performed to determine if the database exists.)

```
\newcommand{\@DTLifdbempty}[3]{%
  \int_if_zero:nTF
    { \int_use:c { dtlrows@ #1 } }
   { #2 } { #3 }
}
```

\DTLnewrow

\DTLnewrow{⟨*db name*⟩}

Add a new row to named database. The starred version doesn't check for the existence of the database.

```
\NewDocumentCommand \DTLnewrow { s m }
{
  \IfBooleanTF { #1 }
   { \@sDTLnewrow { #2 } }
   { \@DTLnewrow { #2 } }
}
```

\@DTLnewrow

\@DTLnewrow{⟨*db name*⟩}

Add a new row to named database. (Checks for the existence of the database.)

```
\newcommand*{\@DTLnewrow}[1]{
 \DTLifdbexists { #1 }
  { \@sDTLnewrow { #1 } }
  {
    \PackageError { datatool }
    {
      \token_to_str:N \DTLnewrow : ~
      Can't ~ add ~ new ~ row ~ to ~ database ~ `#1': ~
```

```
        database ~ doesn't ~ exist
      }
      { }
    }
  }
```

\@sDTLnewrow

\@sDTLnewrow{⟨*db name*⟩}

Add a new row to named existing database. (No check performed to determine if the database exists.) Version 3.0: check global boolean. Note that pre 3.0 this command only made a global change.

```
\newcommand*{\@sDTLnewrow}[1]{%
  \bool_if:NTF \l__datatool_db_global_bool
  {
```

Globally increment row count.

```
    \int_gincr:c { dtlrows@#1 }
  }
  {
```

Locally increment row count.

```
    \int_incr:c { dtlrows@#1 }
  }
```

Append an empty row to the database

```
  \__datatool_dtldb_put_right:nx
    { #1 }
   {
     \__datatool_row_markup:vn
      { dtlrows@#1 } % row count
      { }
   }
```

Display message on terminal and log file if in verbose mode.

```
  \dtl@message
    { New ~ row ~ added ~ to ~ database ~ `#1' }
}
```

\dtlcolumnnum Count register to keep track of column index.

```
\newcount\dtlcolumnnum
```

\dtlrownum Count register to keep track of row index.

```
\newcount\dtlrownum
```

\DTLifhaskey

\DTLifhaskey⟨*db name*⟩⟨*key*⟩⟨*true part*⟩⟨*false part*⟩

Checks if the named database ⟨*db name*⟩ has a column with label ⟨*key*⟩. If column exists, do ⟨*true part*⟩ otherwise do ⟨*false part*⟩. The starred version doesn't check if the named database exists.

```
\NewDocumentCommand \DTLifhaskey { s m m m m }
{
  \IfBooleanTF { #1 }
  {
    \@sDTLifhaskey { #2 } { #3 } { #4 } { #5 }
  }
  {
    \@DTLifhaskey { #2 } { #3 } { #4 } { #5 }
  }
}
```

`\@DTLifhaskey` Unstarred version of `\DTLifhaskey`

```
\newcommand{\@DTLifhaskey} [4]
{
  \DTLifdbexists { #1 }
  {
    \@sDTLifhaskey { #1 } { #2 } { #3 } { #4 }
  }
  {
    \PackageError { datatool }
    { Database ~ `#1' ~ doesn't ~ exist }
    { }
  }
}
```

`\@sDTLifhaskey` Starred version of `\DTLifhaskey`. This simply tests for the existence of the key to column command.

```
\newcommand{\@sDTLifhaskey}{ \datatool_if_has_key:nnTF }
```

Version 3.0: the above now defined to use the new conditional:

```
\prg_new_conditional:Npnn \datatool_if_has_key:nn #1 #2 { p, T, F, TF }
{
  \tl_if_exist:cTF { dtl@ci@ #1 @ #2 }
  {
```

Key defined

```
    \prg_return_true:
  }
  {
```

Key not defined

```
    \prg_return_false:
  }
}
```

\DTLgetcolumnindex

\DTLgetcolumnindex{⟨*cs*⟩}{⟨*db*⟩}{⟨*key*⟩}

Gets index for column with label ⟨*key*⟩ from database ⟨*db*⟩ and stores in ⟨*cs*⟩ which must be a control sequence. Unstarred version checks if database and key exist, unstarred version doesn't perform any checks.

```
\NewDocumentCommand \DTLgetcolumnindex { s m m }
{
  \IfBooleanTF { #1 }
  { \@sdtl@getcolumnindex { #2 } { #3 } }
  { \@dtl@getcolumnindex { #2 } { #3 } }
}
```

\@dtl@getcolumnindex Unstarred version of \DTLgetcolumnindex

```
\newcommand*{\@dtl@getcolumnindex} [3]
{
```

Check if database exists.

```
  \DTLifdbexists { #2 }
  {
```

Database exists. Now check if key exists.

```
    \@sDTLifhaskey { #2 } { #3 }
    {
```

Key exists so go ahead and get column index.

```
      \@sdtl@getcolumnindex { #1 } { #2 } { #3 }
    }
    {
```

Key doesn't exists in named database.

```
      \PackageError { datatool }
      { Database ~ `#2' ~ doesn't ~ contain ~ key ~ `#3' }
      { }
    }
  }
  {
```

Named database doesn't exist.

```
    \PackageError { datatool }
    { Database ~ `#2' ~ doesn't ~ exist }
    { }
  }
}
```

\@dtl@getcolumnindex Starred version of \DTLgetcolumnindex.

```
\newcommand*{\@sdtl@getcolumnindex}[3]{%
  \tl_set_eq:Nc #1 { dtl@ci@#2@#3 }
}
```

\dtlcolumnindex

\dtlcolumnindex{⟨*db*⟩}{⟨*key*⟩}

Column index corresponding to ⟨*key*⟩ in database ⟨*db*⟩. Expands to zero if the key or database doesn't exist.

```
\newcommand*{\dtlcolumnindex}[2]{%
  \tl_if_exist:cTF { dtl@ci@#1@#2 }
   { \tl_use:c { dtl@ci@#1@#2 } }
   { \c_zero_int }
}

\ExplSyntaxOff
```

\DTLgetkeyforcolumn

\DTLgetkeyforcolumn{⟨*key cs*⟩}{⟨*db*⟩}{⟨*column index*⟩}

Gets the key associated with the given column index and stores in ⟨*key cs*⟩. Unstarred version doesn't perform checks.

```
\newrobustcmd*{\DTLgetkeyforcolumn}{%
  \@ifstar\@sdtlgetkeyforcolumn\@dtlgetkeyforcolumn}

\@dtlgetkeyforcolumn

\newcommand*{\@dtlgetkeyforcolumn}[3]{%
  \DTLifdbexists{#2}%
  {%
```

Check if index is in range.

```
    \ifnum#3<1\relax
      \PackageError{datatool}{Invalid column index \number#3}{%
      Column indices start at 1}%
    \else
      \expandafter\ifnum\csname dtlcols@#2\endcsname<#3\relax
        \PackageError{datatool}{Index \number#3\space out of
        range for database `#2'}{Database `#2' only has
        \expandafter\number\csname dtlcols@#2\endcsname\space
        columns}%
      \else
        \@sdtlgetkeyforcolumn{#1}{#2}{#3}%
      \fi
    \fi
  }%
  {%
    \PackageError{datatool}{Database `#2' doesn't exists}{}%
  }%
}
```

\@sdtlgetkeyforcolumn

\@sdtlgetkeyforcolumn{⟨*key cs*⟩}{⟨*db*⟩}{⟨*column index*⟩}

Gets the key associated with the given column index and stores in ⟨*key cs*⟩

```
\newcommand*{\@sdtlgetkeyforcolumn}[3]{%
  \edef\@dtl@dogetkeyforcolumn{\noexpand\@dtl@getkeyforcolumn
    {\noexpand#1}{#2}{\number#3}}%
  \@dtl@dogetkeyforcolumn
}
```

`\@dtl@getkeyforcolumn` Column index must be fully expanded before use.

```
\newcommand*{\@dtl@getkeyforcolumn}[3]{%
  \def\@dtl@get@keyforcolumn##1% before stuff
    \db@plist@elt@w% start of block
    \db@col@id@w #3\db@col@id@end@% index
    \db@key@id@w ##2\db@key@id@end@% key
    \db@type@id@w ##3\db@type@id@end@% data type
    \db@header@id@w ##4\db@header@id@end@% header
    \db@col@id@w #3\db@col@id@end@% index
    \db@plist@elt@end@% end of block
    ##5\q@nil{\def#1{##2}}%
  \edef\@dtl@tmp{\expandafter\the\csname dtlkeys@#2\endcsname}%
  \expandafter\@dtl@get@keyforcolumn\@dtl@tmp
    \db@plist@elt@w% start of block
    \db@col@id@w #3\db@col@id@end@ %index
    \db@key@id@w \@nil\db@key@id@end@% key
    \db@type@id@w \db@type@id@end@% data type
    \db@header@id@w \db@header@id@end@% header
    \db@col@id@w #3\db@col@id@end@% index
    \db@plist@elt@end@% end of block
    \q@nil
}
```

Define some commands to indicate the various data types a database may contain. Version 3.0: note that datatool-base now has constants but the unknown type is -1 in that case.

`\DTLunsettype` Unknown data type. (All entries in the column are blank so the type can't be determined.)

```
\newcommand*\DTLunsettype{}
```

`\DTLstringtype` Data type representing strings.

```
\newcommand*\DTLstringtype{0}
```

`\DTLinttype` Data type representing integers.

```
\newcommand*\DTLinttype{1}
```

`\DTLrealtype` Data type representing real numbers.

```
\newcommand*\DTLrealtype{2}
```

`\DTLcurrencytype` Data type representing currency.

```
\newcommand*\DTLcurrencytype{3}
```

\DTLgetdatatype

\DTLgetdatatype{⟨*cs*⟩}{⟨*db*⟩}{⟨*key*⟩}

Gets data type associated with column labelled ⟨*key*⟩ in database ⟨*db*⟩ and stores in ⟨*cs*⟩. Type may be: ⟨*empty*⟩ (unset), 0 (string), 1 (int), 2 (real), 3 (currency). Unstarred version checks if the database and key exist, starred version doesn't.

```
\newrobustcmd*{\DTLgetdatatype}{%
  \@ifstar\@sdtlgetdatatype\@dtlgetdatatype
}
```

\@dtlgetdatatype Unstarred version of \DTLgetdatatype.

```
\newcommand*{\@dtlgetdatatype}[3]{%
```

Check if database exists.

```
  \DTLifdbexists{#2}%
  {%
```

Check if key exists in this database.

```
    \@sDTLifhaskey{#2}{#3}%
    {%
```

Get data type for this database and key.

```
      \@sdtlgetdatatype{#1}{#2}{#3}%
    }%
    {%
```

Key doesn't exist in this database.

```
      \PackageError{datatool}{Key `#3' undefined in database `#2'}{}%
    }%
  }%
  {%
```

Database doesn't exist.

```
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
```

\@sdtlgetdatatype Starred version of \DTLgetdatatype. This ensures that the key is fully expanded before being passed to \@dtl@getdatatype.

```
\newcommand*{\@sdtlgetdatatype}[3]{%
  \edef\@dtl@dogetdata{\noexpand\@dtl@getdatatype{\noexpand#1}%
   {\expandafter\the\csname dtlkeys@#2\endcsname}%
   {\dtlcolumnindex{#2}{#3}}}%
  \@dtl@dogetdata
}
```

\@dtl@getdatatype

\@dtl@getdatatype{⟨*cs*⟩}{⟨*data specs*⟩}{⟨*column index*⟩}

Column index must be expanded.

```
\newcommand*{\@dtl@getdatatype}[3]{%
  \def\@dtl@get@keydata##1% stuff before
    \db@plist@elt@w% start of key block
     \db@col@id@w #3\db@col@id@end@% column index
      \db@key@id@w ##2\db@key@id@end@% key id
      \db@type@id@w ##3\db@type@id@end@% data type
      \db@header@id@w ##4\db@header@id@end@% header
     \db@col@id@w #3\db@col@id@end@% column index
    \db@plist@elt@end@% end of key block
    ##5% stuff afterwards
    \q@nil{\def#1{##3}}%
  \@dtl@get@keydata#2\q@nil
}

\ExplSyntaxOn
```

\@dtl@getprops

\@dtl@getprops{⟨*key cs*⟩}{⟨*type cs*⟩}{⟨*header toks*⟩}{⟨*before toks*⟩}{⟨*after toks*⟩}{⟨*data specs*⟩}{⟨*column index*⟩}

Column index must be expanded.

```
\newcommand*{\@dtl@getprops}[7]{%
  \__datatool_get_keydata:NNNNNnn #1 #2 #3 #4 #5 { #6 } { #7 }
}
\cs_new:Nn \__datatool_get_keydata:NNNNNnn
{
  \cs_set:Npn \__datatool_get_keydata:w
     ##1 % stuff before
    \db@plist@elt@w % start of key block
     \db@col@id@w #7\db@col@id@end@ % column index
      \db@key@id@w ##2\db@key@id@end@ % key id
      \db@type@id@w ##3\db@type@id@end@ % data type
      \db@header@id@w ##4\db@header@id@end@ % header
     \db@col@id@w #7\db@col@id@end@ % column index
    \db@plist@elt@end@ % end of key block
    ##5% stuff afterwards
    \q_nil
    {
      \tl_set:Nn #1 { ##2 } % key
      \tl_set:Nn #2 { ##3 } % data type
      \__datatool_token_register_set:Nn #3 { ##4 } % header
      \__datatool_token_register_set:Nn #4 { ##1 } % before stuff
      \__datatool_token_register_set:Nn #5 { ##5 } % after stuff
    }
  \__datatool_get_keydata:w #6 \q_nil
}
\cs_generate_variant:Nn  \__datatool_get_keydata:NNNNNnn
 { NNNNNvV , NNNNNve }
```

Provide a newer L3 alternative. NB unlike `\@dtl@getprops` this uses token list variables not registers. Syntax: ⟨*key cs*⟩⟨*type cs*⟩⟨*header tl var*⟩⟨*before tl var*⟩⟨*after tl var*⟩`{`⟨*data specs*⟩`}{`⟨*column index*⟩`}`

```
\cs_new:Nn \__datatool_get_props:NNNNNnn
{
  \cs_set:Npn \__datatool_get_keydata:w
     ##1 % stuff before
    \db@plist@elt@w % start of key block
     \db@col@id@w #7\db@col@id@end@ % column index
      \db@key@id@w ##2\db@key@id@end@ % key id
      \db@type@id@w ##3\db@type@id@end@ % data type
      \db@header@id@w ##4\db@header@id@end@ % header
     \db@col@id@w #7\db@col@id@end@ % column index
    \db@plist@elt@end@ % end of key block
    ##5 % stuff afterwards
    \q_nil
    {
      \tl_set:Nn #1 { ##2 } % key
      \tl_set:Nn #2 { ##3 } % data type
      \tl_set:Nn #3 { ##4 } % header
      \tl_set:Nn #4 { ##1 } % before stuff
      \tl_set:Nn #5 { ##5 } % after stuff
    }
  \__datatool_get_keydata:w #6 \q_nil
}
\cs_generate_variant:Nn \__datatool_get_props:NNNNNnn
 { NNNNNvx, NNNNNvV, NNNNNvn }
```

Get all meta data for the given column where the before and after stuff isn't required. Automatically sets scratch variables.

Syntax: `{`⟨*col specs*⟩`}{`⟨*col idx*⟩`}`

Sets: `\l__datatool_item_key_tl`, `\l__datatool_item_head_tl` and `\l__datatool_item_type_int`.

```
\cs_new:Nn \__datatool_get_col_data:nn
{
  \tl_if_empty:nTF { #1 }
  {
    \tl_set:Nn \l__datatool_item_head_tl { \q_no_value }
    \int_set_eq:NN \l__datatool_item_type_int \c_datatool_unknown_int
    \tl_set:Nn \l__datatool_item_key_tl { \q_no_value }
  }
  {
    \cs_set:Npn \__datatool_get_col_data:w ##1% stuff before
      \db@plist@elt@w% start of key block
       \db@col@id@w #2\db@col@id@end@% column index
        \db@key@id@w ##2\db@key@id@end@% key id
        \db@type@id@w ##3\db@type@id@end@% data type
        \db@header@id@w ##4\db@header@id@end@% header
       \db@col@id@w #2\db@col@id@end@% column index
```

```
      \db@plist@elt@end@% end of key block
      ##5% stuff afterwards
      \q_nil
      {
        \tl_set:Nn \l__datatool_item_key_tl { ##2 }
        \tl_set:Nn \l__datatool_item_head_tl { ##4 }
        \tl_if_empty:nTF { ##3 }
        {
          \int_set_eq:NN \l__datatool_item_type_int \c_datatool_unknown_int
        }
        {
          \int_set:Nn \l__datatool_item_type_int { ##3 }
        }
      }
    \__datatool_get_col_data:w #1
```

In case the column isn't defined:

```
      \db@plist@elt@w% start of key block
       \db@col@id@w #2\db@col@id@end@% column index
        \db@key@id@w \q_no_value \db@key@id@end@% key id
        \db@type@id@w \db@type@id@end@% data type
        \db@header@id@w \q_no_value \db@header@id@end@% header
       \db@col@id@w #2\db@col@id@end@% column index
      \db@plist@elt@end@% end of key block
    \q_nil
  }
}
\cs_generate_variant:Nn \__datatool_get_col_data:nn { vV, vn }
```

Similar but only type required.

```
\cs_new:Nn \__datatool_get_col_type:nn
{
  \tl_if_empty:nTF { #1 }
  {
    \int_set_eq:NN \l__datatool_item_type_int \c_datatool_unknown_int
  }
  {
    \cs_set:Npn \__datatool_get_col_type:w ##1% stuff before
      \db@plist@elt@w% start of key block
       \db@col@id@w #2\db@col@id@end@% column index
        \db@key@id@w ##2\db@key@id@end@% key id
        \db@type@id@w ##3\db@type@id@end@% data type
        \db@header@id@w ##4\db@header@id@end@% header
       \db@col@id@w #2\db@col@id@end@% column index
      \db@plist@elt@end@% end of key block
      ##5% stuff afterwards
      \q_nil
      {
        \tl_if_empty:nTF { ##3 }
        {
          \int_set_eq:NN \l__datatool_item_type_int \c_datatool_unknown_int
```

```
        }
        {
          \int_set:Nn \l__datatool_item_type_int { ##3 }
        }
      }
    \__datatool_get_col_type:w #1
```

In case the column isn't defined:

```
      \db@plist@elt@w% start of key block
       \db@col@id@w #2\db@col@id@end@% column index
        \db@key@id@w \q_no_value \db@key@id@end@% key id
        \db@type@id@w \db@type@id@end@% data type
        \db@header@id@w \q_no_value \db@header@id@end@% header
       \db@col@id@w #2\db@col@id@end@% column index
      \db@plist@elt@end@% end of key block
    \q_nil
  }
}
\cs_generate_variant:Nn \__datatool_get_col_type:nn { vn, vV, vv }
```

Only type and header required:

```
\cs_new:Nn \__datatool_get_col_type_header:nn
{
  \tl_if_empty:nTF { #1 }
  {
    \tl_set:Nn \l__datatool_item_head_tl { \q_no_value }
    \int_set_eq:NN \l__datatool_item_type_int \c_datatool_unknown_int
  }
  {
    \cs_set:Npn \__datatool_get_col_type_header:w ##1% stuff before
      \db@plist@elt@w% start of key block
       \db@col@id@w #2\db@col@id@end@% column index
        \db@key@id@w ##2\db@key@id@end@% key id
        \db@type@id@w ##3\db@type@id@end@% data type
        \db@header@id@w ##4\db@header@id@end@% header
       \db@col@id@w #2\db@col@id@end@% column index
      \db@plist@elt@end@% end of key block
      ##5% stuff afterwards
      \q_nil
      {
        \tl_set:Nn \l__datatool_item_head_tl { ##4 }
        \tl_if_empty:nTF { ##3 }
        {
          \int_set_eq:NN \l__datatool_item_type_int \c_datatool_unknown_int
        }
        {
          \int_set:Nn \l__datatool_item_type_int { ##3 }
        }
      }
    \__datatool_get_col_type_header:w #1
```

In case the column isn't defined:

```
       \db@plist@elt@w% start of key block
        \db@col@id@w #2\db@col@id@end@% column index
         \db@key@id@w \q_no_value \db@key@id@end@% key id
         \db@type@id@w \db@type@id@end@% data type
         \db@header@id@w \q_no_value \db@header@id@end@% header
        \db@col@id@w #2\db@col@id@end@% column index
       \db@plist@elt@end@% end of key block
     \q_nil
   }
 }
```

Only type and key required:

```
 \cs_new:Nn \__datatool_get_col_type_key:nn
 {
   \tl_if_empty:nTF { #1 }
   {
     \tl_set:Nn \l__datatool_item_key_tl { \q_no_value }
     \int_set_eq:NN \l__datatool_item_type_int \c_datatool_unknown_int
   }
   {
     \cs_set:Npn \__datatool_get_col_type_header:w ##1% stuff before
       \db@plist@elt@w% start of key block
        \db@col@id@w #2\db@col@id@end@% column index
         \db@key@id@w ##2\db@key@id@end@% key id
         \db@type@id@w ##3\db@type@id@end@% data type
         \db@header@id@w ##4\db@header@id@end@% header
        \db@col@id@w #2\db@col@id@end@% column index
       \db@plist@elt@end@% end of key block
       ##5% stuff afterwards
       \q_nil
       {
         \tl_set:Nn \l__datatool_item_key_tl { ##2 }
         \tl_if_empty:nTF { ##3 }
         {
           \int_set_eq:NN \l__datatool_item_type_int \c_datatool_unknown_int
         }
         {
           \int_set:Nn \l__datatool_item_type_int { ##3 }
         }
       }
     \__datatool_get_col_type_header:w #1
```

In case the column isn't defined:

```
       \db@plist@elt@w% start of key block
        \db@col@id@w #2\db@col@id@end@% column index
         \db@key@id@w \q_no_value \db@key@id@end@% key id
         \db@type@id@w \db@type@id@end@% data type
         \db@header@id@w \db@header@id@end@% header
        \db@col@id@w #2\db@col@id@end@% column index
```

```
      \db@plist@elt@end@% end of key block
    \q_nil
  }
}
\cs_generate_variant:Nn \__datatool_get_col_type_key:nn { vV, VV }
```

Only key required:

```
\cs_new:Nn \__datatool_get_col_key:nn
{
  \tl_if_empty:nTF { #1 }
  {
    \tl_set:Nn \l__datatool_item_key_tl { \q_no_value }
  }
  {
    \cs_set:Npn \__datatool_get_col_type_header:w ##1% stuff before
      \db@plist@elt@w% start of key block
       \db@col@id@w #2\db@col@id@end@% column index
        \db@key@id@w ##2\db@key@id@end@% key id
        \db@type@id@w ##3\db@type@id@end@% data type
        \db@header@id@w ##4\db@header@id@end@% header
       \db@col@id@w #2\db@col@id@end@% column index
      \db@plist@elt@end@% end of key block
      ##5% stuff afterwards
      \q_nil
      {
        \tl_set:Nn \l__datatool_item_key_tl { ##2 }
      }
    \__datatool_get_col_type_header:w #1
```

In case the column isn't defined:

```
      \db@plist@elt@w% start of key block
       \db@col@id@w #2\db@col@id@end@% column index
        \db@key@id@w \q_no_value \db@key@id@end@% key id
        \db@type@id@w \db@type@id@end@% data type
        \db@header@id@w \db@header@id@end@% header
       \db@col@id@w #2\db@col@id@end@% column index
      \db@plist@elt@end@% end of key block
    \q_nil
  }
}
\cs_generate_variant:Nn \__datatool_get_col_key:nn { vV, VV }
\ExplSyntaxOff
```

`\@dtl@before`

```
\newtoks\@dtl@before
```

`\@dtl@after`

```
\newtoks\@dtl@after
```

`\@dtl@colhead`

```
\newtoks\@dtl@colhead
```

```
\ExplSyntaxOn
```

Set dtlkeys@⟨*db-name*⟩ register according to the global setting. Syntax: {⟨*db-name*⟩} {⟨*content*⟩}

```
\cs_new:Nn \__datatool_dtlkeys_set:nn
{
  \bool_if:NTF \l__datatool_db_global_bool
   {
     \__datatool_token_register_gset:cn
       { dtlkeys@ #1 } { #2 }
   }
   {
     \__datatool_token_register_set:cn
       { dtlkeys@ #1 } { #2 }
   }
}
\cs_generate_variant:Nn \__datatool_dtlkeys_set:nn
 { nx, Vx, VV }
```

Append to dtlkeys@⟨*db-name*⟩ register according to the global setting. Syntax: {⟨*db-name*⟩}{⟨*content*⟩}

```
\cs_new:Nn \__datatool_dtlkeys_put_right:nn
{
  \bool_if:NTF \l__datatool_db_global_bool
   {
     \__datatool_token_register_gput_right:cn
       { dtlkeys@ #1 } { #2 }
   }
   {
     \__datatool_token_register_put_right:cn
       { dtlkeys@ #1 } { #2 }
   }
}
\cs_generate_variant:Nn \__datatool_dtlkeys_put_right:nn
 { nx, Vx }
```

Set dtldb@⟨*db-name*⟩ register according to the global setting. Syntax: {⟨*db-name*⟩} {⟨*content*⟩}

```
\cs_new:Nn \__datatool_dtldb_set:nn
{
  \bool_if:NTF \l__datatool_db_global_bool
   {
     \__datatool_token_register_gset:cn
       { dtldb@ #1 } { #2 }
   }
   {
     \__datatool_token_register_set:cn
       { dtldb@ #1 } { #2 }
   }
}
\cs_generate_variant:Nn \__datatool_dtldb_set:nn
```

```
  { nx, Vx, nV, VV }
```

Append to dtldb@⟨*db-name*⟩ register according to the global setting. Syntax: {⟨*db-name*⟩}{⟨*content*⟩}

```
\cs_new:Nn \__datatool_dtldb_put_right:nn
{
  \bool_if:NTF \l__datatool_db_global_bool
   {
     \__datatool_token_register_gput_right:cn
       { dtldb@ #1 } { #2 }
   }
   {
     \__datatool_token_register_put_right:cn
       { dtldb@ #1 } { #2 }
   }
}
\cs_generate_variant:Nn \__datatool_dtldb_put_right:nn
 { nx, Vx }
```

Set dtldb@⟨*db-name*⟩ register according to the global setting where concatenation is used with before and after registers. Syntax: {⟨*db-name*⟩}⟨*before reg*⟩{⟨*content*⟩}⟨*after reg*⟩

```
\cs_new:Nn \__datatool_dtldb_concat:nNnN
{
  \bool_if:NTF \l__datatool_db_global_bool
   {
     \__datatool_token_register_gconcat_middle:cNnN
       { dtldb@#1 } #2 { #3 } #4
   }
   {
     \__datatool_token_register_concat_middle:cNnN
       { dtldb@#1 } #2 { #3 } #4
   }
}
\cs_generate_variant:Nn \__datatool_dtldb_concat:nNnN
 { nNxN , VNxN }
```

Expands to numeric value of the given data type. Empty will expand to -1.

```
\cs_new:Nn \__datatool_use_datatype:n
{
  \tl_if_empty:nTF { #1 }
  { -1 }
  {
    \tl_if_single:nTF { #1 }
    {
      \tl_if_empty:NTF #1 { -1 } { \int_eval:n { #1 } }
    }
    { \int_eval:n { #1 } }
  }
}
```

\DTLaddcolumn

\DTLaddcolumn{⟨*db*⟩}{⟨*key*⟩}

Adds a column with given key to given column. No data is added to the column. The starred version doesn't check for the existence of the database.

```
\newrobustcmd*{\DTLaddcolumn}{%
   \@ifstar\@sDTLaddcolumn\@DTLaddcolumn
}
\newcommand{\@DTLaddcolumn}[2]{
  \DTLifdbexists { #1 }
  {
    \@sDTLifhaskey { #1 } { #2 }
    {
      \PackageError {datatool}
      {
        Can't ~ add ~ new ~ column ~ `#2' ~ to ~ database ~ `#1': ~
        column ~ with ~ that ~ key ~ already ~ exists
      }
      { }
    }
    {
      \s@DTLaddcolumn { #1 } { #2 }
    }
  }
  {
    \PackageError {datatool}
    {
      Can't ~ add ~ new ~ column ~ to ~ database ~ `#1': ~
      database ~ doesn't ~ exist
    }
    { }
  }
}
\newcommand{\s@DTLaddcolumn}[2]{%
  \__datatool_add_column_with_header:nxnn { #1 } { #2 } { -1 } { #2 }
}
```

\DTLaddcolumnwithheader

\DTLaddcolumnwithheader{⟨*db*⟩}{⟨*key*⟩}{⟨*header*⟩}

```
\NewDocumentCommand \DTLaddcolumnwithheader { s m m m }
{
  \IfBooleanTF { #1 }
  {
    \__datatool_add_column_with_header:nxnn { #2 } { #3 } { -1 } { #4 }
  }
  {
    \@DTLaddcolumnwithheader { #2 } { #3 } { #4 }
  }
```

```
}
```

\@DTLaddcolumnwithheader

```
\newcommand \@DTLaddcolumnwithheader [3]
{
  \DTLifdbexists { #1 }
  {
    \@sDTLifhaskey {#1} {#2}
    {
      \PackageError {datatool}
      {
        Can't ~ add ~ new ~ column ~ `#2' ~ to ~ database ~ `#1': ~
        column ~ with ~ that ~ key ~ already ~ exists
      }
      {}
    }
    {
      \__datatool_add_column_with_header:nxnn { #1 } { #2 } { -
1 } { #3 }
    }
  }
  {
    \PackageError {datatool}
    {
      Can't ~ add ~ new ~ column ~ to ~ database ~ `#1': ~
      database ~ doesn't ~ exist
    }
    {}
  }
}
```

Syntax: `{`⟨*db*⟩`}{`⟨*key*⟩`}{`⟨*type*⟩`}{`⟨*header*⟩`}`

```
\cs_new:Nn \__datatool_add_column_with_header:nnnn
{
  \bool_if:NTF \l__datatool_db_global_bool
  {
```

Globally increment column count.

```
    \int_gincr:c { dtlcols@#1 }
    \int_set_eq:Nc \dtlcolumnnum { dtlcols@#1 }
```

Set column index for this key.

```
    \csxdef{dtl@ci@#1@#2}{\number\dtlcolumnnum}%
```

Globally append to property list

```
    \__datatool_token_register_gput_right:cx
      { dtlkeys@#1 }
    {
      \__datatool_column_markup:Vnnn
         \dtlcolumnnum { #2 } { #3 } { #4 }
    }
```

```
   }
   {
```
Locally increment column count.
```
     \int_incr:c { dtlcols@#1 }
     \int_set_eq:Nc \dtlcolumnnum { dtlcols@#1 }
```
Set column index for this key.
```
     \csedef{dtl@ci@#1@#2}{\number\dtlcolumnnum}%
```
Locally append to property list
```
     \__datatool_token_register_put_right:cx
       { dtlkeys@#1 }
     {
       \__datatool_column_markup:Vnnn
          \dtlcolumnnum { #2 } { #3 } { #4 }
     }
   }
 }
 \cs_generate_variant:Nn \__datatool_add_column_with_header:nnnn
  { nxxn, nxnn, nxxx, VVVV, VVnV }
```

\@dtl@updatekeys

```
\@dtl@updatekeys{⟨db⟩}{⟨key⟩}{⟨value⟩}
```

Adds key to database's key list if it doesn't exist. The value is used to update the data type associated with that key. Key must be fully expanded. Doesn't check if database exists.

```
 \newcommand*{\@dtl@updatekeys}[3]{%
```
Check if key already exists
```
   \@sDTLifhaskey{#1}{#2}%
   {%
```
Key exists, may need to update data type. First get the column index.
```
     \int_set:Nn \dtlcolumnnum { \dtlcolumnindex { #1 } { #2 } }
```
Get the properties for this column
```
     \__datatool_get_keydata:NNNNNvV
       \@dtl@key \@dtl@type \@dtl@colhead \@dtl@before \@dtl@after
       { dtlkeys@#1 } \dtlcolumnnum
```
Is the value empty?
```
     \ifstrempty{#3}%
     {%
```
Leave data type as it is
```
     }%
     {%
```
Make a copy of current data type
```
       \let\@dtl@oldtype\@dtl@type
```

Check the data type for this entry (stored in \@dtl@datatype)

```
\@dtl@checknumerical{#3}%
\ifnum \@dtl@datatype = \c_datatool_unknown_int
\else
```

If this column currently has no data type assigned to it then use the new type.

```
  \ifdefempty{\@dtl@type}%
  {%
    \edef\@dtl@type{\number\@dtl@datatype}%
  }%
  {%
```

This column already has an associated data type but it may need updating.

```
    \ifcase\@dtl@datatype % string
```

String overrides all other types

```
      \def\@dtl@type{0}%
    \or % int
```

All other types override int, so leave it as it is

```
    \or % real
```

Real overrides int, but not currency or string

```
      \ifnum\@dtl@type=1\relax
        \def\@dtl@type{2}%
      \fi
    \or % currency
```

Currency overrides int and real but not string

```
      \ifnum\@dtl@type>0\relax
        \def\@dtl@type{3}%
      \fi
    \fi
  }%
\fi
```

Has the data type been updated?

```
\ifx\@dtl@oldtype\@dtl@type
```

No change needed

```
\else
```

Update required

```
  \tl_if_empty:NTF \@dtl@type
   {
     \int_set:Nn \@dtl@datatype { \c_datatool_unknown_int }
   }
   {
     \int_set:Nn \@dtl@datatype { \@dtl@type }
   }
  \bool_if:NTF \l__datatool_db_global_bool
  {
    \__datatool_token_register_gconcat_middle:cNxN
```

```
          { dtlkeys@#1 }
        \@dtl@before
        {
          \__datatool_column_markup:VnVV
            \dtlcolumnnum
            { #2 }
            \@dtl@datatype
            \@dtl@colhead
        }
        \@dtl@after
      }
      {
        \__datatool_token_register_concat_middle:cNxN
         { dtlkeys@#1 }
        \@dtl@before
        {
          \__datatool_column_markup:VnVV
            \dtlcolumnnum
            { #2 }
            \@dtl@datatype
            \@dtl@colhead
        }
        \@dtl@after
      }
    \fi
  }%
}%
{%
  \bool_if:NTF \l__datatool_db_global_bool
  {
```

Key doesn't exist. Increment column count.

```
      \int_gincr:c { dtlcols @ #1}
      \int_set_eq:Nc \dtlcolumnnum { dtlcols @ #1 }
```

Set column index for this key.

```
      \csxdef { dtl@ci@#1@#2 } { \int_use:N \dtlcolumnnum }
    }
    {
      \int_incr:c { dtlcols @ #1}
      \int_set_eq:Nc \dtlcolumnnum { dtlcols @ #1 }
      \csedef { dtl@ci@#1@#2 } { \int_use:N \dtlcolumnnum }
    }
```

Get data type for this entry (stored in `\@dtl@datatype`)

```
    \tl_if_empty:nTF { #3 }
    {
      \tl_clear:N \@dtl@type % don't know data type yet
      \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int
    }
    {
```

```
      \@dtl@checknumerical { #3 }
      \int_compare:nNnTF
        { \@dtl@datatype } = { \c_datatool_unknown_int }
      {
        \tl_clear:N \@dtl@type
      }
      {
        \tl_set:NV \@dtl@type \@dtl@datatype
      }
    }
```

Append to property list

```
    \bool_if:NTF \l__datatool_db_global_bool
    {
      \__datatool_token_register_gput_right:cx
        { dtlkeys @ #1 }
      {
        \__datatool_column_markup:VnVn
          \dtlcolumnnum
          { #2 }
          \@dtl@datatype
          { #2 }
      }
    }
    {
      \__datatool_token_register_put_right:cx
       { dtlkeys @ #1 }
      {
        \__datatool_column_markup:VnVn
          \dtlcolumnnum
          { #2 }
          \@dtl@datatype
          { #2 }
      }
    }
  }
}
```

Syntax: `{`⟨*db*⟩`}{`⟨*key*⟩`}{`⟨*type*⟩`}`

Update column meta data identified by column key where the type has already been found. If no column exists, a new one will be created.

```
\cs_new:Nn \__datatool_update_meta_data_with_type:nnn
{
```

Check if key already exists

```
  \@sDTLifhaskey { #1 } { #2 }
  {
```

Key exists, may need to update data type. First get the column index.

```
    \int_set:Nn \dtlcolumnnum
      { \dtlcolumnindex { #1 } { #2 } }
```

Get the properties for this column

```
\__datatool_get_props:NNNNNvV
   \l__datatool_item_key_tl
   \l__datatool_item_type_tl
   \l__datatool_item_head_tl
   \l__datatool_before_tl
   \l__datatool_after_tl
   { dtlkeys@ #1 } \dtlcolumnnum
```

Treat empty type as unknown.

```
\tl_if_empty:NT \l__datatool_item_type_tl
 {
   \tl_set:Nn \l__datatool_item_type_tl
     { \c_datatool_unknown_int }
 }
```

Current column type:

```
\int_set:Nn \l__datatool_tmp_datatype_int
  { \l__datatool_item_type_tl }
```

New column type:

```
\int_set:Nn \@dtl@datatype { #3 }
```

Check if type needs updating:

```
\__datatool_update_datatype:
```

If type has changed, update required.

```
\int_compare:nNnF { \@dtl@datatype } = { \l__datatool_tmp_datatype_int }
 {
```

Update required

```
  \__datatool_dtlkeys_set:nx { #1 }
   {
     \exp_not:V \l__datatool_before_tl
     \__datatool_column_markup:VnVV
       \dtlcolumnnum
       { #2 }
       \@dtl@datatype
       \l__datatool_item_head_tl
     \exp_not:V \l__datatool_after_tl
   }
 }
}
{
 \bool_if:NTF \l__datatool_db_global_bool
 {
```

Key doesn't exist. Increment column count.

```
   \int_gincr:c { dtlcols @ #1}
   \int_set_eq:Nc \dtlcolumnnum { dtlcols @ #1 }
```

Set column index for this key.

```
      \csxdef { dtl@ci@#1@#2 } { \int_use:N \dtlcolumnnum }
    }
    {
      \int_incr:c { dtlcols @ #1}
      \int_set_eq:Nc \dtlcolumnnum { dtlcols @ #1 }
      \csedef { dtl@ci@#1@#2 } { \int_use:N \dtlcolumnnum }
    }
```

Append to property list

```
    \bool_if:NTF \l__datatool_db_global_bool
    {
      \__datatool_token_register_gput_right:cx
        { dtlkeys @ #1 }
      {
        \__datatool_column_markup:Vnnn
          \dtlcolumnnum
          { #2 } { #3 } { #2 }
      }
    }
    {
      \__datatool_token_register_put_right:cx
       { dtlkeys @ #1 }
      {
        \__datatool_column_markup:Vnnn
          \dtlcolumnnum
          { #2 } { #3 } { #2 }
      }
    }
  }
}
\cs_generate_variant:Nn
 \__datatool_update_meta_data_with_type:nnn
 { nnV, VVV }
```

Syntax: `{`⟨*db*⟩`}{`⟨*col idx*⟩`}{`⟨*type*⟩`}`
Similar but the column is referenced by its index.

```
\cs_new:Nn \__datatool_update_meta_data_col_index_with_type:nnn
{
```

Does the column already exist?

```
  \int_compare:nNnTF
    { #2 } > { \DTLcolumncount { #1 } }
   {
```

New column but the index can only be one more than the total column count.

```
     \int_compare:nNnTF
       { #2 } = { \DTLcolumncount { #1 } + \c_one_int }
      {
```

Create default column key and add new column.

```
        \tl_set:Nn \l__datatool_item_key_tl { \dtldefaultkey #2 }
        \int_gincr:c { dtlcols @ #1}
```
Append to property list
```
        \bool_if:NTF \l__datatool_db_global_bool
        {
          \__datatool_token_register_gput_right:cx
            { dtlkeys @ #1 }
          {
            \__datatool_column_markup:nVnV
              { #2 }
              \l__datatool_item_key_tl
              { #3 }
              \l__datatool_item_key_tl
          }
        }
        {
          \__datatool_token_register_put_right:cx
           { dtlkeys @ #1 }
          {
            \__datatool_column_markup:nVnV
              { #2 }
              \l__datatool_item_key_tl
              { #3 }
              \l__datatool_item_key_tl
          }
        }
      }
      {
        \PackageError { datatool }
         {
           Invalid ~ column ~ index ~ #2 . ~ A ~ new ~ column ~
           requires ~ the ~ index ~ to ~ be ~ one ~ more ~ than ~
           the ~ current ~ column ~ count
         }
         {
           The ~ database ~ ` #1 ' ~ only ~ has ~
           \DTLcolumncount { #1 } ~ column(s). ~ The ~ index ~
           #2 ~ is ~ too ~ large
         }
      }
    }
    {
```
Check index not invalid.
```
      \int_compare:nNnTF { #2 } < { \c_one_int }
       {
         \PackageError { datatool }
          {
            Invalid ~ column ~ index ~ #2 . ~ The ~ column ~
            index ~ must ~ start ~ at ~ 1
```

```
      }
      {
        The ~ supplied ~ column ~ index ~ to ~ update ~
        column ~ meta ~ data ~ must ~ be ~ from ~ 1 ~
        to ~ one ~ more ~ than ~ the ~ total ~ number ~
        of ~ columns ~ (\DTLcolumncount { #1 }+1 ~ for ~
        database ~ `#1' )
      }
    }
    {
```

Existing column. Get the properties.

```
      \__datatool_get_props:NNNNNvn
        \l__datatool_item_key_tl
        \l__datatool_item_type_tl
        \l__datatool_item_head_tl
        \l__datatool_before_tl
        \l__datatool_after_tl
        { dtlkeys@ #1 } { #2 }
```

Treat empty type as unknown.

```
      \tl_if_empty:NT
        \l__datatool_item_type_tl
       {
        \tl_set:Nn
          \l__datatool_item_type_tl
          { \c_datatool_unknown_int }
       }
```

If the original type is a string, don't update.

```
      \int_compare:nNnF
        { \l__datatool_item_type_tl }
         =
        { \c_datatool_string_int }
       {
```

If the new type is greater than the old, then it needs updating.

```
        \int_compare:nNnT
            { #3 } > { \l__datatool_item_type_tl }
         {
```

Update required

```
           \bool_if:NTF \l__datatool_db_global_bool
           {
             \__datatool_token_register_gconcat_middle:cNxN
              { dtlkeys@ #1 }
             \l__datatool_before_tl
              {
               \__datatool_column_markup:nVnV
                  { #2 }
                 \l__datatool_item_key_tl
                  { #3 }
```

```
                        \l__datatool_item_head_tl
                      }
                    \l__datatool_after_tl
                  }
                  {
                    \__datatool_token_register_concat_middle:cNxN
                     { dtlkeys@#1 }
                    \l__datatool_before_tl
                     {
                      \__datatool_column_markup:nVnV
                         { #2 }
                        \l__datatool_item_key_tl
                         { #3 }
                        \l__datatool_item_head_tl
                     }
                    \l__datatool_after_tl
                  }
               }
            }
         }
      }
   }
\cs_generate_variant:Nn
\__datatool_update_meta_data_col_index_with_type:nnn
 { nnV, VVV, VnV }
```

\DTLsetheader

\DTLsetheader{⟨*db*⟩}{⟨*key*⟩}{⟨*header*⟩}

Sets header for column given by ⟨*key*⟩ in database ⟨*db*⟩. Starred version doesn't check for existence of database or key.

```
\newrobustcmd*{\DTLsetheader}{\@ifstar\@sDTLsetheader\@DTLsetheader}
```

\@DTLsetheader Unstarred version

```
\newcommand*{\@DTLsetheader}[3]{%
```

Check if database exists

```
  \DTLifdbexists{#1}%
  {%
```

Check if key exists.

```
    \@sDTLifhaskey{#1}{#2}%
    {%
      \@sDTLsetheader{#1}{#2}{#3}%
    }%
    {%
      \PackageError{datatool}{Database ~ `#1' ~ doesn't ~ contain ~ key ~
      `#2'}{}%
    }%
  }%
```

```
    {%
      \PackageError{datatool}{Database ~ `#1' ~ doesn't ~ exist}{}%
    }%
  }
```

`\@sDTLsetheader` Starred version

```
\newcommand*{\@sDTLsetheader}[3]{%
  \expandafter\dtlcolumnnum\expandafter
    =\dtlcolumnindex{#1}{#2}\relax
  \@dtl@setheaderforindex{#1}{\dtlcolumnnum}{#3}%
}
```

`\@dtl@setheaderforindex`

`\@dtl@setheaderforindex{`⟨*db*⟩`}{`⟨*column index*⟩`}{`⟨*header*⟩`}`

Sets the header for column given by ⟨*column index*⟩ in database ⟨*db*⟩. The header must be expanded.

```
\newcommand*{\@dtl@setheaderforindex}[3]{%
```

Get the properties for this column

```
  \__datatool_get_keydata:NNNNNve
    \@dtl@key \@dtl@type \@dtl@colhead \@dtl@before \@dtl@after
    { dtlkeys@#1 } { \int_eval:n { #2 } }
```

Ensure data type variable is numeric.

```
  \tl_if_empty:NTF \@dtl@type
   {
     \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int
   }
   {
     \int_set:Nn \@dtl@datatype { \@dtl@type }
   }
```

Store the header in `\@dtl@toks`

```
  \@dtl@colhead={#3}%
```

Reconstruct property list

```
  \tl_set:Nx \@dtl@colnum { \int_eval:n { #2 } }
  \bool_if:NTF \l__datatool_db_global_bool
  {
    \__datatool_token_register_gconcat_middle:cNxN
     { dtlkeys@#1 }
     \@dtl@before
    {
      \__datatool_column_markup:VVVV
        \@dtl@colnum
        \@dtl@key
        \@dtl@datatype
        \@dtl@colhead
```

```
    }
    \@dtl@after
  }
  {
    \__datatool_token_register_concat_middle:cNxN
     { dtlkeys@#1 }
    \@dtl@before
    {
      \__datatool_column_markup:VVVV
        \@dtl@colnum
        \@dtl@key
        \@dtl@datatype
        \@dtl@colhead
    }
    \@dtl@after
  }
}
```

Process value to be added to a database according to the above settings. This will set both `\@dtl@toks` and `\l__datatool_item_value_tl`

```
\cs_new:Nn \__datatool_process_new_value:n
{
  \tl_set:Nn \l__datatool_item_value_tl { #1 }
  \datatool_if_null:NTF
    \l__datatool_item_value_tl
  {
    \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int
  }
  {
```

Special case if value starts with `\dtlspecialvalue`:

```
    \tl_if_head_eq_meaning:nNTF
      { #1 } \dtlspecialvalue
    {
      \group_begin:
```

Expand and parse to get the data type.

```
        \exp_args:Nx \__datatool_parse_datum:n { #1 }
       \exp_args:NNNV
      \group_end:
       \int_set:Nn \@dtl@datatype \@dtl@datatype
```

Store as is without following the expansion, trim or store datum settings.

```
      \@dtl@toks { #1 }
      \tl_set:NV \l__datatool_item_value_tl \@dtl@toks
    }
    {
```

This sets the `\@dtl@toks` token register for backward-compatibility. It also follows the trim and expand new value settings.

```
      \@dtl@setnewvalue { #1 }
```

Update `\l__datatool_item_value_tl` (which won't be set if trim not on.)

```
\tl_set:NV \l__datatool_item_value_tl \@dtl@toks
\bool_if:NTF \l__datatool_db_store_datum_bool
 {
   \__datatool_parse:N \l__datatool_item_value_tl
 }
 {
```

Just parse to get the data type.

```
   \exp_args:NV \__datatool_parse_datum:n
     \l__datatool_item_value_tl
 }
```

In case the value was provided in datum format.

```
\__datatool_to_weird_datum_no_parse:N
  \l__datatool_item_value_tl
```

Update `\@dtl@toks`:

```
\datatool_if_null:NF
  \l__datatool_item_value_tl
 {
   \__datatool_token_register_set:NV
     \@dtl@toks
     \l__datatool_item_value_tl
 }
    }
  }
}
\cs_generate_variant:Nn
  \__datatool_process_new_value:n
  { V }
```

`\@dtl@storeandupdate`

```
\newcommand*{\@dtl@storeandupdate}[3]{%
```

Store the value of this entry in `\@dtl@toks` taking the expansion setting into account.

```
  \__datatool_process_new_value:n { #3 }
```

This will have parsed the value and set `\@dtl@datatype` so it can be passed here:

```
  \__datatool_update_meta_data_with_type:nnV
    { #1 } { #2 } \@dtl@datatype
}
```

`\DTLnewdbentry`

`\DTLnewdbentry{`⟨*db name*⟩`}{`⟨*id*⟩`}{`⟨*value*⟩`}.`

Adds an entry to the last row (adds new row if database is empty) and updates general column information if necessary. The starred version doesn't check if the database exists.

```
\NewDocumentCommand \DTLnewdbentry { s }
{
  \int_zero:N \dtlcolumnnum
  \IfBooleanTF { #1 } { \@sDTLnewdbentry } { \@DTLnewdbentry }
}
```

`\@DTLnewdbentry` Unstarred version of `\DTLnewdbentry`.

```
\newcommand{\@DTLnewdbentry}[3]{%
  \DTLifdbexists { #1 }
  {
    \exp_args:Nv \int_if_zero:nTF { dtlrows@#1 }
    {
      \PackageError { datatool }
      {
        Can't ~ add ~ an ~ entry ~ to ~ the ~ last ~ row ~ of ~ `#1': ~
        database ~ has ~ no ~ rows
      }
      {
        You ~ need ~ to ~ create ~ a ~ new ~ row ~ before ~ you ~
        can ~ add ~ an ~ entry ~ to ~ it ~ with ~
       \token_to_str:N \DTLnewrow {#1} ~ or ~
       \token_to_str:N \DTLaction [name={#1}] { new ~ row }
      }
    }
    {
      \@sDTLnewdbentry { #1 } { #2 } { #3 }
    }
  }
  {
    \PackageError { datatool }
     {
       Can't ~ add ~ new ~ entry ~ to ~ database ~ `#1': ~
       database ~ doesn't ~ exist
     }
     {
       You ~ need ~ to ~ define ~ the ~ database ~ first ~ with ~
       \token_to_str:N \DTLnewdb {#1} ~ or ~
       \token_to_str:N \DTLaction [name={#1}] { new }
     }
  }
}
```

`\@sDTLnewdbentry` Starred version of `\DTLnewdbentry` (doesn't check if the database exists).

```
\newcommand*{\@sDTLnewdbentry}[3]{%
```

Store the value of this entry in `\@dtl@toks` and update key list.

```
  \@dtl@storeandupdate { #1 } { #2 } { #3 }
```

Get the column index

```
  \int_set:Nn \dtlcolumnnum
    { \dtlcolumnindex { #1 } { #2 } }
```

Now split off the last row and insert the new entry:

```
   \tl_set:Nn \l__datatool_item_key_tl { #2 }
   \@s@DTLnewdbentry { #1 }
 }
```

`\@s@DTLnewdbentry` Argument is the database name. The value is expected to be in `\@dtl@toks`, having been pre-processed by `\__datatool_process_new_value:n`

```
 \newcommand*{\@s@DTLnewdbentry}[1]{%
```

Get the current row:

```
   \dtlgetrow { #1 } { \int_use:c { dtlrows@#1} }
```

Check if this row already has an entry for the given column.

```
   \exp_args:NNV
     \dtlgetentryfromcurrentrow
     \dtl@entry
     \dtlcolumnnum
   \datatool_if_null:NTF \dtl@entry
   {
```

There are no entries in this row for the given column. Add this entry.

```
     \__datatool_token_register_put_right:Nx
      \dtlcurrentrow
       {
         \__datatool_row_element_markup:VV
           \dtlcolumnnum
           \@dtl@toks
       }
     \__datatool_dtldb_concat:nNxN
        { #1 }
       \dtlbeforerow
        {
          \__datatool_row_markup:vV
            { dtlrows@#1 }
           \dtlcurrentrow
        }
       \dtlafterrow
```

Print information message if in verbose mode.

```
     \dtl@message
     {
      Added ~ \l__datatool_item_key_tl \c_space_tl ~
       -> ~ \the\@dtl@toks \c_space_tl ~ to ~ database ~ `#1'
     }
   }
   {
```

There’s already an entry for the given column in this row

```
     \PackageError { datatool }
     {
       Can't ~ add ~ entry ~ with ~ ID ~
```

```
        `\l__datatool_item_key_tl' ~ to ~
        current ~ row ~ of ~ database ~ `#1' ~ in ~ column ~
        \int_use:N \dtlcolumnnum
      }
      {
        There ~ is ~ already ~ an ~ entry ~ in ~ this ~
        column ~ in ~ the ~ current ~ row
      }
    }
  }
```

\DTLifdbexists

```
\DTLifdbexists{⟨db name⟩}{⟨true part⟩}{⟨false part⟩}
```

Checks if a data base with the given name exists.

```
\newcommand{\DTLifdbexists}[3]{
  \ifcsdef { dtldb@#1 } { #2 } { #3 }
}
```

These commands are provided for DTLTEX v3.0 format.

\DTLdbNewRow

```
\NewDocumentCommand \DTLdbNewRow {  }
 {
   \@sDTLnewrow { \l__datatool_default_dbname_tl }
 }
```

\DTLdbNewEntry

```
\NewDocumentCommand \DTLdbNewEntry { m m }
 {
  \@sDTLnewdbentry { \l__datatool_default_dbname_tl } { #1 } { #2 }
 }
```

\DTLdbSetHeader

```
\NewDocumentCommand \DTLdbSetHeader { m m }
 {
  \@sDTLsetheader { \l__datatool_default_dbname_tl } { #1 } { #2 }
 }
```

12.4 Accessing Data

\DTLassign

```
\DTLassign{⟨db⟩}{⟨row idx⟩}{⟨assign list⟩}
```

Assigns values given in ⟨*assign list*⟩ for row ⟨*row idx*⟩ in database ⟨*db*⟩. (Where ⟨*assign list*⟩ is in the same form as in `\DTLforeach`.) This command doesn't check the 'global' option.

```
\NewDocumentCommand \DTLassign { m m m }
{
  \DTLifdbexists { #1 }
  {
```

Grouped in the event that `\dtlcurrentrow` is already in use. (Assignments in `\@dtl@assign` are global.)

```
    {
      \dtlgetrow { #1 } { #2 }
      \@dtl@assign { #3 } { #1 }
    }
  }
  {
    \PackageError {datatool}
    {
      \token_to_str:N \DTLassign : ~
       Database ~ `#1' ~ doesn't ~ exist
    }
    {
      Check ~ that ~ you ~ have ~ spelt ~ the ~
      database ~ name ~ correctly
    }
  }
}
```

`\DTLassignfirstmatch`

`\DTLassignfirstmatch{⟨db⟩}{⟨col key⟩}{⟨value⟩}{⟨assign list⟩}`

Applies the assignment list to the first row that has the given value in the given column. (Value must be expanded.) This command doesn't check the 'global' option.

```
\NewDocumentCommand \DTLassignfirstmatch { m m m m }
{
  \dtl@assignfirstmatch { #3 } { #1 } { #2 } { #4 }
}
```

`\xDTLassignfirstmatch`

`\xDTLassignfirstmatch{⟨db⟩}{⟨col key⟩}{⟨value⟩}{⟨assign list⟩}`

Applies the assignment list to the first row that has the given value in the given column. (Performs *one level* expansion on ⟨*value*⟩.)

```
\NewDocumentCommand \xDTLassignfirstmatch { m m m m }
{
  \exp_args:Nx
   \dtl@assignfirstmatch
     { \expandonce { #3 } }
     { #1 } { #2 } { #4 }
}
```

\dtl@assignfirstmatch

```
\dtl@assignfirstmatch{⟨value⟩}{⟨db⟩}{⟨col key⟩}
{⟨assign list⟩}
```

Internal swaps the ordering around so the value is first. (This just makes it easier for `\xDTLassignfirstmatch`.)

```
\newcommand*{\dtl@assignfirstmatch}[4]{
  \DTLifdbexists { #2 }
  {%
```

Grouped in the event that `\dtlcurrentrow` is already in use. (Assignments in `\@dtl@assign` are global.)

```
    {
```

Get row idx:

```
      \dtlgetrowindex \dtl@asg@rowidx
        { #2 }
        { \dtlcolumnindex { #2 } { #3 } }
        { #1 }
      \datatool_if_null:NTF \dtl@asg@rowidx
      {
        \PackageError {datatool}
        {
          No ~ match ~ found ~ for ~
          \token_to_str:N \DTLassignfirstmatch
            \tl_to_str:n { { #2 } { #3 } { #1 } { #4 } }
        }
        {
          The ~ database ~ `#2' ~ doesn't ~ contain ~
          an ~ entry ~ that ~ exactly ~ matches ~
          `\tl_to_str:n { #1 } ' ~ for ~ column ~
          `#3'
        }
      }
      {
        \dtlgetrow { #2 } \dtl@asg@rowidx
        \@dtl@assign { #4 } { #2 }
      }
    }
  }
  {
    \PackageError {datatool}
    {
      Can't ~ assign ~ first ~ match : ~ database ~ `#2' ~
      doesn't ~ exist
    }
    {
      Check ~ that ~ you ~ have ~ correctly ~ spelt ~
      the ~ database ~ name
    }
```

```
  }
}
```

\DTLassignfromcurrentrow

\DTLassignfromcurrentrow{⟨*assign list*⟩}

For use where the current row has already been selected. The placeholder `\dtldbname` should already have been set. Unlike the above, this performs a local assignment.

```
\NewDocumentCommand \DTLassignfromcurrentrow { m }
{
  \tl_if_empty:NTF \dtldbname
   {
     \PackageError { datatool }
     {
       Can't ~ assign ~ from ~ current ~ row: ~
       current ~ row ~ not ~ assigned ~ or ~ placeholders ~
       have ~ been ~ changed
     }
     {
       You ~ need ~ to ~ select ~ the ~ current ~ row ~
       (for ~ example ~ with ~ \token_to_str:N \dtlgetrow ) ~
       before ~ you ~ can ~ query ~ the ~ current ~ row
     }
   }
   {
     \DTLifdbexists { \dtldbname }
     {
       \__datatool_assign_cskey_list:nnn { #1 }
         { \dtldbname } { \c_false_bool }
     }
     {
       \PackageError { datatool }
       {
         Can't ~ assign ~ from ~ current ~ row: ~
         database ~ `\dtldbname' ~ doesn't ~ exist
       }
       {
         It ~ seems ~ that ~ a ~ placeholder ~ normally ~ assigned ~
         when ~ a ~ row ~ is ~ selected ~ as ~ the ~ current ~ row ~
         doesn't ~ match ~ a ~ defined ~ database
       }
     }
   }
}
```

\@dtl@assign

\@dtl@assign{⟨*list*⟩}{⟨*db*⟩}

Assigns commands according to the given keys. The current row must be stored in `\dtlcurrentrow`. Version 3.0: definition rewritten in LaTeX3. The command name and syntax remains the same.

```
\newcommand*{\@dtl@assign}[2]{%
  \__datatool_assign_cskey_list:nnn { #1 } { #2 } { \c_true_bool }
}
```

New implementation:

```
\cs_new:Nn \__datatool_assign_cskey_list:nnn
{
  \keyval_parse:nnn
    { \__datatool_cskey_missing_val:n }
    { \__datatool_assign_parsed_cskey:nnnn { #2 } { #3 } }
    { #1 }
}
\cs_new:Nn \__datatool_assign_parsed_cskey:nnnn
{
  \__datatool_assign_cskey:Nnnn { #3 } { #4 } { #1 } { #2 }
}
```

`\__datatool_assign_cskey:Nnn` ⟨*tl var*⟩{⟨*key*⟩}{⟨*db name*⟩}

Assign command to the value of the item in the current row at the column identified by key. Local assignment:

```
\cs_new:Nn \__datatool_assign_cskey:Nnn
{
  \__datatool_assign_cskey:Nnnn { #1 } { #2 } { #3 } { \c_false_bool }
}
```

Global assignment:

```
\cs_new:Nn \__datatool_gassign_cskey:Nnn
{
  \__datatool_assign_cskey:Nnnn { #1 } { #2 } { #3 } { \c_false_bool }
}
```

`\__datatool_assign_cskey:Nnnn` ⟨*tl var*⟩{⟨*key*⟩}{⟨*db name*⟩}{⟨*bool*⟩}

If ⟨*bool*⟩ true assign globally, otherwise local assignment.

```
\cs_new:Nn \__datatool_assign_cskey:Nnnn
{
  \tl_if_single:nTF { #1 }
  {
```

Replicate old behaviour of `\@dtl@assigncmd` which set `\@dtl@dbname` to the database name:

```
    \tl_gset:Nx \@dtl@dbname { #3 }
```

Check if the given key exists:

```
\@sDTLifhaskey { #3 } { #2 }
{
  \exp_args:NNx \dtlgetentryfromcurrentrow
    #1
    { \dtlcolumnindex { #3 } { #2 } }
  \datatool_if_null:NT #1
    {
      \__datatool_set_null:Nnn { #1 } { #2 } { \@dtl@dbname }
    }
  \bool_if:nT { #4 }
  {
    \global \let #1 #1
  }
}
{
  \PackageError { datatool }
  {
     Can't ~ assign ~ `\tl_to_str:n { #1 = #2 }': ~ there ~
     is ~ no ~ key ~ `#2' ~ in ~ database ~ `#3'
  }
  {
    Check ~ that ~ you ~ have ~ correctly ~ spelt ~ the ~
    column ~ key
  }
  \bool_if:nTF { #4 }
  {
    \global \let #1 \DTLstringnull
  }
  {
    \global \let #1 \DTLstringnull
  }
}
}
{
  \PackageError { datatool}
  {
     Invalid ~ cs=key ~ assignment ~ syntax ~ in ~
     `\tl_to_str:n { #1 = #2 }': ~
     single ~ control ~ sequence ~ required ~ before ~ `='
  }
  {
    Check ~ that ~ you ~ have ~ the ~ assignment ~ the ~ correct ~
    way ~ round
  }
}
}
```

\@dtl@assigncmd

```
\@dtl@assigncmd⟨cmd⟩=⟨id⟩\@nil
```

Version 3.0: removed.

\@dtl@assigncmdnoop Version 3.0: removed.

\@dtl@setnull `\@dtl@setnull{`⟨*cmd*⟩`}{`⟨*id*⟩`}` globally sets ⟨*cmd*⟩ to either `\@dtlstringnull` or `\@dtlnumbernull` depending on the data type for ⟨*id*⟩. (Database name should be stored in `\@dtl@dbname` prior to use.)

```
\newcommand*{\@dtl@setnull}[2]{
```

Check if database given by `\@dtl@dbname` has the required key.

```
  \@sDTLifhaskey { \@dtl@dbname } { #2 }
  {
```

Set to the null applicable to the column data type.

```
    \@@dtl@setnull #1 {#2}
  }
  {
```

Key not defined in database `\@dtl@dbname`.

```
    \tl_gset_eq:NN #1 \DTLstringnull
  }
}
```

\@@dtl@setnull As above, but doesn't check if key exists

```
\newcommand*{\@@dtl@setnull}[2]{%
 \__datatool_gset_null:Nnn #1 { #2 } { \@dtl@dbname }
}
```

New to v3.0:
Global assignment:

```
\cs_new:Nn \__datatool_gset_null:Nnn
{
```

Get the data type associated with this key and store in `\@dtl@type`.

```
  \@sdtlgetdatatype \@dtl@type { #3 } { #2 }
```

Check data type.

```
  \bool_lazy_or:nnTF
    { \tl_if_empty_p:N \@dtl@type }
    { \int_compare_p:n { \@dtl@type < 1 } }
  {
```

Data type is unknown or 0, so set to string null.

```
    \tl_gset_eq:NN #1 \DTLstringnull
  }
  {
```

Data type is numerical, so set to number null.

```
      \tl_gset_eq:NN #1 \DTLnumbernull
    }
 }
```

Local assignment:

```
 \cs_new:Nn \__datatool_set_null:Nnn
 {
```

Get the data type associated with this key and store in `\@dtl@type`.

```
   \@sdtlgetdatatype \@dtl@type { #3 } { #2 }
```

Check data type.

```
    \bool_lazy_or:nnTF
      { \tl_if_empty_p:N \@dtl@type }
      { \int_compare_p:n { \@dtl@type < 1 } }
    {
```

Data type is unknown or 0, so set to string null.

```
      \tl_set_eq:NN #1 \DTLstringnull
    }
    {
```

Data type is numerical, so set to number null.

```
      \tl_set_eq:NN #1 \DTLnumbernull
    }
 }
```

`\DTLifnull`

`\DTLifnull{`⟨*command*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Checks if ⟨*command*⟩ is null (either `\DTLstringnull` or `\DTLnumbernull`) if true, does ⟨*true part*⟩ otherwise does ⟨*false part*⟩.

```
 \newcommand*{\DTLifnull}[3]{%
   \datatool_if_null:nTF { #1 } { #2 } { #3 }
 }
```

`\DTLifnulloremptyempty`

`\DTLifnulloremptyempty{`⟨*command*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

```
 \newcommand*{\DTLifnulloremptyempty}[3]{%
   \datatool_if_null_or_empty:nTF { #1 } { #2 } { #3 }
 }
```

`\DTLgetkeydata`

`\DTLgetkeydata{`⟨*key*⟩`}{`⟨*db*⟩`}{`⟨*col cs*⟩`}{`⟨*type cs*⟩`}{`⟨*header cs*⟩`}`

Gets data for given key in database ⟨*db*⟩: the column index is stored in ⟨*col cs*⟩ and data type is stored in ⟨*type cs*⟩. The unstarred version checks for the existence of the database and key, the starred version doesn't.

```
\newrobustcmd*{\DTLgetkeydata}{%
  \@ifstar\@sdtlgetkeydata\@dtlgetkeydata
}
```

`\@dtlgetkeydata` Unstarred version of `\DTLgetkeydata`

```
\newcommand*{\@dtlgetkeydata}[5]{
```

Check if the database exists.

```
  \DTLifdbexists { #2 }
  {
```

Check if the given key exists in the database.

```
    \@sDTLifhaskey { #2 } { #1 }
    {
```

Get the data.

```
      \@sdtlgetkeydata { #1 } { #2 } #3 #4 #5
    }
    {
```

Key not defined in the given database.

```
      \PackageError {datatool}
      {
        Can't ~ get ~ data ~ for ~ key ~ `#1' :
        key ~ `#1' ~ not ~ defined ~ in ~ database ~ `#2'
      }
      {
        Check ~ that ~ you ~ have ~ correctly ~ spelt ~
        the ~ column ~ key ~ and ~ the ~ database ~ name
      }
    }
  }
  {
```

Database not defined.

```
    \PackageError {datatool}
    {
      Can't ~ get ~ data ~ for ~ key ~ `#1' :
      database ~ `#2' ~ doesn't ~ exist
    }
    {
      Check ~ that ~ you ~ have ~ correctly ~ spelt ~
      the ~ database ~ name
    }
  }
}
```

\@sdtlgetkeydata `\@sdtlgetkeydata{⟨key⟩}{⟨db⟩}{⟨col cs⟩}{⟨type cs⟩}{⟨header cs⟩}` Starred verison of `\DTLgetkeydata`.

```
\newcommand*{\@sdtlgetkeydata}[5]{
  \@sdtl@getcolumnindex #3 { #2 } { #1 }
  \__datatool_get_props:NNNNNvx
    \@dtl@key #4 #5
    \l__datatool_before_tl
    \l__datatool_after_tl
    { dtlkeys@#2 }
    { #3 }
```

Backward-compatibility. TODO is this needed?

```
    \__datatool_token_register_set:NV
      \@dtl@colhead
      \l__datatool_item_head_tl
    \__datatool_token_register_set:NV
      \@dtl@before
      \l__datatool_before_tl
    \__datatool_token_register_set:NV
      \@dtl@after
      \l__datatool_after_tl
}
```

The gather value commands are used in databib.

\dtl@gathervalues `\dtl@gathervalues[⟨label⟩]{⟨db name⟩}{⟨row toks⟩}`

Stores each element of ⟨*row*⟩ in ⟨*db name*⟩ into the command `\@dtl@`⟨*label*⟩`@`⟨*key*⟩, where ⟨*key*⟩ is the key for that element, and ⟨*label*⟩ defaults to `key`.

```
\newcommand{\dtl@gathervalues}[3][key]{
  \dtlforeachkey
    ( \@dtl@key , \@dtl@col , \@dtl@type , \@dtl@head )
    \in { #2 }
   \do
   {

     \dtlgetentryfromrow \@dtl@tmp { \@dtl@col } #3
     \datatool_if_null:NT \@dtl@tmp
      {
        \@dtl@setnull \@dtl@tmp \@dtl@key
      }
      \tl_set_eq:cN { @dtl@#1@ \@dtl@key } \@dtl@tmp
   }
}
```

\dtl@g@gathervalues `\dtl@g@gathervalues[⟨label⟩]{⟨db name⟩}{⟨row toks⟩}`

As above but makes global assignments

```
\newcommand{\dtl@g@gathervalues}[3][key]{%
  \dtlforeachkey
    ( \@dtl@key , \@dtl@col , \@dtl@type , \@dtl@head )
    \in { #2 }
   \do
   {
     \dtlgetentryfromrow \@dtl@tmp { \@dtl@col } #3
     \datatool_if_null:NT \@dtl@tmp
      {
        \@dtl@setnull \@dtl@tmp \@dtl@key
      }
      \tl_gset_eq:cN { @dtl@#1@ \@dtl@key } \@dtl@tmp
   }
}
```

`\dtlcurrentrow` Define token register to store current row.

```
\newtoks\dtlcurrentrow
```

`\dtlbeforerow` Define token register to store everything before the current row.

```
\newtoks\dtlbeforerow
```

`\dtlafterrow` Define token register to store everything after the current row.

```
\newtoks\dtlafterrow
```

`\dtlgetrow`

```
\dtlgetrow{⟨db⟩}{⟨row idx⟩}
```

Gets row with index ⟨*row idx*⟩ from database named ⟨*db*⟩ and stores the row in `\dtlcurrentrow`, the preceding rows in `\dtlbeforerow` and the following rows in `\dtlafterrow`. The row index, ⟨*row idx*⟩, is stored in `\dtlrownum` and the database name, ⟨*db*⟩, is stored in `\dtldbname`. This assumes that the given row exists.

```
\NewDocumentCommand \dtlgetrow { m m }
{
  \datatool_get_row:nnTF { #1 } { #2 } { } { }
}

\cs_new:Nn \datatool_get_row:nnTF
{
  \int_compare:nNnTF
   { #2 } < { \c_one_int }
   {
     \PackageError { datatool }
      {
        \token_to_str:N \dtlgetrow : ~ invalid ~ row ~
        index ~ \int_eval:n { #1 } ~ for ~ database ~
        ` #1 '
```

```
      }
      {
        The ~ row ~ index ~ must ~ be ~ between ~ 1 ~
        and ~ the ~ total ~ number ~ of ~ rows ~ in ~ database
      }
      #4
    }
    {
      \int_compare:nNnTF
       { #2 } > { \DTLrowcount { #1 } }
      {
        \PackageError { datatool }
         {
           \token_to_str:N \dtlgetrow : ~ row ~
           index ~ \int_eval:n { #1 } ~ out ~ of ~ range
            ~ for ~ database ~ ` #1 '
         }
         {
            Database ~ `#1' ~ only ~ has ~ \DTLrowcount{#1} ~ row(s)
         }
         #4
      }
      {
        \int_set:Nn \dtlrownum { #2 }
        \tl_set:Nx \dtldbname { #1 }
        \__datatool_get_row:vx
          { dtldb@#1 } { \int_eval:n { #2 } }
        \int_compare:nNnTF { \l__datatool_row_idx_int } = { -1 }
        {
          \PackageError { datatool }
          {
            \token_to_str:N \dtlgetrow :  ~
            Something's ~ gone ~ wrong. ~ ~
            No ~ such ~ row ~ \int_eval:n { #2 } ~ \c_space_tl ~
            found ~ in ~ database ~ `#1'
          }
          {
            Database ~ `#1' ~ seems ~ to ~ be ~ missing ~ row ~
            \int_eval:n { #2 }
          }
         #4
        }
        { #3 }
      }
    }
}
\cs_new:Nn \datatool_get_row:nnT
{
  \datatool_get_row:nnTF { #1 } { #2 } { #3 } { }
}
```

\edtlgetrowforvalue

```
\edtlgetrowforvalue{⟨db⟩}{⟨column idx⟩}{⟨value⟩}
```

A version of `\dtlgetrowforvalue` that expands its arguments.

```
\newrobustcmd{\edtlgetrowforvalue}[3]{%
  \__datatool_get_row_for_value:xxx
    { #1 } { \int_eval:n { #2 } } { #3 }
}
```

\DTLfetch

```
\DTLfetch{⟨db name⟩}{⟨column1 name⟩}{⟨column1
value⟩}{⟨column2 name⟩}
```

Fetches and displays the value for ⟨*column2 name*⟩ in the first row where the value of ⟨*column1 name*⟩ is ⟨*column1 value*⟩. Note that all arguments are expanded.

```
\NewDocumentCommand \DTLfetch { m m m m }
{
  \edtlgetrowforvalue
    { #1 } { \dtlcolumnindex { #1 } { #2 } } { #3 }
  \dtlgetentryfromcurrentrow
   \dtlcurrentvalue { \dtlcolumnindex { #1 } { #4 } }
  \dtlcurrentvalue
}
```

\dtlgetrowforvalue

```
\dtlgetrowforvalue{⟨db⟩}{⟨column idx⟩}{⟨value⟩}
```

Like `\dtlgetrow`, but gets the row where the entry in column ⟨*column index*⟩ matches ⟨*value*⟩. Produces an error if row not found.

```
\NewDocumentCommand \dtlgetrowforvalue { m m m }
{
  \__datatool_get_row_for_value:xxn
    { #1 } { \int_eval:n { #2 } } { #3 }
}
```

Triggers an error if row not found:

```
\cs_new:Nn \__datatool_get_row_for_value:nnn
{
  \__datatool_get_row_for_value:nnnF { #1 } { #2 } { #3 }
  {
    \PackageError {datatool}
    {
      No ~ row ~ found ~ in ~ database ~ `#1' ~ for ~
      column ~ `\int_eval:n { #2 } ' ~ matching ~
      `\tl_to_str:n { #3 }'
    }
    { }
  }
```

```
}
\cs_generate_variant:Nn
 \__datatool_get_row_for_value:nnn
 { xxn , xvn , xxx }
\cs_new:Nn \__datatool_get_row_for_value:nnnTF
{
  \__datatool_get_row_index:Nnnn
    \l__datatool_row_idx_tl { #1 } { #2 } { #3 }
  \datatool_if_null:NTF \l__datatool_row_idx_tl
   { #5 }
   {
     \int_set:Nn \dtlrownum { \l__datatool_row_idx_tl }
     \tl_set:Nx \dtldbname { #1 }
     \__datatool_get_row:vx
      { dtldb@#1 } { \l__datatool_row_idx_tl }
     #4
   }
}
\cs_generate_variant:Nn
 \__datatool_get_row_for_value:nnnTF
 { xxxTF }
\cs_new:Nn \__datatool_get_row_for_value:nnnT
{
   \__datatool_get_row_for_value:nnnTF
    { #1 } { #2 } { #3 } { #4 } { }
}
\cs_generate_variant:Nn
 \__datatool_get_row_for_value:nnnT
 { xxnT , xvnT , xxxT, VVVT }
\cs_new:Nn \__datatool_get_row_for_value:nnnF
{
   \__datatool_get_row_for_value:nnnTF
    { #1 } { #2 } { #3 } { } { #4 }
}
\cs_generate_variant:Nn
 \__datatool_get_row_for_value:nnnF
 { xxnF , xvnF , xxxF }
```

\@dtlgetrow

\@dtlgetrow{⟨*data specs*⟩}{⟨*row idx*⟩}

Gets the row specs from ⟨*data specs*⟩ for row with index ⟨*row idx*⟩ which must be fully expanded.

```
\newcommand*{\@dtlgetrow}[2]{%
 \__datatool_get_row:nn { #1 } { #2 }
}
\cs_new:Nn \__datatool_get_row:nn
{
```

```
  \tl_if_in:nnTF { #1 } { \db@row@id@w #2\db@row@id@end@ }
  {
    \cs_set:Npn \__datatool_get_row:w ##1% before stuff
       \db@row@elt@w% start of the row
         \db@row@id@w #2\db@row@id@end@% row id
           ##2%
         \db@row@id@w #2\db@row@id@end@% row id
       \db@row@elt@end@% end of the row
           ##3% after stuff
       \q_nil
     {
       \dtlbeforerow = {##1}
       \dtlcurrentrow = {##2}
       \dtlafterrow = {##3}
       \int_set:Nn \l__datatool_row_idx_int { #2 }
     }
     \__datatool_get_row:w #1 \q_nil
   }
   {
```

No such row.

```
     \dtlbeforerow = { }
     \dtlcurrentrow = { }
     \dtlafterrow = { }
     \int_set:Nn \l__datatool_row_idx_int { -1 }
   }
 }
 \cs_generate_variant:Nn \__datatool_get_row:nn
   { vn , vV, vx }
```

__datatool_get_row:nnN{⟨*data specs*⟩}{⟨*row index*⟩}⟨*tl var*⟩

Similar but provide variables in which to store row markup. Before and after content not required. The token list will be cleared if row not found, otherwise it will be set to the row content.

```
 \cs_new:Nn \__datatool_get_row:nnN
 {
  \tl_if_in:nnTF { #1 } { \db@row@id@w #2\db@row@id@end@ }
  {
    \cs_set:Npn \__datatool_get_row:w ##1% before stuff
       \db@row@elt@w% start of the row
         \db@row@id@w #2\db@row@id@end@% row id
           ##2%
         \db@row@id@w #2\db@row@id@end@% row id
       \db@row@elt@end@% end of the row
           ##3% after stuff
       \q_nil
     {
```

```
      \tl_set:Nn #3 { ##2 }
    }
    \__datatool_get_row:w #1 \q_nil
  }
  {
    \tl_clear:N #3
  }
}
\cs_generate_variant:Nn \__datatool_get_row:nnN
 { vnN, vVN }
```

\dtlrecombine

\dtlrecombine

Recombines database contents from \dtlbeforerow, \dtlcurrentrow and \dtlafterrow. Version 3.0: check global option.

```
\newrobustcmd* \dtlrecombine
{
  \__datatool_dtldb_concat:VNxN
    \dtldbname
    \dtlbeforerow
      {
        \__datatool_row_markup:VV
          \dtlrownum
          \dtlcurrentrow
      }
    \dtlafterrow
}
```

\dtlrecombineomitcurrent

\dtlrecombineomitcurrent

Like \dtlrecombine but omits \dtlcurrentrow. Version 3.0: check global option.

```
\newrobustcmd* \dtlrecombineomitcurrent
{
```

Decrement row indices in \dtlafterrow:

```
  \dtl@decrementrows{\dtlafterrow}{\dtlrownum}
```

Reconstruct database contents by concatenating \dtlbeforerow and \dtlafterrow

```
  \__datatool_dtldb_set:Vx
    \dtldbname
    {
      \exp_not:V \dtlbeforerow
      \exp_not:V \dtlafterrow
    }
```

Decrement row counter.

```
  \bool_if:NTF \l__datatool_db_global_bool
```

```
    {
      \int_gdecr:c { dtlrows@ \dtldbname }
    }
    {
      \int_decr:c { dtlrows@ \dtldbname }
    }
  \dtl@message
    {
      Removed ~ row ~ \int_use:N \dtlrownum \c_space_tl ~
      from ~ database ~ ` \dtldbname '
    }
}
```

\dtlsplitrow

\dtlsplitrow{⟨*row specs*⟩}{⟨*col num*⟩}{⟨*before cs*⟩}{⟨*after cs*⟩}

Splits the row around the entry given by ⟨*col num*⟩. The entries before the split are stored in ⟨*before cs*⟩ and the entries after the split are stored in ⟨*after cs*⟩. ⟨*row specs*⟩ and ⟨*col num*⟩ need to be expanded before use.

```
\NewDocumentCommand \dtlsplitrow { m m m m }
{
  \exp_args:Nnx
    \__datatool_split_row:nnNN
    { #1 } { \int_eval:n { #2 } } #3 #4
}
```

Syntax: {⟨*row specs*⟩}{⟨*col idx*⟩}⟨*before var*⟩⟨*after var*⟩

```
\cs_new:Nn \__datatool_split_row:nnNN
{
  \__datatool_if_split_row:nnNNF
   { #1 } { #2 } #3 #4
  {
    \PackageWarning { datatool }
     {
       \token_to_str:N \dtlsplitrow : ~
       no ~ column ~ with ~ index ~ \tl_to_str:n { #2 } ~
       in ~ current ~ row
     }
  }
}
\cs_generate_variant:Nn \__datatool_split_row:nnNN
  { VvNN , VVNN, VxNN, VnNN }
\cs_new:Nn \__datatool_if_split_row:nnNNTF
{
 \tl_if_in:nnTF
   { #1 } { \db@col@id@w #2 \db@col@id@end@ }
  {
```

```
    \cs_set:Npn \__datatool_split_row:w
      ##1 % before stuff
      \db@col@id@w #2\db@col@id@end@ % column id
        ##2 % unwanted stuff
      \db@col@id@w #2\db@col@id@end@ % column id
      ##3 % after stuff
      \q_nil
      {
         \tl_set:Nn #3 { ##1 }
         \tl_set:Nn #4 { ##3 }
         #5
      }
    \__datatool_split_row:w #1 \q_nil
  }
  {
    \tl_set:Nn #3 { #1 }
    \tl_clear:N #4
    #6
  }
}
\cs_new:Nn \__datatool_if_split_row:nnNNT
{
  \__datatool_if_split_row:nnNNTF
    { #1 } { #2 } #3 #4 { #5 } { }
}
\cs_new:Nn \__datatool_if_split_row:nnNNF
{
  \__datatool_if_split_row:nnNNTF
    { #1 } { #2 } #3 #4 { } { #5 }
}
```

replaceentryincurrentrow

\dtlreplaceentryincurrentrow{⟨*new value*⟩}{⟨*col num*⟩}

Replaces entry for column ⟨*col num*⟩ in \dtlcurrentrow with ⟨*new value*⟩

```
\NewDocumentCommand \dtlreplaceentryincurrentrow { m m }
{
  \__datatool_replace_in_current:nx
   { #1 } { \int_eval:n { #2 } }
}

\cs_new:Nn \__datatool_replace_in_current:nn
{
```

Split row

```
  \__datatool_split_row:VnNN
    \dtlcurrentrow
     { #2 }
    \l__datatool_before_tl
    \l__datatool_after_tl
```

Process value according to current options, which will save the value in the token register `\@dtl@toks` and in the token list variable `\l__datatool_item_value_tl`. The datatype will also be set in the `\dtl@datatype` variable.

```
\__datatool_process_new_value:n { #1 }
```

Recombine with new value and store in `\dtlcurrentrow`

```
\__datatool_token_register_set:Nx
 \dtlcurrentrow
  {
    \exp_not:V \l__datatool_before_tl
    \__datatool_row_element_markup:nV
      { #2 } \l__datatool_item_value_tl
    \exp_not:V \l__datatool_after_tl
  }
```

Update column specs. This will set `\l__datatool_item_key_tl`

```
\__datatool_update_meta_data_col_index_with_type:VnV
  \dtldbname { #2 } \@dtl@datatype
\dtl@message
 {
   Updated ~ \l__datatool_item_key_tl \c_space_tl ~ -> ~
   \exp_not:n { #1 } ~ in ~ database ~ ` \dtldbname '
 }
}
\cs_generate_variant:Nn
  \__datatool_replace_in_current:nn
  { nx }
```

lremoveentryincurrentrow

`\dtlremoveentryincurrentrow{`⟨*col idx*⟩`}`

Removes entry for column ⟨*col idx*⟩ from `\dtlcurrentrow`.

```
\NewDocumentCommand \dtlremoveentryincurrentrow { m }
{
```

Split row

```
\__datatool_split_row:VxNN
  \dtlcurrentrow
   { \int_eval:n { #1 } }
  \l__datatool_before_tl
  \l__datatool_after_tl
```

Recombine without given column and store in `\dtlcurrentrow`

```
\__datatool_token_register_set:Nx
  \dtlcurrentrow
 {
   \exp_not:V \l__datatool_before_tl
   \exp_not:V \l__datatool_after_tl
 }
\dtl@message
```

```
    {
      Removed ~ entry ~ from ~ column ~ \int_eval:n { #1 } \c_space_tl ~
       in ~ database ~ `\dtldbname'
    }
  }
```

\dtlswapentriesincurrentrow{⟨*col1 num*⟩}{⟨*col2 num*⟩}

\dtlswapentriesincurrentrow

Swaps columns ⟨*col1 num*⟩ and ⟨*col2 num*⟩ in \dtlcurrentrow

```
\NewDocumentCommand \dtlswapentriesincurrentrow { m m }
{
  \dtlgetentryfromcurrentrow \@dtl@entryI { #1 }
  \dtlgetentryfromcurrentrow \@dtl@entryII { #2 }
  \exp_args:NV \dtlreplaceentryincurrentrow
   \@dtl@entryII { #1 }
  \exp_args:NV \dtlreplaceentryincurrentrow
   \@dtl@entryI { #2 }
}
```

\dtlgetentryfromcurrentrow{⟨*cs*⟩}{⟨*col num*⟩}

\dtlgetentryfromcurrentrow

Gets value for column ⟨*col num*⟩ from \dtlcurrentrow and stores in ⟨*cs*⟩. If not found, ⟨*cs*⟩ is set to \dtlnovalue.

```
\NewDocumentCommand \dtlgetentryfromcurrentrow { m m }
{
  \dtlgetentryfromrow { #1 } { #2 } \dtlcurrentrow
}
```

\dtlgetentryfromrow{⟨*cs*⟩}{⟨*col num*⟩}{⟨*row toks*⟩}

\dtlgetentryfromrow

```
\NewDocumentCommand \dtlgetentryfromrow { m m m }
{
  \__datatool_get_entry_from_row:NxV
    #1 { \int_eval:n { #2 } } #3
}
```

\dtl@getentryfromrow{⟨*cs*⟩}{⟨*col num*⟩}{⟨*row specs*⟩}

\dtl@getentryfromrow

```
\newcommand*{\dtl@getentryfromrow}[3]{%
  \__datatool_get_entry_from_row:Nnn #1 { #2 } { #3 }
}
```

```
\cs_new:Nn \__datatool_get_entry_from_row:Nnn
{
  \cs_set:Npn \__datatool_get_entry_from_row:w
     ##1 % before stuff
    \db@col@id@w #2\db@col@id@end@ % Column id
      \db@col@elt@w ##2\db@col@elt@end@ % Value
    \db@col@id@w #2\db@col@id@end@ % Column id
    ##3 % Remaining stuff
    \q_nil { \def #1 { ##2 } }
  \__datatool_get_entry_from_row:w
    #3
    \db@col@id@w #2 \db@col@id@end@
      \db@col@elt@w \c_datatool_nullvalue_tl \db@col@elt@end@
    \db@col@id@w #2 \db@col@id@end@
    \q_nil
  \__datatool_rm_weird_datum:N #1
}
\cs_generate_variant:Nn
 \__datatool_get_entry_from_row:Nnn
 { NVV, NxV, NVn }
```

lappendentrytocurrentrow

\dtlappendentrytocurrentrow{⟨*key*⟩}{⟨*value*⟩}

Appends entry to \dtlcurrentrow. NB this originally always expanded ⟨*value*⟩. As from v3.0, this will use the same settings as \DTLnewdbentry.

```
\NewDocumentCommand \dtlappendentrytocurrentrow { m m }
{
```

Parse and process value. This will set \l__datatool_item_value_tl and \@dtl@datatype

```
  \__datatool_process_new_value:n { #2 }
```

Update information about this column (adding new column if necessary).

```
  \__datatool_update_meta_data_with_type:nnV
    { \dtldbname } { #1 } \@dtl@datatype
```

Get column index and store in \dtlcolumnnum

```
  \int_set:Nn \dtlcolumnnum
    { \dtlcolumnindex { \dtldbname } { #1 } }
```

Does this row already have an entry with this key?

```
  \exp_args:NNV \dtlgetentryfromcurrentrow
     \dtl@entry \dtlcolumnnum
  \datatool_if_null:NTF \dtl@entry
    {
```

There are no entries in this row for the given key. Append this entry to the current row.

```
      \bool_if:NTF \l__datatool_db_global_bool
      {
        \__datatool_token_register_gput_right:Nx
```

```
      \dtlcurrentrow
    {
      \__datatool_row_element_markup:VV
        \dtlcolumnnum
        \l__datatool_item_value_tl
    }
  }
  {
    \__datatool_token_register_put_right:Nx
      \dtlcurrentrow
    {
      \__datatool_row_element_markup:VV
        \dtlcolumnnum
        \l__datatool_item_value_tl
    }
  }
```

Print information to terminal and log file if in verbose mode.

```
   \dtl@message
     {
       Appended ~ #1 \c_space_tl ->
       \exp_not:V \l__datatool_item_value_tl \c_space_tl ~
       to ~ current ~ row ~ of ~ database ~ `\dtldbname'
     }
 }
 {
```

There is already an entry in this row for the given key

```
   \PackageError {datatool}
    {
      Can't ~ append ~ entry ~ to ~ row: ~ there ~ is ~
      already ~ an ~ entry ~ for ~ key ~ `#1' ~ in ~
      the ~ current ~ row
    }
    {
      Use ~ \token_to_str:N \dtlupdateentryincurrentrow
      \c_space_tl ~ to ~ replace ~ an ~ existing ~ entry
    }
 }
}
```

lupdateentryincurrentrow

\dtlupdateentryincurrentrow{⟨*key*⟩}{⟨*value*⟩}

Appends entry to `\dtlcurrentrow` if column with given key doesn't exist, otherwise updates the value.

```
\NewDocumentCommand \dtlupdateentryincurrentrow { m m }
{
```

Parse and process value. This will set `\l__datatool_item_value_tl` and `\@dtl@datatype`

```
\__datatool_process_new_value:n { #2 }
```

Update information about this column (adding new column if necessary)

```
\__datatool_update_meta_data_with_type:nnV
  { \dtldbname } { #1 } \@dtl@datatype
```

Get column index and store in `\dtlcolumnnum`

```
\int_set:Nn \dtlcolumnnum
  { \dtlcolumnindex { \dtldbname } { #1 } }
```

Does this row already have an entry with this key?

```
\exp_args:NNV \dtlgetentryfromcurrentrow
   \dtl@entry \dtlcolumnnum
\datatool_if_null:NTF \dtl@entry
 {
```

There are no entries in this row for the given key. Append this entry to the current row. NB this used to always globally append. Version 3.0: check boolean setting.

```
    \bool_if:NTF \l__datatool_db_global_bool
     {
       \__datatool_token_register_gput_right:Nx
         \dtlcurrentrow
       {
         \__datatool_row_element_markup:VV
           \dtlcolumnnum
           \l__datatool_item_value_tl
       }
     }
     {
       \__datatool_token_register_put_right:Nx
         \dtlcurrentrow
       {
         \__datatool_row_element_markup:VV
           \dtlcolumnnum
           \l__datatool_item_value_tl
       }
     }
```

Print information to terminal and log file if in verbose mode.

```
    \dtl@message
      {
         Appended ~ #1 \c_space_tl -> ~
         \exp_not:V \l__datatool_item_value_tl
         \c_space_tl ~ to ~ the ~ current ~ row ~ of ~
         database ~ `\dtldbname'
      }
  }
  {
```

There is already an entry in this row for the given key Split row

```
\__datatool_split_row:VVNN
  \dtlcurrentrow
  \dtlcolumnnum
  \l__datatool_before_tl
  \l__datatool_after_tl
```

Recombine with new value and store in `\dtlcurrentrow`

```
\__datatool_token_register_set:Nx
  \dtlcurrentrow
   {
     \exp_not:V \l__datatool_before_tl
     \__datatool_row_element_markup:VV
       \dtlcolumnnum \l__datatool_item_value_tl
     \exp_not:V \l__datatool_after_tl
   }
 \dtl@message
  {
    Updated ~ #1 \c_space_tl ~ -> ~
    \exp_not:V \l__datatool_item_value_tl \c_space_tl ~
     in ~ database ~ ` \dtldbname '
  }
 }
}
```

`\DTLgetvalue`

`\DTLgetvalue{`⟨*cs*⟩`}{`⟨*db*⟩`}{`⟨*r*⟩`}{`⟨*c*⟩`}`

Gets the element in row ⟨*r*⟩, column ⟨*c*⟩ from database ⟨*db*⟩ and stores in ⟨*cs*⟩.

```
\NewDocumentCommand \DTLgetvalue { m m m m }
{
  \__datatool_get_value:Nnxx #1 { #2 }
    { \int_eval:n { #3 } }
    { \int_eval:n { #4 } }
  \datatool_if_null:NT #1
   {
     \PackageError { datatool }
      {
        There ~ is ~ no ~ element ~ at ~ (row=#3, ~ column=#4) ~
        in ~ database ~ `#2'
      }
      { }
   }
}
```

`\dtl@getvalue`

```
\newcommand*{\dtl@getvalue}[4]{%
  \__datatool_get_value:Nnnn #1 { #2 } { #3 } { #4 }
  \datatool_if_null:NT #1
```

```
    {
      \PackageError { datatool }
       {
         There ~ is ~ no ~ element ~ at ~ (row=#3, ~ column=#4) ~
         in ~ database ~ `#2'
       }
       { }
    }
}
```

Syntax: ⟨*tl var*⟩ {⟨*db*⟩}{⟨*r*⟩}{⟨*c*⟩}

```
\cs_new:Nn \__datatool_get_value:Nnnn
{
  \cs_set:Npn \__datatool_get_value:w ##1 % stuff before row <r>
     \db@row@id@w #3\db@row@id@end@ % row <r> id
        ##2 % stuff in row <r> before column <c>
     \db@col@id@w #4\db@col@id@end@ % column <c> id
       \db@col@elt@w ##3\db@col@elt@end@ % value
     \db@col@id@w #4\db@col@id@end@ % column <c> id
     ##4 % stuff after value
     \q_nil
      {
        \tl_set:Nn #1 { ##3 }
      }
  \exp_last_unbraced:Nv \__datatool_get_value:w
     { dtldb@ #2 }
     \db@row@id@w #3 \db@row@id@end@
       \db@col@id@w #4 \db@col@id@end@
       \db@col@elt@w
         \c_datatool_nullvalue_tl % undefined value
       \db@col@elt@end@
       \db@col@id@w #4 \db@col@id@end@
     \q_nil
}
\cs_generate_variant:Nn \__datatool_get_value:Nnnn
  { NnVV , Nnxx }
```

\DTLgetlocation

\DTLgetlocation{⟨*row cs*⟩}{⟨*column cs*⟩}{⟨*database*⟩}
{⟨*value*⟩}

Assigns ⟨*row cs*⟩ and ⟨*column cs*⟩ to the indices of the first entry in ⟨*database*⟩ that matches ⟨*value*⟩.

```
\NewDocumentCommand \DTLgetlocation { m m m m }
{
  \__datatool_get_location:NNnn #1 #2 { #3 } { #4 }
  \datatool_if_null:NT #1
   {
    \PackageError { datatool }
```

```
      {
        There ~ is ~ no ~ element ~
        ` \exp_not:n { #4 } ' ~ in ~ database ~ `#3'
      }
      { }
    }
}
```

Syntax: ⟨*row tl var*⟩ ⟨*column tl var*⟩ {⟨*database*⟩}{⟨*value*⟩}

```
\cs_new:Nn \__datatool_get_location:NNnn
{
  \tl_set_eq:NN #1 \dtlnovalue
```

Test if the value starts with datum marker.

```
  \tl_if_head_eq_meaning:nNT
      { #4 } \__datatool_datum:nnnn
   {
     \__datatool_get_location:NnnnnwNNn
      #4 \q_stop #1 #2 { #3 }
   }
  \tl_if_eq:NNT #1 \dtlnovalue
   {
```

Try an exact match.

```
      \cs_set:Npn \__datatool_get_location:w
        ##1 % stuff before value
        \db@col@elt@w #4\db@col@elt@end@ % value
        \db@col@id@w ##2\db@col@id@end@ % column id
        ##3 % stuff after this column
        \db@row@id@w ##4\db@row@id@end@ % row id
        ##5 % stuff after row
        \q_nil
         {
           \tl_set:Nn #1 { ##4 }
           \tl_set:Nn #2 { ##2 }
         }
      \exp_last_unbraced:Nv \__datatool_get_location:w
        { dtldb@#3 } % contents of data base
        \db@col@elt@w
        #4 % value
        \db@col@elt@end@
        \db@col@id@w
           \c_datatool_nullvalue_tl % undefined column id
        \db@col@id@end@
        \db@row@id@w
           \c_datatool_nullvalue_tl % undefined row id
        \db@row@id@end@
        \q_nil
   }
  \tl_if_eq:NNT #1 \dtlnovalue
   {
```

Try matching weird datum.

```
\tl_set:Nn \l__datatool_cs_builder_tl
 {
  \cs_set:Npn \__datatool_get_location:w
    ##1 % stuff before value
    \db@col@elt@w
```

Encapsulate value with weird datum

```
    \__datatool_datum:w
 }
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
   \c__datatool_datum_weird_marker_tl
   \exp_not:n { #4 } % string value
   \c__datatool_datum_weird_marker_tl
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 { ##2 } % numeric value
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
   \c__datatool_datum_weird_marker_tl
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 { ##3 } % currency symbol
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
   \c__datatool_datum_weird_marker_tl
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 { ##4 } % data type
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
   \c__datatool_datum_weird_marker_tl
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 {
    \db@col@elt@end@
    \db@col@id@w ##5\db@col@id@end@ % column id
    ##6 % stuff after this column
    \db@row@id@w ##7\db@row@id@end@ % row id
    ##8 % stuff after row
    \q_nil
     {
       \tl_set:Nn #1 { ##7 } % row
       \tl_set:Nn #2 { ##5 } % column
     }
  \exp_last_unbraced:Nv \__datatool_get_location:w
    { dtldb@#3 } % contents of data base
    \db@col@elt@w
 }
```

```
      \tl_put_right:Nx \l__datatool_cs_builder_tl
       {
          \__datatool_weird_datum:nnnn
           { #4 } { } { } { }
       }
      \tl_put_right:Nn \l__datatool_cs_builder_tl
       {
          \db@col@elt@end@
          \db@col@id@w
             \c_datatool_nullvalue_tl % undefined column id
          \db@col@id@end@
          \db@row@id@w
             \c_datatool_nullvalue_tl % undefined row id
          \db@row@id@end@
          \q_nil
       }
      \l__datatool_cs_builder_tl
    }
}
\cs_generate_variant:Nn
 \__datatool_get_location:NNnn
 { NNnx , NNnV }
```

Similarly but to match a value supplied as a datum marker. Arguments: ⟨*datum-marker-cs*⟩`{`⟨*string*⟩`}{`⟨*value*⟩`}{`⟨*currency*⟩`}{`⟨*type*⟩`}` ⟨*extra*⟩`\q_stop` ⟨*row tl var*⟩ ⟨*column tl var*⟩ `{`⟨*database*⟩`}` If ⟨*extra*⟩ isn't empty then there's some trailing material.

```
\cs_new:Npn \__datatool_get_location:NnnnnwNNn
  #1 #2 #3 #4 #5 #6 \q_stop #7 #8 #9
{
 \tl_if_empty:nT { #6 }
  {
```

No trailing material. Just match on the string value.

```
    \__datatool_get_location:NNnn #7 #8 { #9 } { #2 }
  }
}
```

`\DTLgetrowindex`

`\DTLgetrowindex{`⟨*row cs*⟩`}{`⟨*database*⟩`}{`⟨*column index*⟩`}{`⟨*value*⟩`}`

Assigns ⟨*row cs*⟩ to the row index of the first entry in ⟨*database*⟩ where the entry in ⟨*column index*⟩ matches ⟨*value*⟩.

```
\NewDocumentCommand \DTLgetrowindex { s m m m m }
{
  \IfBooleanTF { #1 }
  { \dtlgetrowindex { #2 } { #3 } { #4 } { #5 } }
  { \@DTLgetrowindex { #2 } { #3 } { #4 } { #5 } }
}
```

\@DTLgetrowindex

```
\newcommand{\@DTLgetrowindex}[4]{%
  \__datatool_get_row_index:Nnxn #1
   { #2 } { \int_eval:n { #3 } } { #4 }
  \datatool_if_null:NT #1
   {
    \PackageError {datatool}
    {
      There ~ is ~ no ~ element ~ `#4' ~ for ~ column ~
      \int_eval:n { #3 } ~ in ~ database ~ `#2'
    }
    { }
  }
}
```

\dtlgetrowindex

\dtlgetrowindex{⟨*row cs*⟩}{⟨*database*⟩}{⟨*column index*⟩}{⟨*value*⟩}

As above but doesn't produce an error if not found.

```
\newrobustcmd*{\dtlgetrowindex}[4]{%
  \__datatool_get_row_index:Nnxn #1
   { #2 } { \int_eval:n { #3 } } { #4 }
}
```

\xdtlgetrowindex

\xdtlgetrowindex{⟨*row cs*⟩}{⟨*database*⟩}{⟨*column index*⟩}{⟨*value*⟩}

As above but expands the value.

```
\newcommand*{\xdtlgetrowindex}[4]{%
  \__datatool_get_row_index:Nnxx #1
   { #2 } { \int_eval:n { #3 } } { #4 }
}
```

\@dtlgetrowindex

\@dtlgetrowindex{⟨*row cs*⟩}{⟨*database*⟩}{⟨*column index*⟩}{⟨*value*⟩}

Column index must be fully expanded.

```
\newcommand*{\@dtlgetrowindex}[4]{
  \__datatool_get_row_index:Nnnn #1 { #2 } { #3 } { #4 }
}
```

Arguments: {⟨*row tl var*⟩}{⟨*database*⟩}{⟨*column index*⟩} {⟨*value*⟩} NB the value mustn't contain any groups or other awkward tokens that can't be used in function parameter syntax.

```
\cs_new:Nn \__datatool_get_row_index:Nnnn
{
  \tl_set_eq:NN #1 \dtlnovalue
```

Test if value starts with datum marker:

```
  \tl_if_head_eq_meaning:nNT { #4 } \__datatool_datum:nnnn
   {
     \__datatool_get_row_index:NnnnnwNnn
      #4 \q_stop #1 { #2 } { #3 }
   }
```

Try an exact match.

```
  \tl_if_eq:NNT #1 \dtlnovalue
   {
     \cs_set:Npn \__datatool_get_row_index:w
        ##1 % before
       \db@col@id@w #3\db@col@id@end@ % column id
       \db@col@elt@w #4\db@col@elt@end@ % value
       \db@col@id@w #3\db@col@id@end@ % column id
        ##2 % remainder of row
       \db@row@id@w ##3\db@row@id@end@ % row id
        ##4 % after row
       \q_nil
        {
          \tl_set:Nn #1 { ##3 }
        }
     \exp_last_unbraced:Nv \__datatool_get_row_index:w
       { dtldb@#2 }
       \db@col@id@w #3\db@col@id@end@% column id
        \db@col@elt@w
          #4
        \db@col@elt@end@
       \db@col@id@w #3\db@col@id@end@% column id
       \db@row@id@w
         \c_datatool_nullvalue_tl % undefined row id
       \db@row@id@end@
       \q_nil
   }
  \tl_if_eq:NNT #1 \dtlnovalue
   {
```

Try matching weird datum.

```
     \tl_set:Nn \l__datatool_cs_builder_tl
      {
        \cs_set:Npn \__datatool_get_row_index:w
         ##1 % before
        \db@col@id@w #3\db@col@id@end@ % column id
        \db@col@elt@w
```

Encapsulate value with weird datum

```
         \__datatool_datum:w
```

```
 }
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
   \c__datatool_datum_weird_marker_tl
   \exp_not:n { #4 }
   \c__datatool_datum_weird_marker_tl
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 { ##2 }
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
   \c__datatool_datum_weird_marker_tl
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 { ##3 }
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
   \c__datatool_datum_weird_marker_tl
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 { ##4 }
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
   \c__datatool_datum_weird_marker_tl
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 {
   \db@col@elt@end@
   \db@col@id@w #3\db@col@id@end@ % column id
    ##5 % remainder of row
   \db@row@id@w ##6\db@row@id@end@ % row id
    ##7 % after row
   \q_nil
    {
      \tl_set:Nn #1 { ##6 }
    }
   \exp_last_unbraced:Nv \__datatool_get_row_index:w
     { dtldb@#2 }
    \db@col@id@w #3\db@col@id@end@% column id
     \db@col@elt@w
 }
\tl_put_right:Nx \l__datatool_cs_builder_tl
 {
    \__datatool_weird_datum:nnnn
     { #4 } { } { } { }
 }
\tl_put_right:Nn \l__datatool_cs_builder_tl
 {
     \db@col@elt@end@
    \db@col@id@w #3\db@col@id@end@% column id
```

```
          \db@row@id@w
            \c_datatool_nullvalue_tl % undefined row id
          \db@row@id@end@
          \q_nil
        }
      \l__datatool_cs_builder_tl
    }
}
\cs_generate_variant:Nn
 \__datatool_get_row_index:Nnnn
 { Nnnx , Nxxx, Nxvn, Nnxn }
```

Similarly but to match a value supplied as a datum marker. Arguments: ⟨*datum-marker-cs*⟩`{`⟨*string*⟩`}{`⟨*value*⟩`}{`⟨*currency*⟩`}{`⟨*type*⟩`}` ⟨*extra*⟩`\q_stop{`⟨*row tl var*⟩`}` `{`⟨*database*⟩`}{`⟨*column index*⟩`}` If ⟨*extra*⟩ isn't empty then there's some trailing material.

```
\cs_new:Npn \__datatool_get_row_index:NnnnnwNnn
  #1 #2 #3 #4 #5 #6 \q_stop #7 #8 #9
{
 \tl_set_eq:NN #7 \dtlnovalue
 \tl_if_empty:nT { #6 }
  {
```

First try an exact match.

```
   \tl_set:Nn \l__datatool_cs_builder_tl
    {
      \cs_set:Npn \__datatool_get_row_index:w
       ##1 % before
      \db@col@id@w #9\db@col@id@end@ % column id
      \db@col@elt@w
    }
```

Encapsulate value with weird datum

```
   \tl_put_right:Nx \l__datatool_cs_builder_tl
    {
       \__datatool_weird_datum:nnnn
        { #2 } { #3 } { #4 } { #5 }
    }
   \tl_put_right:Nn \l__datatool_cs_builder_tl
    {
      \db@col@elt@end@
      \db@col@id@w #9\db@col@id@end@ % column id
       ##2 % remainder of row
      \db@row@id@w ##3\db@row@id@end@ % row id
       ##4 % after row
      \q_nil
       {
         \tl_set:Nn #7 { ##3 }
       }
      \exp_last_unbraced:Nv \__datatool_get_row_index:w
        { dtldb@#8 }
```

```
      \db@col@id@w #9\db@col@id@end@% column id
       \db@col@elt@w
   }
  \tl_put_right:Nx \l__datatool_cs_builder_tl
   {
      \__datatool_weird_datum:nnnn
       { #2 } { #3 } { #4 } { #5 }
   }
  \tl_put_right:Nn \l__datatool_cs_builder_tl
   {
      \db@col@elt@end@
      \db@col@id@w #9\db@col@id@end@% column id
      \db@row@id@w
        \c_datatool_nullvalue_tl % undefined row id
      \db@row@id@end@
      \q_nil
   }
  \l__datatool_cs_builder_tl
  \tl_if_eq:NNT #7 \dtlnovalue
   {
```

Try with just the string part.

```
      \tl_set:Nn \l__datatool_cs_builder_tl
       {
         \cs_set:Npn \__datatool_get_row_index:w
          ##1 % before
         \db@col@id@w #9\db@col@id@end@ % column id
         \db@col@elt@w
       }
```

Encapsulate value with weird datum

```
      \tl_put_right:Nx \l__datatool_cs_builder_tl
       {
          \__datatool_weird_datum:nnnn
           { #2 } { ##2 } { ##3 } { ##4 }
       }
      \tl_put_right:Nn \l__datatool_cs_builder_tl
       {
         \db@col@elt@end@
         \db@col@id@w #9\db@col@id@end@ % column id
          ##5 % remainder of row
         \db@row@id@w ##6\db@row@id@end@ % row id
          ##7 % after row
         \q_nil
          {
            \tl_set:Nn #7 { ##6 }
          }
         \exp_last_unbraced:Nv \__datatool_get_row_index:w
           { dtldb@#8 }
          \db@col@id@w #9\db@col@id@end@% column id
           \db@col@elt@w
```

```
      }
     \tl_put_right:Nx \l__datatool_cs_builder_tl
      {
         \__datatool_weird_datum:nnnn
          { #2 } { } { } { }
      }
     \tl_put_right:Nn \l__datatool_cs_builder_tl
      {
         \db@col@elt@end@
         \db@col@id@w #9\db@col@id@end@% column id
         \db@row@id@w
           \c_datatool_nullvalue_tl % undefined row id
         \db@row@id@end@
         \q_nil
      }
    \l__datatool_cs_builder_tl
   }
  }
}
```

12.5 Iterating Through Databases

Temporary variables for use in `\DTLmapdata`

```
\tl_new:N \l__datatool_map_data_tl
\tl_new:N \l__datatool_map_data_loop_body_tl
\tl_new:N \l__datatool_map_data_row_tl
\tl_new:N \l__datatool_map_data_pending_tl
\bool_new:N \l__datatool_map_data_readonly_bool
\bool_new:N \l__datatool_map_data_update_pending_bool
\int_new:N \l__datatool_map_data_max_cols_int
\prop_new:N \l__datatool_map_data_col_type_prop
```

Options for `\DTLmapdata`:

```
\keys_define:nn { datatool/mapdata }
{
  name .str_set_x:N = \dtldbname,
  read-only .bool_set:N = \l__datatool_map_data_readonly_bool,
  allow-edits .bool_set_inverse:N = \l__datatool_map_data_readonly_bool,
}
```

`\DTLmapdata`

`\DTLmapdata[`⟨*options*⟩`]{`⟨*loop*⟩`}`

`\DTLforeach` has a lot of extra overhead. Version 3.0 provides `\DTLmapdata`, which is designed to be simpler but shouldn't be used in a tabular context.

```
\NewDocumentCommand \DTLmapdata
 { O{} +m }
 {
```

Initialise defaults:

```
\tl_set_eq:NN
   \dtldbname \l__datatool_default_dbname_tl
\bool_set_true:N \l__datatool_map_data_readonly_bool
\keys_set:nn { datatool/mapdata } { #1 }
\DTLifdbexists { \dtldbname }
{
  \__datatool_map_data_enable:
```

Get the contents of the database token register:

```
\tl_set:Nv \l__datatool_map_data_tl { dtldb@\dtldbname }
```

Save the loop action:

```
\tl_set:Nn \l__datatool_map_data_loop_body_tl { #2 }
\int_zero:N \dtlrownum
```

Recurse:

```
\exp_after:wN \__datatool_map_data:w
  \l__datatool_map_data_tl
  \db@row@elt@w
   \db@row@id@w \q_recursion_tail \db@row@id@end@
   \db@row@id@w \q_recursion_tail \db@row@id@end@
  \db@row@elt@end@
\q_recursion_stop
\prg_break_point:Nn \__datatool_map_data_break: { }
```

Update database if not read-only.

```
\bool_if:NF \l__datatool_map_data_readonly_bool
{
```

Add any pending updates.

```
  \__datatool_append_map_data_pending:
```

Append any remaining. The row indexes may be out if any rows have been removed, so iterate.

```
  \int_step_inline:nnn
    { \dtlrownum + 1 } { \DTLrowcount { \dtldbname } }
  {
    \exp_args:Nv \@dtlgetrow { dtldb@\dtldbname } { ##1 }
    \int_incr:N \l__datatool_row_idx_int
    \tl_put_right:Nx \l__datatool_map_data_pending_tl
     {
       \__datatool_row_markup:VV
         \l__datatool_row_idx_int
         \dtlcurrentrow
    }
  }
```

Have any column types changed?

```
  \tl_clear:N \l__datatool_tmpa_tl
  \prop_if_empty:NF
       \l__datatool_map_data_col_type_prop
```

```
{
  \int_step_inline:nn
    { \l__datatool_map_data_max_cols_int }
  {
```

Get the column metadata if it exists.

```
    \__datatool_get_col_data:vn
      { dtlkeys@ \dtldbname }
      { ##1 }
```

Has the metadata been updated for this column?

```
    \__datatool_map_data_get_col_type_prop:n { ##1 }
```

Ensure the data type is an integer.

```
    \tl_if_empty:NTF \l__datatool_item_type_tl
     {
       \int_set_eq:NN
         \l__datatool_item_type_int
         \c_datatool_unknown_int
     }
     {
       \int_set:Nn \l__datatool_item_type_int
        { \l__datatool_item_type_tl }
     }
```

Append markup for this column in the temporary token list.

```
    \tl_put_right:Nx \l__datatool_tmpa_tl
    {
       \__datatool_column_markup:nVVV
        { ##1 }
        \l__datatool_item_key_tl
        \l__datatool_item_type_int
        \l__datatool_item_head_tl
    }
```

Any new columns will need to have the key to column mapping defined:

```
    \tl_if_exist:cF
     {
       dtl@ci@ \dtldbname
       @  \l__datatool_item_key_tl
     }
     {
       \bool_if:NTF \l__datatool_db_global_bool
        {
          \csgdef
           {
             dtl@ci@ \dtldbname
             @ \l__datatool_item_key_tl
           }
           { ##1 }
        }
        {
```

```
                \csdef
                 {
                   dtl@ci@ \dtldbname
                   @ \l__datatool_item_key_tl
                 }
                 { ##1 }
               }
             }
           }
         }
```

Update database:

```
        \__datatool_dtldb_set:VV
          \dtldbname
          \l__datatool_map_data_pending_tl
```

Update the column metadata if it has changed.

```
        \tl_if_empty:NF \l__datatool_tmpa_tl
         {
           \__datatool_dtlkeys_set:VV
             \dtldbname
             \l__datatool_tmpa_tl
         }
        \bool_if:NTF \l__datatool_db_global_bool
         {
```

Update the row count.

```
          \int_gset_eq:cN { dtlrows@\dtldbname }
           \l__datatool_row_idx_int
```

Update the column count.

```
          \int_gset_eq:cN
            { dtlcols@\dtldbname }
            \l__datatool_map_data_max_cols_int
         }
         {
```

Update the row count.

```
          \int_set_eq:cN { dtlrows@\dtldbname }
           \l__datatool_row_idx_int
```

Update the column count.

```
          \int_set_eq:cN
            { dtlcols@\dtldbname }
            \l__datatool_map_data_max_cols_int
         }
      }
```

Disable `\DTLmaponly` commands.

```
      \__datatool_map_data_disable:
    }
    {
```

```
      \PackageError { datatool }
      { database ~ `\dtldbname' ~ does ~ not ~ exist }
      { }
    }
  }

\cs_new:Nn \__datatool_map_data_break:
 { \prg_map_break:Nn \__datatool_map_data_break: { } }
```

`\DTLmapdatabreak` Break out of the `\DTLmapdata` loop. The starred form will discard edits.

```
\NewDocumentCommand \DTLmapdatabreak { s }
 {
   \IfBooleanT { #1 }
   {
     \bool_set_true:N \l__datatool_map_data_readonly_bool
   }
   \__datatool_map_data_break:
 }
```

`DTLenvmapdata` (*env.*) Environment equivalent.

```
\NewDocumentEnvironment { DTLenvmapdata }
 { O{} +b }
 { \DTLmapdata [ #1 ] { #2 } }
 { }
```

Recursive command for `\DTLmapdata`:

```
\cs_new:Npn \__datatool_map_data:w
  \db@row@elt@w
    \db@row@id@w #1 \db@row@id@end@
     #2
    \db@row@id@w #3 \db@row@id@end@
  \db@row@elt@end@
{
  \quark_if_recursion_tail_stop:n { #1 }
  \int_set:Nn \dtlrownum { #1 }
  \bool_if:NF \l__datatool_map_data_readonly_bool
  {
    \bool_set_true:N \l__datatool_map_data_update_pending_bool
  }
  \tl_set:Nn \l__datatool_map_data_row_tl { #2 }
  \l__datatool_map_data_loop_body_tl
  \bool_if:NF \l__datatool_map_data_readonly_bool
  {
    \__datatool_append_map_data_pending:
  }
  \__datatool_map_data:w
}
```

Append current row to pending:

```
\cs_new:Nn \__datatool_append_map_data_pending:
```

```
{
  \bool_if:NT \l__datatool_map_data_update_pending_bool
  {
    \tl_if_empty:NF \l__datatool_map_data_row_tl
    {
      \int_incr:N \l__datatool_row_idx_int
      \tl_put_right:Nx \l__datatool_map_data_pending_tl
       {
         \__datatool_row_markup:VV
           \l__datatool_row_idx_int
           \l__datatool_map_data_row_tl
      }
    }
    \bool_set_false:N \l__datatool_map_data_update_pending_bool
  }
}
```

Command to enable commands that may only be used in `\DTLmapdata`.

```
\cs_new:Nn \__datatool_map_data_enable:
{
```

Enable `\DTLmapget`:

```
  \cs_set_eq:NN
    \__datatool_map_get:n
    \__datatool_map_get_op:n
```

Enable `\DTLmapgetvalues`:

```
  \cs_set_eq:NN
    \__datatool_map_get_values:n
    \__datatool_map_get_values_op:n
  \cs_set_eq:NN
    \__datatool_map_get_values_noerr:n
    \__datatool_map_get_values_noerr_op:n
```

Enable `\DTLmaprow`:

```
  \cs_set_eq:NN
    \__datatool_map_row:Nn
    \__datatool_map_row_op:Nn
```

Enable ‘row aggregate’ action:

```
  \cs_set_eq:cN
    { __datatool_action_row ~ aggregate: }
    \__datatool_action_row_aggregate_op:
```

Adjust settings according to read-only setting.

```
  \bool_if:NTF \l__datatool_map_data_readonly_bool
  {
    \RenewDocumentCommand \DTLrmrow { }
     {
       \__datatool_map_data_cs_noedit:N \DTLrmrow
     }
    \RenewDocumentCommand \DTLsetentry { m }
```

```
      {
        \__datatool_map_data_cs_noedit:N \DTLsetentry
      }
    \RenewDocumentCommand \DTLrmentry { m }
      {
        \__datatool_map_data_cs_noedit:N \DTLrmentry
      }
  }
  {
```

If not read-only, setup token registers to reconstruct database:

```
    \tl_clear:N \l__datatool_map_data_pending_tl
    \int_zero:N \l__datatool_row_idx_int
    \prop_clear:N \l__datatool_map_data_col_type_prop
    \int_set:Nn \l__datatool_map_data_max_cols_int
     { \DTLcolumncount { \dtldbname } }
    \RenewDocumentCommand \DTLrmrow { }
     {
       \__datatool_map_data_rm_row:
     }
    \RenewDocumentCommand \DTLsetentry { m }
     {
       \__datatool_map_data_set_column:n { ##1 }
     }
    \RenewDocumentCommand \DTLrmentry { m }
     {
       \__datatool_map_data_rm_column:n { ##1 }
     }
  }
}
```

Command to disable commands that may only be used in `\DTLmapdata`.

```
\cs_new:Nn \__datatool_map_data_disable:
{
  \cs_set_eq:NN
    \__datatool_map_get:n
    \__datatool_map_get_noop:n
  \cs_set_eq:NN
    \__datatool_map_get_values:n
    \__datatool_map_get_values_noop:n
  \cs_set_eq:NN
    \__datatool_map_get_values_noerr:n
    \__datatool_map_get_values_noop:n
  \cs_set_eq:NN
    \__datatool_map_row:Nn
    \__datatool_map_row_noop:Nn
  \RenewDocumentCommand \DTLrmrow { }
  {
    \__datatool_map_data_cs_noop:N \DTLrmrow
  }
  \RenewDocumentCommand \DTLrmentry { m }
```

```
  {
    \__datatool_map_data_cs_noop:N \DTLrmentry
  }
  \RenewDocumentCommand \DTLsetentry { m }
  {
    \__datatool_map_data_cs_noop:N \DTLsetentry
  }
  \tl_clear:N \l__datatool_map_data_tl
  \tl_clear:N \l__datatool_map_data_loop_body_tl
  \tl_clear:N \l__datatool_map_data_row_tl
```

Disable 'row aggregate' action:

```
  \cs_set:cn { __datatool_action_row ~ aggregate: }
  {
    \__datatool_action_noop:Nn
      \DTLmapdata
      { row ~ aggregate: }
  }
}
```

Provide commands to edit data. Note that the database won't be updated until the loop has finished.

```
\cs_new:Nn \__datatool_map_data_cs_noop:N
{
  \PackageError { datatool }
  {
    \token_to_str:N #1 \c_space_tl ~ may ~ only ~ be ~ used ~
    in ~ the ~ loop ~ body ~ of ~ \token_to_str:N \DTLmapdata
  }
  { }
}
```

Read-only no-op:

```
\cs_new:Nn \__datatool_map_data_cs_noedit:N
{
  \PackageError { datatool }
  {
    \token_to_str:N #1 \c_space_tl ~ requires ~ read-only=false ~
    setting ~ in ~ \token_to_str:N \DTLmapdata
  }
  {
    Use ~ \token_to_str:N \DTLmapdata [ read-only=false ] { ... } ~
    or ~  \token_to_str:N \DTLmapdata [ allow-edits=true ] { ... }
  }
}
```

Options for editing commands:

```
\int_new:N \l__datatool_map_data_edit_column_int
\tl_new:N \l__datatool_map_data_edit_key_tl
\tl_new:N \l__datatool_map_data_edit_value_tl
\keys_define:nn { datatool/mapdata/edit }
```

```
{
  value .tl_set:N = \l__datatool_map_data_edit_value_tl,
  expand-value .tl_set_x:N = \l__datatool_map_data_edit_value_tl,
  expand-once-value .code =
   {
     \tl_set:No \l__datatool_map_data_edit_value_tl { #1 }
   },
  column .code:n =
   {
     \int_compare:nNnTF { #1 } > { 0 }
     {
       \int_set:Nn \l__datatool_map_data_edit_column_int { #1 }
     }
     {
       \PackageError { datatool }
       {
          Column ~ index ~ must ~ be ~ greater ~ than ~ 0
       }
       { }
     }
   },
  key .str_set_x:N = \l__datatool_map_data_edit_key_tl,
}
```

Initialise edit options:

```
\cs_new:Nn \__datatool_if_init_mapdata_edit_options:NnT
{
  \int_zero:N \l__datatool_map_data_edit_column_int
  \tl_clear:N \l__datatool_map_data_edit_key_tl
  \tl_set:Nn \l__datatool_map_data_edit_value_tl { \c_novalue_tl }
  \keys_set:nn { datatool/mapdata/edit } { #2 }
  \int_if_zero:nTF \l__datatool_map_data_edit_column_int
  {
    \tl_if_empty:NTF \l__datatool_map_data_edit_key_tl
    {
      \PackageError { datatool }
      {
        Missing ~ column ~ identifier ~ in ~ \token_to_str:N #1
      }
      {
        You ~ need ~ to ~ use ~ `column' ~ or ~ `key' ~ to ~
        identify ~ the ~ required ~ column
      }
    }
    {
      \tl_if_exist:cTF
        { dtl@ci@\dtldbname @ \l__datatool_map_data_edit_key_tl }
      {
        \int_set:Nn \l__datatool_map_data_edit_column_int
         { \tl_use:c { dtl@ci@\dtldbname @ \l__datatool_map_data_edit_key_tl } }
```

```
        #3
      }
      {
        \PackageError { datatool }
        {
           No ~ such ~ column ~ `\l__datatool_map_data_edit_key_tl' ~
           in ~ database ~ `\dtldbname' ~ referenced ~ in ~
           \token_to_str:N #1
        }
        { }
      }
    }
  }
  {
    \tl_if_empty:NF \l__datatool_map_data_edit_key_tl
    {
      \tl_if_exist:cT
        { dtl@ci@\dtldbname @ \l__datatool_map_data_edit_key_tl }
      {
        \int_compare:nNnF
          { \l__datatool_map_data_edit_column_int }
           =
          { \tl_use:c { dtl@ci@\dtldbname @ \l__datatool_map_data_edit_key_tl } }
        {
          \PackageWarning { datatool }
          {
            Incompatible ~ options ~
            column=\l__datatool_map_data_edit_column_int ,
            key={ \l__datatool_map_data_edit_key_tl } ~
            (`\l__datatool_map_data_edit_key_tl' ~ maps ~ to ~
             column ~ index ~
             \tl_use:c { dtl@ci@\dtldbname @ \l__datatool_map_data_edit_key_tl }).
             Ignoring `key'
          }
        }
      }
    }
    #3
  }
}
```

`\DTLsetentry` Set the value in the given column.

```
\NewDocumentCommand \DTLsetentry { m }
{
  \__datatool_map_data_cs_noop:N \DTLsetentry
}
\cs_new:Nn \__datatool_map_data_set_column:n
{
  \__datatool_if_init_mapdata_edit_options:NnT \DTLsetentry { #1 }
  {
```

```
\exp_args:No
 \tl_if_novalue:nTF { \l__datatool_map_data_edit_value_tl }
{
  \PackageError { datatool }
  {
    Missing ~ value ~ in ~ \token_to_str:N \DTLsetentry
  }
  {
     Use ~ value={} ~ to ~ set ~ an ~ empty ~ value ~
     or ~ use ~ \token_to_str:N \DTLrmentry \c_space_tl ~ to ~
     remove ~ a ~ column
  }
}
{
```

Save the value in the token register `\@dtl@toks` and in the token list variable `\l__datatool_item_value_tl` according to new-value-expand, trim and store datum settings.

```
    \__datatool_process_new_value:V
      \l__datatool_map_data_edit_value_tl
```

Does the current row already have this column set?

```
    \exp_args:NNx \tl_if_in:NnTF
       \l__datatool_map_data_row_tl
       {
          \exp_not:N \db@col@id@w
            \int_use:N \l__datatool_map_data_edit_column_int
          \exp_not:N \db@col@id@end@
       }
    {
```

Replace existing entry.

```
      \exp_args:NVV \dtlsplitrow
         \l__datatool_map_data_row_tl
         \l__datatool_map_data_edit_column_int
         { \l__datatool_map_data_edit_before_tl }
         { \l__datatool_map_data_edit_after_tl }
      \tl_set_eq:NN
         \l__datatool_map_data_row_tl
         \l__datatool_map_data_edit_before_tl
      \tl_put_right:Nx
         \l__datatool_map_data_row_tl
         {
           \__datatool_row_element_markup:VV
             \l__datatool_map_data_edit_column_int
             \l__datatool_item_value_tl
         }
      \tl_put_right:NV
         \l__datatool_map_data_row_tl
         \l__datatool_map_data_edit_after_tl
    }
```

```
      {
```
Append new entry.
```
        \tl_put_right:Nx
          \l__datatool_map_data_row_tl
          {
            \__datatool_row_element_markup:VV
              \l__datatool_map_data_edit_column_int
              \l__datatool_item_value_tl
          }
        \int_compare:nNnT
          { \l__datatool_map_data_edit_column_int }
           >
         { \l__datatool_map_data_max_cols_int }
        {
          \int_set_eq:NN
            \l__datatool_map_data_max_cols_int
            \l__datatool_map_data_edit_column_int
        }
      }
```
Does the metadata for this column need updating?
```
      \tl_if_empty:NTF \l__datatool_map_data_edit_key_tl
      {
        \tl_clear:N \l__datatool_item_key_tl
      }
      {
        \tl_set_eq:NN
          \l__datatool_item_key_tl
          \l__datatool_map_data_edit_key_tl
      }
      \int_set_eq:NN \l__datatool_item_type_int \@dtl@datatype
      \tl_clear:N \l__datatool_item_head_tl
```
Has this column already been changed in a previous row?
```
      \__datatool_map_data_get_col_type_prop:V
        \l__datatool_map_data_edit_column_int
```
Update property.
```
      \__datatool_map_data_put_col_type_prop:VVV
        \l__datatool_map_data_edit_column_int
        \l__datatool_item_type_tl
        \l__datatool_item_key_tl
    }
  }
}
```

Add column type property. Syntax: `{`⟨*col-index*⟩`}{`⟨*col-type*⟩`}{`⟨*col-key*⟩`}` The ⟨*col-key*⟩ may be empty to denote no change or use the default.

```
\cs_new:Nn \__datatool_map_data_put_col_type_prop:nnn
{
```

```
  \prop_put:Nnn \l__datatool_map_data_col_type_prop
  { #1 } { { #2 } { #3 } }
}
\cs_generate_variant:Nn
  \__datatool_map_data_put_col_type_prop:nnn
 { VVV, Vxx, xxx }
```

Get the column type property. NB get the current metadata first which should set `\l__datatool_item_type_int`, `\l__datatool_item_key_tl` and `\l__datatool_item_head_tl`.

```
\cs_new:Nn \__datatool_map_data_get_col_type_prop:n
{
  \tl_clear:N \l__datatool_item_type_tl
  \prop_get:NnNT
    \l__datatool_map_data_col_type_prop
    { #1 }
    \l__datatool_tmpb_tl
  {
    \tl_set:Nx \l__datatool_item_type_tl
      { \exp_after:wN \use_i:nn \l__datatool_tmpb_tl }
    \tl_set:Nx \l__datatool_tmpb_tl
       { \exp_after:wN \use_ii:nn \l__datatool_tmpb_tl }
    \tl_if_empty:NF \l__datatool_tmpb_tl
    {
      \tl_set_eq:NN \l__datatool_item_key_tl \l__datatool_tmpb_tl
    }
  }
  \exp_args:No \quark_if_no_value:nT { \l__datatool_item_key_tl }
  {
    \tl_set:Nx \l__datatool_item_key_tl { \dtldefaultkey #1 }
  }
  \exp_args:No \quark_if_no_value:nT { \l__datatool_item_head_tl }
  {
    \tl_set_eq:NN \l__datatool_item_head_tl \l__datatool_item_key_tl
  }
```

Does the new type override the original type?

```
  \tl_if_empty:NTF \l__datatool_item_type_tl
  {
```

New type unknown so retain original.

```
    \tl_set:Nx \l__datatool_item_type_tl
      { \int_use:N \l__datatool_item_type_int }
  }
  {
    \int_compare:nNnT
      { \l__datatool_item_type_tl } > { \c_datatool_string_int }
    {
```

New type not a string, so must be numerical. In which case the higher type dominates.

```
      \int_compare:nNnT
        { \l__datatool_item_type_tl } < { \l__datatool_item_type_int }
```

```
      {
        \tl_set:Nx \l__datatool_item_type_tl
         { \int_use:N \l__datatool_item_type_int }
      }
    }
  }
}
\cs_generate_variant:Nn \__datatool_map_data_get_col_type_prop:n
 { V, x }
```

`\DTLrmentry` Removes the given column.

```
\NewDocumentCommand \DTLrmentry { m }
{
  \__datatool_map_data_cs_noop:N \DTLrmentry
}
\cs_new:Nn \__datatool_map_data_rm_column:n
{
  \int_zero:N \l__datatool_map_data_edit_column_int
  \keys_set:nn { datatool/mapdata/edit } { #1 }
  \int_if_zero:nTF \l__datatool_map_data_edit_column_int
  {
    \PackageError { datatool }
    { Missing ~ column ~ in ~ \token_to_str:N \DTLrmentry}
    { }
  }
  {
```

Does the current row already have this column set?

```
    \exp_args:NNx \tl_if_in:NnT
       \l__datatool_map_data_row_tl
       {
          \exp_not:N \db@col@id@w
            \int_use:N \l__datatool_map_data_edit_column_int
          \exp_not:N \db@col@id@end@
       }
    {
```

Remove existing entry.

```
      \exp_args:NVV \dtlsplitrow
         \l__datatool_map_data_row_tl
         \l__datatool_map_data_edit_column_int
         { \l__datatool_map_data_edit_before_tl }
         { \l__datatool_map_data_edit_after_tl }
      \tl_set_eq:NN
         \l__datatool_map_data_row_tl
         \l__datatool_map_data_edit_before_tl
      \tl_put_right:NV
         \l__datatool_map_data_row_tl
         \l__datatool_map_data_edit_after_tl
    }
```

```
  }
}
```

`\DTLrmrow` Remove the current row.

```
\NewDocumentCommand \DTLrmrow { }
{
  \__datatool_map_data_cs_noop:N \DTLrmrow
}

\cs_new:Nn \__datatool_map_data_rm_row:
{
  \tl_clear:N \l__datatool_map_data_row_tl
}
```

`\DTLmaprow` `\DTLmaprow{⟨cs⟩}{⟨body⟩}`

Maps over the columns in the current row.

```
\NewDocumentCommand \DTLmaprow { m m }
{
  \__datatool_map_row:Nn #1 { #2 }
}

\cs_new:Nn \__datatool_map_row_noop:Nn
{
  \__datatool_map_data_cs_noop:N \DTLmaprow
}
\cs_set_eq:NN \__datatool_map_row:Nn \__datatool_map_row_noop:Nn
```

Map over each column in the current row:

```
\cs_new:Nn \__datatool_map_row_op:Nn
{
  \cs_set:Nn \__datatool_map_row_loop_body:n
   {
     \tl_set:Nn #1 { ##1 }
     #2
   }
  \exp_after:wN \__datatool_map_row:w
    \l__datatool_map_data_row_tl
    \db@col@id@w \q_recursion_tail \db@col@id@end@
    \db@col@elt@w \db@col@elt@end@
    \db@col@id@w \q_recursion_tail \db@col@id@end@
  \q_recursion_stop
  \prg_break_point:Nn \__datatool_map_row_break: { }
}
```

Break command.

```
\cs_new:Nn \__datatool_map_row_break:
 { \prg_map_break:Nn \__datatool_map_row_break: { } }
```

\DTLmaprowbreak

```
\newcommand{\DTLmaprowbreak}{ \__datatool_map_row_break: }
```

Macro within loop body:

```
\cs_new:Nn \__datatool_map_row_loop_body:n { }
```

Recursive macro:

```
\cs_new:Npn \__datatool_map_row:w
  \db@col@id@w #1 \db@col@id@end@
  \db@col@elt@w #2 \db@col@elt@end@
  \db@col@id@w #3 \db@col@id@end@
{
  \quark_if_recursion_tail_stop:n { #1 }
  \int_set:Nn \dtlcolumnnum { #1 }
  \__datatool_map_row_loop_body:n { #2 }
  \__datatool_map_row:w
}
```

\DTLmapget

\DTLmapget{⟨*key=value options*⟩}

Gets the value of the given key in the current row.

```
\NewDocumentCommand \DTLmapget { m }
{
  \__datatool_map_get:n { #1 }
}
```

The no-op command:

```
\cs_new:Nn \__datatool_map_get_noop:n
{
  \PackageError { datatool }
  {
    \token_to_str:N \DTLmapget \c_space_tl ~ is ~ only ~
    available ~ within ~ \token_to_str:N \DTLmapdata
  }
  { }
}
\cs_set_eq:NN
  \__datatool_map_get:n \__datatool_map_get_noop:n
```

```
\tl_new:N \l__datatool_map_get_return_tl
```

Define options for \DTLmapget:

```
\keys_define:nn { datatool/mapdata }
{
   key .code:n =
     {
       \cs_set:Nn
         \__datatool_map_get_action:
         {
```

```
          \exp_args:Nx
            \__datatool_map_get_from_key:n { #1 }
        }
    },
  key .value_required:n = true,
  column .code:n =
    {
      \cs_set:Nn
        \__datatool_map_get_action:
        {
          \exp_args:Nx
            \__datatool_map_get_from_idx:n { #1 }
        }
    },
  column .value_required:n = true,
  return .code:n =
    {
      \tl_if_empty:nTF { #1 }
      {
        \cs_set:Nn \__datatool_map_get_action_result:
         { \l__datatool_map_get_return_tl }
      }
      {
        \cs_set:Nn \__datatool_map_get_action_result:
         { \tl_set_eq:NN #1 \l__datatool_map_get_return_tl }
      }
    },
  return .default:n = { },
}
```

The actual command:

```
\cs_new:Nn \__datatool_map_get_op:n
{
  \cs_set_eq:NN
    \__datatool_map_get_action:
    \__datatool_map_get_op_missing:
  \cs_set:Nn \__datatool_map_get_action_result:
    { \l__datatool_map_get_return_tl }
```

Parse settings.

```
  \keys_set:nn { datatool/mapdata } { #1 }
```

Fetch the required information.

```
  \__datatool_map_get_action:
```

Save or do the value.

```
  \__datatool_map_get_action_result:
}
```

Missing information

```
\cs_new:Nn \__datatool_map_get_op_missing:
{
```

```
    \PackageError { datatool }
    {
       Missing ~ `key' ~ or ~ `column' ~ option ~ in ~ \DTLmapget
    }
    { }
    \tl_set_eq:NN \l__datatool_map_get_return_tl \dtlnovalue
}
```

Get value from current row given column index:

```
\cs_new:Nn \__datatool_map_get_from_idx:n
{
  \tl_if_empty:NTF \l__datatool_map_data_row_tl
  {
    \tl_set_eq:NN \l__datatool_map_get_return_tl \dtlnovalue
  }
  {
     \exp_args:NNno
       \dtl@getentryfromrow
       \l__datatool_map_get_return_tl
       { #1 }
       \l__datatool_map_data_row_tl
```

Was a value found?

```
      \tl_if_eq:NNT \l__datatool_map_get_return_tl \dtlnovalue
      {
        \__datatool_get_col_type:vn { dtlkeys@ \dtldbname } { #1 }
        \int_case:nnF { \l__datatool_item_type_int }
        {
          { \c_datatool_string_int }
            { \tl_set_eq:NN \l__datatool_map_get_return_tl \DTLstringnull }
          { \c_datatool_unknown_int } { }
        }
        {
          \tl_set_eq:NN \l__datatool_map_get_return_tl \DTLnumbernull
        }
      }
  }
}
```

Get value from current row given column key:

```
\cs_new:Nn \__datatool_map_get_from_key:n
{
  \cs_if_exist:cTF { dtl@ci@ \dtldbname @ #1 }
  {
     \exp_args:Nx
     \__datatool_map_get_from_idx:n
     { \use:c { dtl@ci@ \dtldbname @ #1 } }
  }
  {
    \tl_set_eq:NN \l__datatool_map_get_return_tl \dtlnovalue
  }
```

```
}
```

`\DTLmapgetvalues` Gets multiple values using the same assignment syntax as used by `\DTLforeach`

```
\NewDocumentCommand \DTLmapgetvalues { s m }
{
  \IfBooleanTF { #1 }
   {
     \__datatool_map_get_values_noerr:n { #2 }
   }
   {
     \__datatool_map_get_values:n { #2 }
   }
}
```

The no-op command:

```
\cs_new:Nn \__datatool_map_get_values_noop:n
{
  \PackageError { datatool }
  {
    \token_to_str:N \DTLmapgetvalues \c_space_tl ~ is ~ only ~
    available ~ within ~ \token_to_str:N \DTLmapdata
  }
  { }
}
\cs_set_eq:NN
  \__datatool_map_get_values:n \__datatool_map_get_values_noop:n
```

Gets values from the current row:

```
\tl_new:N \l__datatool_assign_tl
\cs_new:Nn \__datatool_map_get_values_op:n
{
  \keyval_parse:nnn
    { \__datatool_cskey_missing_val:n }
    { \__datatool_map_assign_cskey:nn }
    { #1 }
}
```

As above but no error if key doesn't exist.

```
\cs_new:Nn \__datatool_map_get_values_noerr_op:n
{
  \keyval_parse:nnn
    { \__datatool_cskey_missing_val:n }
    { \__datatool_map_assign_cskey_noerr:nn }
    { #1 }
}
```

`\__datatool_map_assign_cskey:nn{`⟨*cs*⟩`}{`⟨*key*⟩`}`

```
\cs_new:Nn \__datatool_map_assign_cskey:nn
{
  \tl_if_exist:cTF { dtl@ci@\dtldbname @ #2 }
   {
```

```
      \int_set:Nn \l__datatool_col_idx_int
       { \use:c { dtl@ci@\dtldbname @ #2 } }
      \exp_args:NV
       \__datatool_map_get_from_idx:n
       \l__datatool_col_idx_int
      \tl_set_eq:NN #1 \l__datatool_map_get_return_tl
    }
    {
      \PackageError { datatool }
       {
         Unknown ~ column ~ key ~ `#2' ~ for ~ database ~ \dtldbname
       }
       {
         Check ~ the ~ spelling ~ of ~ the ~ column ~ label ~
         in ~ your ~ assignment ~ \tl_to_str:n { #1 } = #2
       }
      \tl_set_eq:NN #1 \dtlnovalue
    }
}
```

As above but no error if column key not defined.

```
\cs_new:Nn \__datatool_map_assign_cskey_noerr:nn
{
  \tl_if_exist:cTF { dtl@ci@\dtldbname @ #2 }
   {
     \int_set:Nn \l__datatool_col_idx_int
      { \use:c { dtl@ci@\dtldbname @ #2 } }
     \exp_args:NV
      \__datatool_map_get_from_idx:n
      \l__datatool_col_idx_int
     \tl_set_eq:NN #1 \l__datatool_map_get_return_tl
   }
   {
     \tl_set_eq:NN #1 \dtlnovalue
   }
}
\ExplSyntaxOff
```

\@dtlforeachrow

```
\@dtlforeachrow(⟨idx cs⟩,⟨row cs⟩)\in{⟨db⟩}
\do{⟨body⟩}
```

Iterates through each row in database. Assigns the current row index to ⟨*idx cs*⟩ and the row specs to ⟨*row cs*⟩

```
\long\def\@dtlforeachrow(#1,#2)\in#3\do#4{%
  \edef\dtl@tmp{\expandafter\the\csname dtldb@#3\endcsname}%
  \expandafter\@dtl@foreachrow\dtl@tmp
    \db@row@elt@w%
      \db@row@id@w \@nil\db@row@id@end@%
```

```
      \db@row@id@w \@nil\db@row@id@end@%
    \db@row@elt@end@%
    \@@{#1}{#2}{#4}\q@nil
}
```

`\@dtl@foreachrow`

```
\long\def\@dtl@foreachrow\db@row@elt@w%
\db@row@id@w #1\db@row@id@end@%
#2\db@row@id@w #3\db@row@id@end@%
\db@row@elt@end@#4\@@#5#6#7\q@nil{%
```

Define control sequence given by `#5`

```
\gdef#5{#1}%
```

Hide the loop body in a macro

```
\gdef\@dtl@loopbody{#7}%
```

Increment level counter to allow for nested loops

```
\global\advance\@dtl@foreach@level by 1\relax
```

Check if we have reached the end of the loop

```
\ifx#5\@nnil
  \expandafter\global\expandafter
    \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
      =\@dtl@foreachnoop
\else
  \gdef#6{#2}%
```

Set up the break function: Make a copy of current break function

```
  \expandafter\let
    \csname @dtl@break@\the\@dtl@foreach@level\endcsname
    \dtlbreak
```

Setup break function for this level

```
  \gdef\dtlbreak{\expandafter\global\expandafter
    \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
      =\@dtl@foreachnoop}%
```

Initialise

```
  \expandafter\global\expandafter
    \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
      =\@dtl@foreachrow
```

Do body of loop

```
  \@dtl@loopbody
```

Restore break function

```
  \expandafter\let\expandafter\dtlbreak
    \csname @dtl@break@\the\@dtl@foreach@level\endcsname
\fi
```

Set up what to do next.

```
\expandafter\let\expandafter\@dtl@foreachnext
  \csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
```

Decrement level counter.

```
   \global\advance\@dtl@foreach@level by -1\relax
```

Repeat loop if necessary.

```
   \@dtl@foreachnext#4\@@{#5}{#6}{#7}\q@nil
 }
```

`\@dtl@foreachnoop`

```
 \long\def\@dtl@foreachnoop#1\@@#2\q@nil{}

 \ExplSyntaxOn
```

Version 3.0: This is a newer alternative to `\dtlforeachkey` implemented in LaTeX3. Note that the unknown type will be -1 not empty so that it can be compared with the integer constants provided by datatool-base. To make it consistent with `\dtlforeachkey` the handler function has four arguments: {⟨*key*⟩}{⟨*col idx*⟩}{⟨*type*⟩}{⟨*header*⟩} In line function:

```
 \cs_new:Nn \datatool_map_keys_inline:nn
 {
   \cs_set:Nn \__datatool_map_keys_fn:nnnn { #2 }
   \tl_set:Nv \l__datatool_keydata_tl { dtlkeys@#1 }
   \tl_if_empty:NF \l__datatool_keydata_tl
   {
     \exp_after:wN \__datatool_map_keys:w
     \l__datatool_keydata_tl
     \db@plist@elt@w
     \db@col@id@w \q_recursion_tail \db@col@id@end@
     \db@key@id@w \db@key@id@end@
     \db@type@id@w \db@type@id@end@
     \db@header@id@w \db@header@id@end@
     \db@col@id@w \q_recursion_tail \db@col@id@end@
     \db@plist@elt@end@
     \q_recursion_stop
   }
 }
```

Function handler:

```
 \cs_new:Nn \datatool_map_keys_function:nN
 {
   \cs_set_eq:NN \__datatool_map_keys_fn:nnnn #2
   \tl_set:Nv \l__datatool_keydata_tl { dtlkeys@#1 }
   \tl_if_empty:NF \l__datatool_keydata_tl
   {
     \exp_after:wN \__datatool_map_keys:w
     \l__datatool_keydata_tl
     \db@plist@elt@w
     \db@col@id@w \q_recursion_tail \db@col@id@end@
     \db@key@id@w \db@key@id@end@
     \db@type@id@w \db@type@id@end@
     \db@header@id@w \db@header@id@end@
```

```
    \db@col@id@w \q_recursion_tail \db@col@id@end@
    \db@plist@elt@end@
    \q_recursion_stop
  }
}
```

Internal recursive command:

```
\cs_new:Npn \__datatool_map_keys:w
    \db@plist@elt@w
    \db@col@id@w #1 \db@col@id@end@
    \db@key@id@w #2 \db@key@id@end@
    \db@type@id@w #3 \db@type@id@end@
    \db@header@id@w #4 \db@header@id@end@
    \db@col@id@w #5 \db@col@id@end@
    \db@plist@elt@end@
{
  \quark_if_recursion_tail_stop:n { #1 }
  \tl_if_empty:nTF { #3 }
  {
    \__datatool_map_keys_fn:nnnn { #2 } { #1 } { \c_datatool_unknown_int } { #4 }
  }
  {
    \__datatool_map_keys_fn:nnnn { #2 } { #1 } { #3 } { #4 }
  }
  \__datatool_map_keys:w
}
\ExplSyntaxOff
```

\dtlforeachkey

```
\dtlforeachkey(⟨key cs⟩,⟨col cs⟩,⟨type cs⟩,⟨header cs⟩)
\in{⟨db⟩}\do{⟨body⟩}
```

Iterates through all the keys in database ⟨*db*⟩. In each iteration, ⟨*key cs*⟩ stores the key, ⟨*col cs*⟩ stores the column index,⟨*type cs*⟩ stores the data type and ⟨*header cs*⟩ stores the header.

```
\long\def\dtlforeachkey(#1,#2,#3,#4)\in#5\do#6{%
  \gdef\@dtl@loopbody{#6}%
  \edef\@dtl@keys{\expandafter\the\csname dtlkeys@#5\endcsname}%
  \expandafter\@dtl@foreachkey\@dtl@keys
    \db@plist@elt@w%
    \db@col@id@w -1\db@col@id@end@%
    \db@key@id@w \db@key@id@end@%
    \db@type@id@w \db@type@id@end@%
    \db@header@id@w \db@header@id@end@%
    \db@col@id@w -1\db@col@id@end@%
    \db@plist@elt@end@%
    \@@{\@dtl@updatefkcs{#1}{#2}{#3}{#4}}\q@nil
}
```

\@dtl@updatefkcs

```
\newcommand*{\@dtl@updatefkcs}[8]{%
  \gdef#1{#5}%
  \gdef#2{#6}%
  \gdef#3{#7}%
  \gdef#4{#8}%
}
```

\@dtl@foreachkey Sets everything globally in case it occurs in a tabular environment Loop body needs to be stored in \@dtl@loopbody. #7 indicates an update macro.

```
\long\def\@dtl@foreachkey\db@plist@elt@w%
\db@col@id@w #1\db@col@id@end@%
\db@key@id@w #2\db@key@id@end@%
\db@type@id@w #3\db@type@id@end@%
\db@header@id@w #4\db@header@id@end@%
\db@col@id@w #5\db@col@id@end@%
\db@plist@elt@end@#6\@@#7\q@nil{%
  \ifnum#1=-1\relax
```

Terminate loop

```
    \let\@dtl@foreachnext\@dtl@foreachnoop
  \else
```

Set up loop variables

```
    #7{#2}{#1}{#3}{#4}%
```

Increment level counter to allow for nested loops

```
    \global\advance\@dtl@foreach@level by 1\relax
```

Set up the break function

```
    \expandafter\let
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
      \dtlbreak
    \gdef\dtlbreak{\expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachnoop}%
```

Initialise

```
    \expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachkey
```

Do body of loop

```
    \@dtl@loopbody
```

Set up what to do next

```
    \expandafter\let\expandafter\@dtl@foreachnext
      \csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
```

Restore break function

```
    \expandafter\let\expandafter\dtlbreak
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
```

Decrement level counter

```
      \global\advance\@dtl@foreach@level by -1\relax
    \fi
```

Recurse if necessary

```
    \@dtl@foreachnext#6\@@{#7}\q@nil
  }
```

```
  \ExplSyntaxOn
```

\dtlforcolumn

\dtlforcolumn{⟨*cs*⟩}{⟨*db*⟩}{⟨*key*⟩}{⟨*body*⟩}

Iterates through column given by ⟨*key*⟩ in database ⟨*db*⟩. ⟨*cs*⟩ is assign to the element of the column in the current iteration. Starred version doesn't check if data base exists

```
  \NewDocumentCommand \dtlforcolumn { s }
  {
   \IfBooleanTF { #1 } { \@sdtlforcolumn } { \@dtlforcolumn }
  }
```

\@dtlforcolumn

```
  \newcommand{\@dtlforcolumn}[4]{
```

Check if data base exists

```
    \DTLifdbexists { #2 }
    {
      \@DTLifhaskey { #2 } { #3 }
      {
        \@sdtlforcolumn { #1 } { #2 } { #3 } { #4 }
      }
```

key not in data base

```
      {
        \PackageError { datatool }
         {
           Database ~ `#2' ~ doesn't ~ contain ~ key ~ `#3'
         }
         { }
      }
    }
    {
      \PackageError { datatool }
      { Database ~ `#2' ~ doesn't ~ exist}
      { }
    }
  }
```

```
  \cs_new:Nn \__datatool_for_column:Nnnn
  {
    \dtl@forcolumn #1 { #2 } { #3 } { #4 }
```

```
}
\cs_generate_variant:Nn  \__datatool_for_column:Nnnn
{ Nven }
```

\@sdtlforcolumn

```
\newcommand{\@sdtlforcolumn}[4]{%
 \__datatool_for_column:Nven #1 { dtldb@#2 }
   { \dtlcolumnindex { #2 } { #3 } } { #4 }
}
```

\dtlforcolumnidx

\dtlforcolumnidx{⟨*cs*⟩}{⟨*db*⟩}{⟨*col num*⟩}{⟨*body*⟩}

Iterates through the column with index ⟨*col num*⟩ in database ⟨*db*⟩. Starred version doesn't check if database exists.

```
\NewDocumentCommand \dtlforcolumnidx { s }
{
  \IfBooleanTF { #1 } { \@sdtlforcolumnidx } { \@dtlforcolumnidx }
}
```

\@sdtlforcolumnidx

```
\newcommand{\@sdtlforcolumnidx}[4]{%
 \__datatool_for_column:Nven #1 { dtldb@#2 }
   { \int_eval:n { #3 } } { #4 }
}

\ExplSyntaxOff
```

\@dtlforcolumnidx

```
\newcommand{\@dtlforcolumnidx}[4]{%
  \DTLifdbexists{#2}%
  {%
    \expandafter\ifnum\csname dtlcols@#2\endcsname<#3\relax
      \PackageError{datatool}{Column index \number#3\space out of
        bounds for database `#2'}{Database `#2' only has
        \expandafter\number\csname dtlcols@#2\endcsname\space
        columns}%
    \else
      \ifnum#3<1\relax
       \PackageError{datatool}{Column index \number#3\space out of
        bounds for database `#2'}{Indices start from 1}%
      \else
        \@sdtlforcolumnidx{#1}{#2}{#3}{#4}%
      \fi
    \fi
  }%
```

data base doesn't exist

```
  {%
```

```
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
```

\dtl@forcolumn

\dtl@forcolumn{⟨*cs*⟩}{⟨*db specs*⟩}{⟨*col num*⟩}{⟨*body*⟩}

⟨*col num*⟩ needs to be fully expanded

```
\newcommand{\dtl@forcolumn}[4]{%
```

make a copy of break function

```
  \let\@dtl@oldbreak\dtlbreak
```

set up break function

```
  \def\dtlbreak{\let\@dtl@forcolnext=\@dtl@forcolnoop}%
```

define loop macro for this column

```
  \def\@dtl@forcolumn##1% before stuff
    \db@col@id@w #3\db@col@id@end@% column index
      \db@col@elt@w ##2\db@col@elt@end@% entry
    \db@col@id@w #3\db@col@id@end@% column index
    ##3% after stuff
    \q@nil{%
      \def#1{##2}% assign value to <cs>
```

check if end of loop

```
      \ifx#1\@nnil
        \let\@dtl@forcolnext=\@dtl@forcolnoop
      \else
```

do body of loop

```
        #4%
        \let\@dtl@forcolnext=\@dtl@forcolumn
      \fi
```

repeat if necessary

```
      \@dtl@forcolnext##3\q@nil
    }%
```

do loop

```
  \@dtl@forcolumn#2%
   \db@col@id@w #3\db@col@id@end@%
     \db@col@elt@w \@nil\db@col@elt@end@%
   \db@col@id@w #3\db@col@id@end@\q@nil
```

restore break function

```
  \let\dtlbreak\@dtl@oldbreak
}
```

\@dtl@forcolnoop

```
\def\@dtl@forcolnoop#1\q@nil{}
```

`\dtlforeachlevel` `\DTLforeach` can only be nested up to three levels. `\dtlforeachlevel` keeps track of the current level.

```
\newcount\dtlforeachlevel
```

The counter DTLrow⟨*n*⟩ keeps track of each row of data during the ⟨*n*⟩ nested `\DTLforeach`. It is only incremented in the conditions (given by the optional argument) are met. In order to work well with hyperref, a parent counter is also defined.

```
\newcounter{DTLrow}
\newcounter{DTLrowi}
\newcounter{DTLrowii}
\newcounter{DTLrowiii}
```

Keep hyperref happy

```
\def\theHDTLrow{\arabic{DTLrow}}
\def\theHDTLrowi{\theHDTLrow.\arabic{DTLrowi}}
\def\theHDTLrowii{\theHDTLrowi.\arabic{DTLrowii}}
\def\theHDTLrowiii{\theHDTLrowii.\arabic{DTLrowiii}}
```

Allow use of these counters outside of `\DTLforeach`.

`\DTLrowreset` Reset counter for the next level.

```
\NewDocumentCommand\DTLrowreset{}{%
 \ifnum\dtlforeachlevel > 2
   \PackageError{datatool}{DTLrow counter level too deep}%
   {The current counter level is \number\dtlforeachlevel}%
 \else
   \refstepcounter{DTLrow}%
   \setcounter{DTLrow\romannumeral\numexpr\dtlforeachlevel+\@ne}{0}%
 \fi
}
```

`\DTLrowincr` Increment counter for the next level.

```
\NewDocumentCommand\DTLrowincr{}{%
 \ifnum\dtlforeachlevel > 2
   \PackageError{datatool}{DTLrow counter level too deep}%
   {The current counter level is \number\dtlforeachlevel}%
 \else
   \refstepcounter{DTLrow\romannumeral\numexpr\dtlforeachlevel+\@ne}%
 \fi
}
```

`\DTLtherow` Show the counter for the next level.

```
\NewDocumentCommand\DTLtherow{}{%
 \ifnum\dtlforeachlevel > 2
   \PackageError{datatool}{DTLrow counter level too deep}%
   {The current counter level is \number\dtlforeachlevel}%
 \else
   \csuse{theDTLrow\romannumeral\numexpr\dtlforeachlevel+\@ne}%
 \fi
}
```

```
\newcount\dtl@rowi
\newcount\dtl@rowii
\newcount\dtl@rowiii
```

```
\ExplSyntaxOn
```

Keep track of filtered rows to help `\DTLiflastrow`:

```
\int_new:N \g__filtered_row_i_int
\int_new:N \g__filtered_row_ii_int
\int_new:N \g__filtered_row_iii_int
```

```
\ExplSyntaxOff
```

```
\newtoks\@dtl@curi
\newtoks\@dtl@previ
\newtoks\@dtl@nexti
\newtoks\@dtl@curii
\newtoks\@dtl@previi
\newtoks\@dtl@nextii
\newtoks\@dtl@curiii
\newtoks\@dtl@previii
\newtoks\@dtl@nextiii
```

`\DTLsavelastrowcount{`⟨*cmd*⟩`}`

`\DTLsaverowcount`

Stores the maximum row count for the last `\DTLforeach`.

```
\newcommand*{\DTLsavelastrowcount}[1]{%
\ifnum\dtlforeachlevel>2\relax
  \def#1{0}%
\else
  \ifnum\dtlforeachlevel<0\relax
    \def#1{0}%
  \else
    \@dtl@tmpcount=\dtlforeachlevel
    \advance\@dtl@tmpcount by 1\relax
    \edef#1{\expandafter\number
      \csname c@DTLrow\romannumeral\@dtl@tmpcount\endcsname}%
  \fi
\fi}
```

`DTLenvforeach` (*env.*) Environment form of `\DTLforeach` (contents are gathered, so verbatim can't be used).

```
\NewDocumentEnvironment{DTLenvforeach}{O{\boolean{true}}mm+b}
{%
  \@DTLforeach[#1]{#2}{#3}{#4}%
}
{}
```

`DTLenvforeach*` (*env.*) Environment form of `\DTLforeach*` (contents are gathered, so verbatim can't be used).

```
\NewDocumentEnvironment{DTLenvforeach*}{O{\boolean{true}}mm+b}
{%
  \@sDTLforeach[#1]{#2}{#3}{#4}%
}
{}
```

NB `\DTLmapdata` is better but `\DTLforeach` retained for backward-compatibility and for use with tabular environments (although there are still situations where it will cause alignment issues).

`\DTLforeach`

`\DTLforeach[`⟨*conditions*⟩`]{`⟨*db name*⟩`}{`⟨*values*⟩`}{`⟨*text*⟩`}`

For each row of data in the database given by ⟨*db name*⟩, do ⟨*text*⟩, if the specified conditions are satisfied. The argument `{`⟨*values*⟩`}` is a comma separated list of ⟨*cmd*⟩`=`⟨*key*⟩ pairs. At the start of each row, each of the commands in this list are set to the value of the entry with the corresponding key ⟨*key*⟩. (`\gdef` is used to ensure `\DTLforeach` works in a tabular environment.) The database may be edited in the unstarred version, in the starred version the database is read only.

```
\newrobustcmd*{\DTLforeach}{\@ifstar\@sDTLforeach\@DTLforeach}

\ExplSyntaxOn
```

Version 3.0: provided common command for starred and unstarred code. Syntax: `{`⟨*conditions*⟩`}{`⟨*db name*⟩`}{`⟨*values*⟩`}{`⟨*text*⟩`}{`⟨*ro*⟩`}` The ⟨*ro*⟩ argument should be a boolean indicating if read-only.

```
\cs_new:Nn \__datatool_for_each:nnnnN
{
```

Check database exists

```
   \DTLifdbexists { #2 }
   {
```

Keep hyperref happy

```
     \refstepcounter {DTLrow}
```

Store database name.

```
     \tl_gset:Nx \@dtl@dbname { #2 }
```

Increment level and check not exceeded 3

```
     \int_gincr:N \dtlforeachlevel
     \int_compare:nNnTF
        { \dtlforeachlevel } > { 3 }
     {
       \PackageError{datatool}
         {
           \token_to_str:N \DTLforeach \c_space_tl ~ nested ~
           too ~ deeply
         }
         {
```

```
        Only ~ 3 ~ levels ~ are ~ allowed
      }
  }
  {
    \@DTLifdbempty { #2 }
```

Do nothing if database is empty

```
  { }
  {
```

Set level dependent information (needs to be global to ensure it works in the tabular environment). Row counter:

```
\int_gzero:c
 { c@DTLrow\romannumeral\dtlforeachlevel }
```

Keep track of the number of filtered rows:

```
\int_gzero:c
 {
    g__filtered_row_
     \romannumeral \dtlforeachlevel
    _int
 }
```

Store previous value of `\DTLiffirstrow`

```
\cs_gset_eq:cN
  { @dtl@iffirstrow\the\dtlforeachlevel }
  \DTLiffirstrow
```

Define current `\DTLiffirstrow`

```
\cs_gset_eq:NN
  \DTLiffirstrow
  \__datatool_foreach_if_first_row:TF
```

Store previous value of `\DTLiflastrow`

```
\cs_gset_eq:cN
  { @dtl@iflastrow\the\dtlforeachlevel }
  \DTLiflastrow
```

Define current `\DTLiflastrow`

```
\cs_gset_eq:NN
  \DTLiflastrow
  \__datatool_foreach_if_last_row:TF
```

Store previous value of `\DTLifoddrow`

```
\cs_gset_eq:cN
  { @dtl@ifoddrow\the\dtlforeachlevel }
  \DTLifoddrow
```

Define current `\DTLifoddrow`

```
\cs_gset_eq:NN
 \DTLifoddrow
 \__datatool_foreach_if_odd_row:TF
```

Store data base name for current level

```
\tl_clear_new:c
  { @dtl@dbname@\romannumeral\dtlforeachlevel }
\tl_gset_eq:cN
  { @dtl@dbname@\romannumeral\dtlforeachlevel }
    \@dtl@dbname
```

Read only setting.

```
\tl_gclear_new:c
     { @dtl@ro@\romannumeral\dtlforeachlevel }
\bool_if:NTF #5
 {
   \tl_gset:cn
     { @dtl@ro@\romannumeral\dtlforeachlevel }
     { \c_one_int }
 }
 {
   \tl_gset:cn
     { @dtl@ro@\romannumeral\dtlforeachlevel }
     { \c_zero_int }
 }
```

Loop through each row. Loop counter given by `\dtl@row`⟨*level*⟩

```
\exp_args:Nc \dtlgforint
  { dtl@row\romannumeral\dtlforeachlevel }
  =1\to\csname dtlrows@#2\endcsname\step1\do
{
```

Get current row from the data base

```
  \exp_args:Nnx
    \dtlgetrow { #2 }
    { \int_use:c { dtl@row\romannumeral\dtlforeachlevel } }
```

Store the current row for this level

```
  \__datatool_token_register_gset_eq:cN
    { @dtl@cur\romannumeral\dtlforeachlevel }
    \dtlcurrentrow
```

Current and previous rows only need to be save if not read only.

```
   \bool_if:NF #5
    {
```

Store the previous rows for this level.

```
      \__datatool_token_register_gset_eq:cN
        { @dtl@prev\romannumeral\dtlforeachlevel }
         \dtlbeforerow
```

Store the subsequent rows for this level.

```
      \__datatool_token_register_gset_eq:cN
        { @dtl@next\romannumeral\dtlforeachlevel }
         \dtlafterrow
    }
```

Assign commands to the required entries

```
\ifblank{#3}{}{\@dtl@assign{#3}{#2}}
```

Do the main body of text if condition is satisfied

```
\ifthenelse { #1 }
{
```

Increment user row counter

```
  \refstepcounter
    { DTLrow \romannumeral \dtlforeachlevel }
  \tl_set:Nx
     \DTLcurrentindex
    {
      \arabic { DTLrow\romannumeral\dtlforeachlevel }
    }
  #4
```

Reconstruct database if not read only.

```
 \bool_if:NF #5
  {
```

Has this row been marked for deletion?

```
    \exp_args:Nx \tl_if_eq:nnTF
      {
        \__datatool_token_register_use:c
         { @dtl@cur\romannumeral \dtlforeachlevel }
      }
      { \@nil }
    {
```

Row needs to be deleted Decrement row indices for rows with a higher index than this one

```
      \exp_args:Ncc
       \dtl@decrementrows
         {
           @dtl@prev
           \romannumeral \dtlforeachlevel
         }
         {
           dtl@row
           \romannumeral \dtlforeachlevel
         }
      \exp_args:Ncc
       \dtl@decrementrows
         {
           @dtl@next
            \romannumeral \dtlforeachlevel
         }
         {
           dtl@row
           \romannumeral \dtlforeachlevel
         }
```

Globally reconstruct data base without this row

```
                \__datatool_token_register_gset:cx
                 { dtldb@#2 }
                 {
                   \__datatool_token_register_use:c
                   {
                      @dtl@prev
                      \romannumeral \dtlforeachlevel
                   }
                   \__datatool_token_register_use:c
                   {
                      @dtl@next
                      \romannumeral \dtlforeachlevel
                   }
                 }
```

Decrement the row count for this database:

```
                \int_gdecr:c { dtlrows@#2 }
```

Decrement the counter for this loop

```
                \int_gdecr:c
                 { dtl@row\romannumeral \dtlforeachlevel }
            }
            {
```

No row deletion. Reconstruct data base.

```
              \__datatool_token_register_set_eq:Nc
                \@dtl@before
                { @dtl@prev\romannumeral \dtlforeachlevel }
              \__datatool_token_register_set_eq:Nc
                \@dtl@after
                {  @dtl@next\romannumeral \dtlforeachlevel }
```

Concatenate.

```
                \__datatool_token_register_gconcat_middle:cNxN
                 { dtldb@#2 }
                 \@dtl@before
                {
```

This row

```
                  \__datatool_row_markup:vv
                   { dtl@row\romannumeral \dtlforeachlevel }
                   { @dtl@cur\romannumeral \dtlforeachlevel }
                  }
                  \@dtl@after
              }
            }
          }
```

Condition not met so just increment the number of filtered rows variable.

```
          {
            \int_gincr:c
```

```
                {
                   g__filtered_row_
                    \romannumeral \dtlforeachlevel
                   _int
                }
             }
          }
```

Restore previous value of `\DTLiffirstrow`

```
        \cs_gset_eq:Nc
          \DTLiffirstrow
          { @dtl@iffirstrow\the\dtlforeachlevel }
```

Restore previous value of `\DTLiflastrow`

```
        \cs_gset_eq:Nc
           \DTLiflastrow
           { @dtl@iflastrow\the\dtlforeachlevel }
```

Restore previous value of `\DTLifoddrow`

```
        \cs_gset_eq:Nc
           \DTLifoddrow
           { @dtl@ifoddrow\the\dtlforeachlevel }
       }
     }
```

Decrement level

```
     \int_gdecr:N \dtlforeachlevel
   }
```

else part (data base doesn't exist):

```
   {
     \PackageError {datatool}
      { Database ~ `#2' ~ doesn't ~ exist }
      {}
   }
 }
```

`\@DTLforeach` `\@DTLforeach` is the unstarred version of `\DTLforeach`. The database is reconstructed to allow for rows to be edited. Use the starred version for faster access.

```
 \newcommand{\@DTLforeach}[4][\boolean{true}]{%
  \__datatool_for_each:nnnnN { #1 } { #2 } { #3 } { #4 }
    \c_false_bool
 }
```

`\@sDTLforeach` `\@sDTLforeach` is the starred version of `\DTLforeach`. The database rows can't be edited.

```
 \newcommand{\@sDTLforeach}[4][\boolean{true}]{
  \__datatool_for_each:nnnnN { #1 } { #2 } { #3 } { #4 }
    \c_true_bool
 }

 \ExplSyntaxOff
```

\@dtlifreadonly

\@dtlifreadonly{⟨*true part*⟩}{⟨*false part*⟩}

Checks if current loop level is read only

```
\newcommand*{\@dtlifreadonly}[2]{%
  \expandafter\ifx
    \csname @dtl@ro@\romannumeral\dtlforeachlevel\endcsname1\relax
```

Read only

```
    #1%
  \else
```

Not read only

```
    #2%
  \fi
}

\ExplSyntaxOn
```

\DTLappendtorow

\DTLappendtorow{⟨*key*⟩}{⟨*value*⟩}

Appends entry to current row. (The current row is given by \@dtl@cur⟨*n*⟩ where ⟨*n*⟩ is roman numeral value of \dtlforeachlevel. One level expansion is applied to ⟨*value*⟩.

```
\newrobustcmd*{\DTLappendtorow}[2]{%
  \int_if_zero:nTF { \dtlforeachlevel }
  {
    \PackageError {datatool}
    {
      \token_to_str:N \DTLappendrow \c_space_tl ~ can ~ only ~ be ~
      used ~ inside ~ \token_to_str:N \DTLforeach
    }
    { }
  }
  {
```

Set \@dtl@thisdb to the current database name:

```
    \tl_set_eq:Nc \@dtl@thisdb
      { @dtl@dbname@\romannumeral\dtlforeachlevel }
```

Check this isn't in \DTLforeach*

```
    \@dtlifreadonly
    {%
      \PackageError {datatool}
       {
         \token_to_str:N \DTLappendtorow \c_space_tl ~ can't ~
         be ~ used ~ inside ~ \token_to_str:N \DTLforeach*
       }
       {
```

```
      The ~ starred ~ version ~ of ~
      \token_to_str:N \DTLforeach \c_space_tl ~ is ~ read ~ only}
  }%
  {%
```

Store current row number in `\dtlrownum`

```
    \int_set_eq:Nc \dtlrownum
      { dtl@row\romannumeral\dtlforeachlevel }
```

Update information about this column (adding new column if necessary)

```
    \@dtl@updatekeys{\@dtl@thisdb}{#1}{#2}%
```

Get column index and store in `\dtlcolumnnum`

```
    \int_set:Nn \dtlcolumnnum
      { \dtlcolumnindex{\@dtl@thisdb}{#1} }
```

Set `\dtlcurrentrow` to the current row

```
    \dtlcurrentrow =
      \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
```

Does this row already have an entry with this key?

```
    \exp_args:NNV
     \dtlgetentryfromcurrentrow
     \dtl@entry
     \dtlcolumnnum
    \datatool_if_null:NTF \dtl@entry
     {
```

There are no entries in this row for the given key. Expand entry value before storing.

```
      \protected@edef\@dtl@tmp{#2}%
      \expandafter\@dtl@toks\expandafter{\@dtl@tmp}%
```

Globally append this entry to the current row.

```
      \__datatool_token_register_gput_right:cx
        { @dtl@cur\romannumeral\dtlforeachlevel }
      {
        \__datatool_row_element_markup:VV
          \dtlcolumnnum
          \@dtl@toks
      }
```

Print information to terminal and log file if in verbose mode.

```
      \dtl@message{Appended ~ #1\space -> ~ \the\@dtl@toks\space to ~
      database ~ `\@dtl@thisdb'}%
    }
    {
```

There is already an entry in this row for the given key

```
      \PackageError {datatool}
       {
         \token_to_str:N \DTLappendtorow \c_space_tl ~
         Can't ~ append ~ entry ~ `\exp_not:n { #2 }' ~ to ~ row: ~
         there ~ is ~ already ~ an ~ entry ~ for ~ key ~
```

```
            `#1' ~ in ~ this ~ row
          }
         { }
      }
    }
  }
}
```

\DTLremoveentryfromrow

\DTLremoveentryfromrow{⟨*key*⟩}

Removes entry given by ⟨*key*⟩ from current row. (The current row is given by \@dtl@cur⟨*n*⟩ where ⟨*n*⟩ is roman numeral value of \dtlforeachlevel.

```
\newrobustcmd*{\DTLremoveentryfromrow}[1]{%
  \int_if_zero:nTF { \dtlforeachlevel }
  {
    \PackageError {datatool}
    {
      \token_to_str:N \DTLremoventryfromrow \c_space_tl ~
      can ~ only ~ be ~ used ~ inside ~ \token_to_str:N \DTLforeach
    }
    {}%
  }
  {
```

Set \@dtl@thisdb to the current database name:

```
    \tl_set_eq:Nv
      \@dtl@thisdb
      { @dtl@dbname@\romannumeral\dtlforeachlevel }
```

Check this isn't in \DTLforeach*

```
    \@dtlifreadonly
    {%
      \PackageError {datatool}
      {
        \token_to_str:N \DTLremoveentryfromrow \c_space_tl
        can't ~ be ~ used ~ inside ~ \token_to_str:N \DTLforeach*
      }
      {
        The ~ starred ~ version ~ of ~
        \token_to_str:N \DTLforeach \c_space_tl ~ is ~ read ~ only
      }%
    }%
    {%
```

Store current row number in \dtlrownum

```
      \int_set_eq:Nc \dtlrownum
        { dtl@row\romannumeral\dtlforeachlevel }
```

Is there a column corresponding to this key?

```
\@DTLifhaskey{\@dtl@thisdb}{#1}%
{%
```

There exists a column for this key, so get the index:

```
  \@dtl@getcolumnindex{\thiscol}{\@dtl@thisdb}{#1}\relax
  \dtlcolumnnum=\thiscol\relax
```

Set `\dtlcurrentrow` to the current row

```
  \dtlcurrentrow =
    \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
```

Does this row have an entry with this key?

```
  \exp_args:NNV
   \dtlgetentryfromcurrentrow
   \dtl@entry
   \dtlcolumnnum
  \datatool_if_null:NTF \dtl@entry
   {
```

This row doesn't contain an entry with this key

```
    \PackageError {datatool}
     {
       Can't ~ remove ~ entry ~ given ~ by ~ `#1' ~
       from ~ current ~ row ~ in ~ database ~
       `\@dtl@thisdb': ~ no ~ such ~ entry
     }
     {
      The ~ current ~ row ~ doesn't ~ contain ~ an ~ entry ~
      for ~ key ~ `#1'
     }%
  }
  {
```

Split the current row around the unwanted entry

```
    \edef\@dtl@dosplitrow{%
      \noexpand\dtlsplitrow{\the\dtlcurrentrow}%
        {\number\dtlcolumnnum}{\noexpand\dtl@pre}%
        {\noexpand\dtl@post}%
    }%
    \@dtl@dosplitrow
```

Reconstruct row without unwanted entry

```
    \expandafter\@dtl@toks\expandafter{\dtl@pre}%
    \expandafter\toks@\expandafter{\dtl@post}%
    \edef\@dtl@tmp{\the\@dtl@toks \the\toks@}%
    \dtlcurrentrow=\expandafter{\@dtl@tmp}%
    \expandafter\global
      \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
        = \dtlcurrentrow
  \dtl@message{Removed ~ entry ~ given ~ by ~ #1\space
    from ~ current ~ row ~ of ~ database ~ `\@dtl@thisdb'}%
  }
```

```
      }
      {
        \PackageError {datatool}
         {
           Can't ~ remove ~ entry ~ given ~ by ~
          `#1' ~ - ~ no ~ such ~ key ~ exists
         }
         {}
      }
    }
  }
}
```

\DTLreplaceentryforrow

\DTLreplaceentryforrow{⟨*key*⟩}{⟨*value*⟩}

Replaces entry given by ⟨*key*⟩ in current row with ⟨*value*⟩. (The current row is given by the token register \@dtl@cur⟨*n*⟩ where ⟨*n*⟩ is roman numeral value of \dtlforeachlevel.

```
\newcommand*{\DTLreplaceentryforrow}[2]{%
  \int_if_zero:nTF { \dtlforeachlevel }
  {
    \PackageError {datatool}
    {
      \token_to_str:N \DTLreplaceentryforrow \c_space_tl ~
      can ~ only ~ be ~ used ~ inside ~ \token_to_str:N \DTLforeach
    }
    {}%
  }
  {
```

Set \@dtl@thisdb to the current database name:

```
    \tl_set_eq:Nc
      \@dtl@thisdb
      { @dtl@dbname@\romannumeral\dtlforeachlevel }
```

Check this isn't in \DTLforeach*

```
    \@dtlifreadonly
    {%
      \PackageError {datatool}
       {
         \token_to_str:N \DTLreplaceentryforrow \c_space_tl ~
         can't ~ be ~ used ~ inside ~ \token_to_str:N \DTLforeach*
       }
       {
         The ~ starred ~ version ~
         of ~ \token_to_str:N \DTLforeach \c_space_tl is ~ read ~ only
       }%
    }%
    {%
```

Store current row number in `\dtlrownum`

```
\int_set_eq:Nc \dtlrownum
  { dtl@row\romannumeral\dtlforeachlevel }
```

Is there a column corresponding to this key?

```
\@DTLifhaskey{\@dtl@thisdb}{#1}%
{%
```

There exists a column for this key, so get the index:

```
  \@dtl@getcolumnindex{\thiscol}{\@dtl@thisdb}{#1}
  \int_set_eq:NN \dtlcolumnnum \thiscol
```

Set `\dtlcurrentrow` to the current row

```
  \dtlcurrentrow =
    \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
```

Does this row have an entry with this key?

```
  \exp_args:NNV
   \dtlgetentryfromcurrentrow
   \dtl@entry
   \dtlcolumnnum
  \datatool_if_null:NTF \dtl@entry
   {
```

This row doesn't contain an entry with this key

```
    \PackageError {datatool}
     {
       Can't ~ replace ~ entry ~ given ~ by ~ `#1' ~
       from ~ current ~ row ~ in ~ database ~ `\@dtl@thisdb': ~
       no ~ such ~ entry
     }
     {
       The ~ current ~ row ~ doesn't ~ contain ~ an ~ entry ~
       for ~ key ~ `#1'
     }%
  }
  {
```

Split the current row around the requested entry

```
    \edef\@dtl@dosplitrow{%
      \noexpand\dtlsplitrow{\the\dtlcurrentrow}%
        {\number\dtlcolumnnum}{\noexpand\dtl@pre}%
        {\noexpand\dtl@post}%
    }%
    \@dtl@dosplitrow
```

Reconstruct row with new value (given by #2).

```
    \protected@edef\@dtl@tmp{#2}%
    \expandafter\@dtl@toks\expandafter{\@dtl@tmp}% new value
    \expandafter\@dtl@before\expandafter{\dtl@pre}%
    \expandafter\@dtl@after\expandafter{\dtl@post}%
    \__datatool_token_register_gconcat_middle:cNxN
```

```
            { @dtl@cur\romannumeral\dtlforeachlevel }
            \@dtl@before
            {
              \__datatool_row_element_markup:VV
                \dtlcolumnnum
                \@dtl@toks
            }
            \@dtl@after
```

Print information to terminal and log file if in verbose mode.

```
          \dtl@message{Updated ~ #1\space -> \the\@dtl@toks\space
            in ~ database ~ `\@dtl@thisdb'}%
          }
       }%
       {%
```

There doesn't exist a column for this key.

```
          \PackageError {datatool}
           {
            Can't ~ replace ~ key ~ `#1' ~ - ~ no ~ such ~
            key ~ in ~ database ~ `\@dtl@thisdb'
           }
           {}%
       }%
     }%
   }
 }
```

\DTLremovecurrentrow

\DTLremovecurrentrow

Removes current row. This just sets the current row to empty

```
\newcommand*{\DTLremovecurrentrow}{%
  \int_if_zero:nTF { \dtlforeachlevel }
  {
    \PackageError {datatool}
    {
      \token_to_str:N \DTLremovecurrentrow\c_space_tl ~
      can ~ only ~ be ~ used ~ inside ~ \token_to_str:N \DTLforeach
    }
    { }
  }
  {
```

Set `\@dtl@thisdb` to the current database name: TODO why is this needed?

```
    \tl_set_eq:Nc
      \@dtl@thisdb
      { @dtl@dbname@\romannumeral\dtlforeachlevel }
```

Check this isn't in `\DTLforeach*`

```
    \@dtlifreadonly
```

```
      {
        \PackageError {datatool}
        {
          \token_to_str:N \DTLreplaceentryforrow\c_space_tl
          can't ~ be ~ used ~ inside ~ \token_to_str:N \DTLforeach*
        }
        {
          The ~ starred ~ version ~ of ~
          \token_to_str:N \DTLforeach \c_space_tl ~ is ~ read ~ only
        }
      }
      {
```

Set the current row to `\@nil` (`\DTLforeach` needs to check for this)

```
        \__datatool_token_register_gset:cn
          { @dtl@cur\romannumeral\dtlforeachlevel }
          { \@nil }
      }
    }
  }

  \ExplSyntaxOff
```

`\DTLaddentryforrow`

`\DTLaddentryforrow{`⟨*db name*⟩`}{`⟨*assign list*⟩`}{`⟨*condition*⟩`}{`⟨*key*⟩`}{`⟨*value*⟩`}`

Adds the entry with key given by ⟨*key*⟩ and value given by ⟨*value*⟩ to the first row in the database ⟨*db name*⟩ which satisfies the condition given by ⟨*condition*⟩. The ⟨*assign list*⟩ is the same as for `\DTLforeach` and may be used to set the values which are to be tested in ⟨*condition*⟩.

```
  \newcommand{\DTLaddentryforrow}[5]{%
```

Iterate through the data base until condition is met

```
    \DTLifdbexists{#1}%
    {%
      \def\@dtl@notdone{\PackageError{datatool}{Unable to add entry
        given by key `#4': condition not met for any row in database
        `#1'}{}}%
```

Iterate through each row

```
      \DTLforeach[#3]{#1}{#2}%
      {%
```

add entry to this row

```
        \DTLappendtorow{#4}{#5}%
```

disable error message

```
        \let\@dtl@notdone\relax
```

break out of loop

```
        \dtlbreak
      }%
      \@dtl@notdone
    }%
    {%
      \PackageError{datatool}{Unable to add entry given by key `#4':
        database `#1' doesn't exist}{}%
    }%
  }
```

\DTLforeachkeyinrow

\DTLforeachkeyinrow{⟨*cmd*⟩}{⟨*text*⟩}

Iterates through each key in the current row of \DTLforeach, and does ⟨*text*⟩.

```
\newcommand*{\DTLforeachkeyinrow}[2]{%
  \ifnum\dtlforeachlevel=0\relax
    \PackageError{datatool}{\string\DTLforeachkeyinrow\space can only
     be used inside \string\DTLforeach}{}%
  \else
```

Set \@dtl@thisdb to the current database name:

```
    \expandafter\let\expandafter\@dtl@thisdb
      \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
```

Iterate through key list

```
    \dtlforeachkey(\dtlkey,\dtlcol,\dtltype,\dtlheader)\in
      \@dtl@thisdb\do{%
```

store row in \dtlcurrentrow (This may get nested so need to do it here instead of outside this loop in case ⟨*text*⟩ changes it.)

```
        \dtlcurrentrow =
          \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
```

Get the value for this key and store in #1

```
        \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
          {\noexpand#1}{\dtlcol}}%
        \dtl@dogetentry
```

Check if null

```
        \ifx#1\dtlnovalue
          \ifnum0\dtltype=0\relax
```

Data type is ⟨*empty*⟩ or 0, so set to string null. TODO: why did v2.13 change this to use \@dtlstringnull and \@dtlnumbernull? (Why not \DTLstringnull and \DTLnumbernull?)

```
            \let#1=\@dtlstringnull
          \else
```

Data type is numerical, so set to number null.

```
            \let#1=\@dtlnumbernull
```

```
        \fi
      \fi
```

Make `#1` global in case this is in a tabular environment (or something similar)

```
      \global\let#1#1%
```

Store loop body so that any scoping commands (such as `&`) don't cause a problem for `\ifx`

```
      \def\@dtl@loop@body{#2}%
      \@dtl@loop@body
    }%
  \fi
}
```

12.6 DTLforeach Conditionals

The following conditionals are only meant to be used within `\DTLforeach` as they depend on the counter `DTLrow`⟨*n*⟩.

```
\ExplSyntaxOn
```

`\DTLiffirstrow`

`\DTLiffirstrow{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Test if the current row is the first row. (This takes ⟨*condition*⟩, the optional argument of `\DTLforeach`, into account, so it may not correspond to row 1 of the database.) Can only be used in `\DTLforeachrow`.

```
\newcommand{\DTLiffirstrow}[2]{
  \PackageError {datatool}
  {
    \token_to_str:N \DTLiffirstrow\c_space_tl ~ can ~ only ~
    be ~ used ~ inside ~ \token_to_str:N \DTLforeach
  }
  {}
}
```

Definition within `\DTLforeach`:

```
\cs_new:Nn \__datatool_foreach_if_first_row:TF
{
  \int_compare:nNnTF
    { \int_use:c { c@DTLrow\romannumeral\dtlforeachlevel } }
     =
    { \c_one_int }
  { #1 } { #2 }
}
```

`\DTLiflastrow`

`\DTLiflastrow{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Checks if the current row is the last row of the database. It doesn't take the condition (the optional argument of `\DTLforeach`) into account, so its possible it may never do ⟨*true part*⟩, as the last row of the database may not meet the condition. It is therefore not very useful and is confusing since it behaves differently to `\DTLiffirstrow` which does take the condition into account, so I have removed its description from the main part of the manual. If you need to use the optional argument of `\DTLforeach`, you will first have to iterate through the database to count up the number of rows which meet the condition, and then do another pass, checking if the current row has reached that number.

```
\newcommand{\DTLiflastrow}[2]{
  \PackageError {datatool}
  {
   \token_to_str:N \DTLiflastrow\c_space_tl ~ can ~ only ~
   be ~ used ~ inside ~ \token_to_str:N \DTLforeach
  }
 { }
}
```

Definition within `\DTLforeach`:

```
\cs_new:Nn \__datatool_foreach_if_last_row:TF
{
  \int_compare:nNnTF
   { \int_use:c { c@DTLrow\romannumeral\dtlforeachlevel } }
    =
   {
     \int_use:c { dtlrows@ \@dtl@dbname }
     -
     \int_use:c
      {
        g__filtered_row_
         \romannumeral \dtlforeachlevel
        _int
      }
   }
  { #1 } { #2 }
}
```

`\DTLifoddrow`

`\DTLifoddrow{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Determines whether the current row is odd (takes the optional argument of `\DTLforeach` into account.)

```
\newcommand{\DTLifoddrow}[2]{%
  \PackageError
   {datatool}
   {
     \token_to_str:N \DTLifoddrow \c_space_tl ~ can ~ only ~
     be ~ used ~ inside ~ \token_to_str:N \DTLforeach
```

```
    }
    { }
}
```

Definition within `\DTLforeach`:

```
\cs_new:Nn \__datatool_foreach_if_odd_row:TF
{
  \int_if_odd:nTF
   { \int_use:c { c@DTLrow\romannumeral\dtlforeachlevel } }
   { #1 } { #2 }
}
\ExplSyntaxOff
```

12.7 Displaying Database

This section defines commands to display the entire database in a tabular or longtable environment.

`\dtlbetweencols` This specifies what to put between the column alignment specifiers.

```
\newcommand*{\dtlbetweencols}{}
```

`\dtlbeforecols` This specifies what to put before the first column alignment specifier.

```
\newcommand*{\dtlbeforecols}{}
```

`\dtlaftercols` This specifies what to put after the last column alignment specifier.

```
\newcommand*{\dtlaftercols}{}
```

`\dtlstringalign` Alignment character for columns containing strings

```
\newcommand*{\dtlstringalign}{l}
```

`\dtlintalign` Alignment character for columns containing integers

```
\newcommand*{\dtlintalign}{r}
```

`\dtlrealalign` Alignment character for columns containing real numbers

```
\newcommand*{\dtlrealalign}{r}
```

`\dtlcurrencyalign` Alignment character for columns containing currency numbers

```
\newcommand*{\dtlcurrencyalign}{r}
```

`\dtldatetimealign` Alignment character for columns containing timestamps.

```
\newcommand*{\dtldatetimealign}{l}
```

`\dtldatealign` Alignment character for columns containing dates.

```
\newcommand*{\dtldatealign}{l}
```

`\dtltimealign` Alignment character for columns containing times.

```
\newcommand*{\dtltimealign}{l}
```

```
\ExplSyntaxOn
```

\dtladdalign

\dtladdalign{⟨*cs*⟩}{⟨*type*⟩}{⟨*col num*⟩}{⟨*max cols*⟩}

Adds tabular column alignment character to ⟨*cs*⟩ for column ⟨*col num*⟩ which contains data type ⟨*type*⟩.

```
\NewDocumentCommand \dtladdalign { m m m m }
{
  \int_compare:nNnTF { #3 } = { 1 }
  {
    \tl_set_eq:NN #1 \dtlbeforecols
  }
  {
    \tl_put_right:NV #1 \dtlbetweencols
  }
  \exp_args:Nx \tl_if_empty:nTF { #2 }
   {
     \int_set_eq:NN
        \l__datatool_tmp_datatype_int \c_datatool_unknown_int
   }
   {
     \int_set:Nn \l__datatool_tmp_datatype_int { #2 }
   }
  \int_case:nnF { \l__datatool_tmp_datatype_int }
  {
    { \c_datatool_string_int }
      {
```

string

```
        \tl_put_right:NV #1 \dtlstringalign
      }
    { \c_datatool_integer_int }
      {
```

integer

```
        \tl_put_right:NV #1 \dtlintalign
      }
    { \c_datatool_decimal_int }
      {
```

real number

```
        \tl_put_right:NV #1 \dtlrealalign
      }
    { \c_datatool_currency_int }
      {
```

currency

```
        \tl_put_right:NV #1 \dtlcurrencyalign
      }
```

```
      { \c_datatool_datetime_int }
        {
```
datetime
```
          \tl_put_right:NV #1 \dtldatetimealign
        }
      { \c_datatool_date_int }
        {
```
date
```
          \tl_put_right:NV #1 \dtldatealign
        }
      { \c_datatool_time_int }
        {
```
time
```
          \tl_put_right:NV #1 \dtltimealign
        }
      { \c_datatool_unknown_int }
        {
```
Unknown type
```
          \tl_put_right:NV #1 \dtlstringalign
          \PackageWarning { datatool }
          {
            Column ~ \int_eval:n { #3 } ~ has ~ no ~
            associated ~ datatype. ~ Assuming ~
            string, ~ but ~ column ~ may ~ be ~ empty
           }
        }
    }
    {
      \tl_put_right:NV #1 \dtlstringalign
      \PackageError { datatool }
        {
           Invalid ~ data ~ type ~ `\int_eval:n{ #2 } ' ~
          for ~ column ~ \int_eval:n { #3 } }
        { }
    }
    \int_compare:nNnT { #3 } = { #4 }
    {
      \tl_put_right:NV #1 \dtlaftercols
    }
  }

  \regex_new:N \l_datatool_colalign_regex
  \regex_set:Nn \l_datatool_colalign_regex { [ m p r c l ] }
```

\dtladdheaderalign

\dtladdheaderalign{⟨*cs*⟩}{⟨*type*⟩}{⟨*col num*⟩}{⟨*max cols*⟩}

Similar but for the column alignment in the header row.

```
\NewDocumentCommand \dtladdheaderalign { m m m m }
{
  \int_compare:nNnTF { #3 } = { 1 }
  {
```

If `\dtlbeforecols` includes `m`, `p`, `r`, `c`, or `l` don't append.

```
    \regex_match:NVF \l_datatool_colalign_regex \dtlbeforecols
     {
       \tl_set_eq:NN #1 \dtlbeforecols
     }
  }
  {
    \tl_put_right:NV #1 \dtlbetweencols
  }
  \tl_put_right:Nn #1 { c }
  \int_compare:nNnT { #3 } = { #4 }
  {
    \regex_match:NVF \l_datatool_colalign_regex \dtlaftercols
     {
       \tl_put_right:NV #1 \dtlaftercols
     }
  }
}
```

`\dtlheaderformat`

`\dtlheaderformat{⟨text⟩}`

Specifies how to format the column title. Pre v3.0 the definition included `\hfil` to centre without affecting vertical lines. This is now dealt with by `\dtlheaderformat`.

```
\newcommand*{\dtlheaderformat}[1]{\textbf{#1}}
```

`\dtlcolumnheader`

`\dtlcolumnheader{⟨align⟩}{⟨text⟩}`

Newer alternative. This needs to expand to avoid a "Misplaced `\omit`" error.

```
\newcommand \dtlcolumnheader [2]
{
   \multicolumn { 1 } { #1 } { \dtlheaderformat{ #2 } }
}
```

`\dtlstringformat`

`\dtlstringformat{⟨text⟩}`

Specifies how to format entries in columns with string data type.

```
\newcommand*{\dtlstringformat}[1]{#1}
```

\dtlnumericformat

\dtlnumericformat{⟨*text*⟩}

Specifies how to format entries in columns with a numeric data type.

```
\newcommand*{\dtlnumericformat}[1]{#1}
```

\dtlintformat

\dtlintformat{⟨*text*⟩}

Specifies how to format entries in columns with integer data type.

```
\newcommand*{\dtlintformat}[1]{\dtlnumericformat{#1}}
```

\dtlrealformat

\dtlrealformat{⟨*text*⟩}

Specifies how to format entries in columns with real data type.

```
\newcommand*{\dtlrealformat}[1]{\dtlnumericformat{#1}}
```

\dtlcurrencyformat

\dtlcurrencyformat{⟨*text*⟩}

Specifies how to format entries in columns with currency data type.

```
\newcommand*{\dtlcurrencyformat}[1]{\dtlnumericformat{#1}}
```

NB the datetime=reformat setting can instead be used to format dates and times. These are just wrappers to change the font etc in tables.

\dtldatetimeformat

\dtldatetimeformat{⟨*text*⟩}

Specifies how to format entries in columns with datetime data type.

```
\newcommand*{\dtldatetimeformat}[1]{#1}
```

\dtldateformat

\dtldateformat{⟨*text*⟩}

Specifies how to format entries in columns with date data type.

```
\newcommand*{\dtldateformat}[1]{#1}
```

\dtltimeformat

\dtltimeformat{⟨*text*⟩}

Specifies how to format entries in columns with time data type.

```
\newcommand*{\dtltimeformat}[1]{#1}
```

`\dtldisplaystarttab` Indicates what to do just after `\begin{tabular}{`⟨*column specs*⟩`}` (e.g. `\hline`).

```
\newcommand*{\dtldisplaystarttab}{}
```

`\dtldisplayendtab` Indicates what to do just before `\end{tabular}`.

```
\newcommand*{\dtldisplayendtab}{}
```

`\dtldisplayafterhead` Indicates what to do after the header row, before the first row of data.

```
\newcommand*{\dtldisplayafterhead}{}
```

`\dtldisplayvalign` Stores the vertical alignment specifier for the tabular environment used in `\DTLdisplaydb`

```
\newcommand*{\dtldisplayvalign}{c}
```

`\dtldisplaystartrow` Indicates what to do at the start of each row (not including the header row or the first row of data).

```
\newcommand*{\dtldisplaystartrow}{}
```

The number of database rows per tabular row.

```
\int_new:N \l_datatool_display_per_row_int
\int_set_eq:NN \l_datatool_display_per_row_int \c_one_int
```

The number of tabular lines (not including header and footer or extra rows inserted by hooks). This value also doesn't take into account filtering. It's simply set to the database row count divided by the above and rounded.

```
\int_new:N \l_datatool_display_tab_rows_int
```

For use in the display hooks, this tests if the given column number is at the start of the tabular row taking replication into account.

Syntax: `{`⟨*row num*⟩`}{`⟨*column num*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

```
\prg_new_conditional:Npnn \datatool_if_row_start:nn #1 #2
  { p, T, F, TF }
{
  \int_compare:nNnTF
    { \l_datatool_display_per_row_int } > { \c_one_int }
   {
     \bool_lazy_and:nnTF
       { \int_compare_p:nNn { #2 } = { \c_one_int } }
       {
         \int_compare_p:nNn
           { \int_mod:nn { #1 } { \l_datatool_display_per_row_int } }
           = { \c_one_int }
       }
      { \prg_return_true: }
      { \prg_return_false: }
   }
   {
     \int_compare:nNnTF { #2 } = { \c_one_int }
      { \prg_return_true: }
      { \prg_return_false: }
```

```
    }
}
```

Function (which must expand to an integer) that obtains the database row index from the step function value that uses `\__datatool_display_db_row_fn:n`.

```
\cs_new:Nn \__datatool_display_row_idx:n { #1 }
```

`\DTLdisplayTBrowidxmap` Convenient function to produce top to bottom instead of left to right arrangement, but only works if there's no filtering.

```
\newcommand \DTLdisplayTBrowidxmap [1]
{
  \int_mod:nn
    { ( #1 - \c_one_int ) }
    { \l_datatool_display_per_row_int }
   * \l_datatool_display_tab_rows_int
   + \int_div_truncate:nn
      { #1 - \c_one_int }
      { \l_datatool_display_per_row_int } + \c_one_int
}
```

`\dtldisplaycr`

```
\newcommand{\dtldisplaycr}{\tabularnewline}
```

`\dtldisplaydbenv`

```
\newcommand{\dtldisplaydbenv}{tabular}
```

`\DTLdisplaydbAddBegin`

`\DTLdisplaydbAddBegin{`*⟨content tl⟩*`}{`*⟨align token list⟩*`}{`*⟨header token list⟩*`}`

```
\NewDocumentCommand \DTLdisplaydbAddBegin { m m m }
{
  \tl_put_right:Nx #1 { \exp_not:N \begin { \dtldisplaydbenv } }
```

Note that `\dtldisplayvalign` needs to expand.

```
  \tl_if_empty:NF \dtldisplayvalign
  {
    \tl_put_right:Nx #1
     {
       [ \dtldisplayvalign ]
     }
  }
  \tl_put_right:Nn #1 {  { #2 }  }
```

Add the header row:

```
  \tl_put_right:NV #1 \dtldisplaystarttab
  \tl_put_right:Nn #1 { #3 }
  \tl_put_right:NV #1 \l_datatool_post_head_tl
```

```
  \tl_put_right:NV #1 \dtldisplaycr
  \tl_put_right:NV #1 \dtldisplayafterhead
}
```

\DTLdisplaydbAddEnd

\DTLdisplaydbAddEnd{⟨*content tl*⟩}

```
\NewDocumentCommand \DTLdisplaydbAddEnd { m }
{
  \tl_if_eq:NnF \l_datatool_foot_tl { \c_novalue_tl }
  {
    \tl_put_right:NV #1 \dtldisplaycr
    \tl_put_right:NV #1 \l_datatool_foot_tl
  }
  \tl_put_right:NV #1 \dtldisplayendtab
  \tl_put_right:Nx #1 { \exp_not:N \end { \dtldisplaydbenv } }
}

\seq_new:N \l__datatool_omit_columns_seq
\seq_new:N \l__datatool_only_columns_seq
\seq_new:N \l__datatool_omit_keys_seq
\seq_new:N \l__datatool_only_keys_seq
\cs_new:Nn \__datatool_if_display_row:NnT { #3 }
\cs_new:Nn \__datatool_display_post_row:Nn { }
\tl_new:N \l__datatool_pre_display_tl
\tl_new:N \l__datatool_init_display_tl
\tl_new:N \l__datatool_user_align_tl
\tl_new:N \l__datatool_user_header_tl
\bool_new:N \l_datatool_include_header_bool
\bool_set_true:N \l_datatool_include_header_bool
```

The following are for `\DTLdisplaylongdb`. These aren't private in case the user wants to customize `\DTLdisplaylongdbAddBegin`. The table caption (no value indicates no caption):

```
\tl_new:N \l_datatool_caption_tl
\tl_set:Nn \l_datatool_caption_tl { \c_novalue_tl }
```

The short form for the lot (no value indicates use the above):

```
\tl_new:N \l_datatool_short_caption_tl
\tl_set:Nn \l_datatool_short_caption_tl { \c_novalue_tl }
```

The continuation caption (no value indicates use the main caption):

```
\tl_new:N \l_datatool_cont_caption_tl
\tl_set:Nn \l_datatool_cont_caption_tl { \c_novalue_tl }
```

The label:

```
\tl_new:N \l_datatool_label_tl
\tl_set:Nn \l_datatool_label_tl { \c_novalue_tl }
```

The footer:

```
\tl_new:N \l_datatool_foot_tl
\tl_set:Nn \l_datatool_foot_tl { \c_novalue_tl }
```

The last footer:

```
\tl_new:N \l_datatool_last_foot_tl
\tl_set:Nn \l_datatool_last_foot_tl { \c_novalue_tl }
```

The following is used for both `\DTLdisplay` and `\DTLdisplaylongdb` after the header but before `\dtldisplaycr` and `\dtldisplayafterhead`:

```
\tl_new:N \l_datatool_post_head_tl
```

Keys for `\DTLdisplaydb*` and `\DTLdisplaylongdb`:

```
\keys_define:nn { datatool/display }
{
  omit-columns .code:n =
    {
      \seq_set_from_clist:Nn \l__datatool_omit_columns_seq { #1 }
      \seq_if_empty:NF \l__datatool_omit_columns_seq
       {
         \seq_clear:N \l__datatool_only_columns_seq
         \seq_clear:N \l__datatool_only_keys_seq
         \seq_clear:N \l__datatool_omit_keys_seq
       }
    },
  only-columns .code:n =
    {
      \seq_set_from_clist:Nn \l__datatool_only_columns_seq { #1 }
      \seq_remove_duplicates:N \l__datatool_only_columns_seq
      \seq_if_empty:NF  \l__datatool_only_columns_seq
       {
         \seq_clear:N \l__datatool_omit_columns_seq
         \seq_clear:N \l__datatool_only_keys_seq
         \seq_clear:N \l__datatool_omit_keys_seq
       }
    },
  omit-keys .code:n =
    {
      \seq_set_from_clist:Nn \l__datatool_omit_keys_seq { #1 }
      \seq_if_empty:NF \l__datatool_omit_keys_seq
       {
         \seq_clear:N \l__datatool_only_columns_seq
         \seq_clear:N \l__datatool_omit_columns_seq
         \seq_clear:N \l__datatool_only_keys_seq
       }
    },
  only-keys .code:n =
    {
      \seq_set_from_clist:Nn \l__datatool_only_keys_seq { #1 }
      \seq_remove_duplicates:N \l__datatool_only_keys_seq
      \seq_if_empty:NF \l__datatool_only_keys_seq
```

```
      {
       \seq_clear:N \l__datatool_only_columns_seq
        \seq_clear:N \l__datatool_omit_columns_seq
        \seq_clear:N \l__datatool_omit_keys_seq
      }
  },
row-condition-inline .cs_set:Np =
    \__datatool_if_display_row:NnT #1 #2 #3 ,
row-condition-function .code:n =
  {
    \cs_set_eq:NN \__datatool_if_display_row:NnT #1
  },
post-row-inline .cs_set:Np =
    \__datatool_display_post_row:Nn #1 #2 ,
post-row-function .code:n =
  {
    \cs_set_eq:NN \__datatool_display_post_row:Nn #1
  } ,
row-idx-map-inline .cs_set:Np =
   \__datatool_display_row_idx:n #1 ,
row-idx-map-function .code:n =
  {
    \cs_set_eq:NN \__datatool_display_row_idx:n #1
  },
init .tl_set:N = \l__datatool_init_display_tl ,
pre-content .tl_set:N = \l__datatool_pre_display_tl ,
pre-head .tl_set:N = \dtldisplaystarttab ,
post-head .tl_set:N = \l_datatool_post_head_tl ,
after-head .tl_set:N = \dtldisplayafterhead ,
align-specs .tl_set:N = \l__datatool_user_align_tl ,
header-row .tl_set:N = \l__datatool_user_header_tl ,
no-header .bool_set_inverse:N = \l_datatool_include_header_bool ,
string-align .tl_set:N = \dtlstringalign,
int-align .tl_set:N = \dtlintalign,
integer-align .tl_set:N = \dtlintalign,
real-align .tl_set:N = \dtlrealalign,
decimal-align .tl_set:N = \dtlrealalign,
currency-align .tl_set:N = \dtlcurrencyalign,
inter-col .tl_set:N = \dtlbetweencols,
pre-col .tl_set:N = \dtlbeforecols,
post-col .tl_set:N = \dtlaftercols,
```

This key is only for the tabular version:

```
tabular-env .choice:,
tabular-env / tabular .code:n =
 {
   \tl_set:Nn \dtldisplaydbenv { tabular }
   \tl_if_eq:NnF \dtldisplayvalign { t }
   {
     \tl_if_eq:NnF \dtldisplayvalign { b }
```

```
      {
        \tl_set:Nn \dtldisplayvalign { c }
      }
    }
  },
 tabular-env / array .code:n =
  {
    \tl_set:Nn \dtldisplaydbenv { array }
    \tl_if_eq:NnF \dtldisplayvalign { t }
    {
      \tl_if_eq:NnF \dtldisplayvalign { b }
      {
        \tl_set:Nn \dtldisplayvalign { c }
      }
    }
  },
 tabular-env / unknown .code:n =
  {
    \cs_if_exist:cTF { #1 }
    {
      \tl_set:Nn \dtldisplaydbenv { #1 }
      \tl_clear:N \dtldisplayvalign
    }
    {
      \PackageError { datatool }
      {
        Unknown ~ environment ~ `#1' ~ specified ~ in ~
        option ~ `tabular-env'
      }
      {
        Check ~ that ~ you ~ have ~ spelt ~ the ~ environment ~
        name ~ correctly ~ and ~ have ~ loaded ~ any ~
        relevant ~ package
      }
    }
  },
 tabular-env .default:n = { tabular },
 tabular-env .groups:n = { tabular },
```

These keys are only for the longtable version:

```
 longtable-env .code:n =
  {
    \cs_if_exist:cTF { #1 }
    {
      \tl_set:Nn \dtldisplaylongdbenv { #1 }
    }
    {
      \PackageError { datatool }
      {
        Unknown ~ environment ~ `#1' ~ specified ~ in ~
```

```
          option ~ `longtable-env'
        }
        {
          Check ~ that ~ you ~ have ~ spelt ~ the ~ environment ~
          name ~ correctly ~ and ~ have ~ loaded ~ any ~
          relevant ~ package
        }
      }
    },
  longtable-env .default:n = { longtable },
  longtable-env .groups:n = { longtable },
  label .tl_set:N = \l_datatool_label_tl ,
  label .groups:n = { longtable },
  caption .tl_set:N = \l_datatool_caption_tl ,
  caption .groups:n = { longtable },
  cont-caption .tl_set:N = \l_datatool_cont_caption_tl ,
  cont-caption .groups:n = { longtable },
  short-caption .tl_set:N = \l_datatool_short_caption_tl ,
  short-caption .groups:n = { longtable },
  foot .tl_set:N = \l_datatool_foot_tl ,
  last-foot .tl_set:N = \l_datatool_last_foot_tl ,
  last-foot .groups:n = { longtable },
  per-row .int_set:N = \l_datatool_display_per_row_int ,
```

Synonyms for backward compatibility:

```
  contcaption .tl_set:N = \l_datatool_cont_caption_tl ,
  contcaption .groups:n = { longtable },
  shortcaption .tl_set:N = \l_datatool_short_caption_tl ,
  shortcaption .groups:n = { longtable },
  lastfoot .tl_set:N = \l_datatool_last_foot_tl ,
  lastfoot .groups:n = { longtable },
  omit .code:n =
    {
      \seq_set_from_clist:Nn \l__datatool_omit_keys_seq { #1 }
      \seq_if_empty:NF \l__datatool_omit_keys_seq
       {
         \seq_clear:N \l__datatool_only_columns_seq
         \seq_clear:N \l__datatool_omit_columns_seq
         \seq_clear:N \l__datatool_only_keys_seq
       }
    },
}
```

\DTLdisplaydb

\DTLdisplaydb[⟨*omit list*⟩]{⟨*db*⟩}

Displays the database ⟨*db*⟩ in a tabular environment. Version 3.0: rewritten so that the content is first built to avoid the problems with having a loop in a tabular context.

```
\NewDocumentCommand \DTLdisplaydb { s o m }
```

```
{
  \DTLifdbexists { #3 }
  {
    \group_begin:
    \IfValueT { #2 }
    {
      \IfBooleanTF { #1 }
      {
        \keys_set_filter:nnnN { datatool/display } { longtable } { #2 }
          \l__datatool_tmpb_tl
        \tl_if_empty:NF \l__datatool_tmpb_tl
        {
          \PackageWarning { datatool }
          {
            Ignoring ~ unsupported ~ \token_to_str:N\DTLdisplaydb* ~
            option(s): ~ \exp_not:o { \l__datatool_tmpb_tl }
          }
        }
      }
      {
        \keys_set:nn { datatool/display } { omit-keys = { #2 } }
      }
    }
    \tl_set:Nx \dtldbname { #3 }
    \__datatool_display_db:
    \group_end:
  }
  {
    \PackageError { datatool }
    { Database ~ `#3' ~ doesn't ~ exist }
    { }
  }
}
```

Internal command:

```
\cs_new:Nn \__datatool_display_db:
{
```

Initialise and check supplied options make sense:

```
  \__datatool_if_display_init:T
  {
    \l__datatool_init_display_tl
```

Start constructing the content token list. Add the begin part and header row:

```
    \exp_args:NNVV \DTLdisplaydbAddBegin \l__datatool_content_tl
      \l__datatool_align_tl \l__datatool_row_tl
```

Add the entry rows. There are two row counts: the row index (as given in the database) and the row number (which may be different if any rows are skipped).

```
    \int_zero:N \dtlrownum
    \int_zero:N \l__datatool_row_idx_int
```

Iterate over all rows:

```
    \__datatool_display_loop:
```

Add the end part:

```
    \DTLdisplaydbAddEnd \l__datatool_content_tl
```

Do the content:

```
    \l__datatool_pre_display_tl
    \l__datatool_content_tl
  }
}
```

Row loop:

```
\cs_new:Nn \__datatool_display_loop:
{
  \int_step_inline:nn
   { \DTLrowcount { \dtldbname } }
   {
     \exp_args:Nx \__datatool_display_db_row_fn:n
      {
        \int_eval:n { \__datatool_display_row_idx:n { ##1 } }
      }
   }
}
```

Initialise and check supplied options make sense:

```
\cs_new:Nn  \__datatool_if_display_init:T
{
  \tl_clear:N \l__datatool_content_tl
  \tl_clear:N \l__datatool_align_tl
  \tl_clear:N \l__datatool_row_tl
  \tl_set:Nv \l__datatool_keydata_tl { dtlkeys@\dtldbname }
```

Keep track of which columns should be included:

```
  \seq_clear:N \l__datatool_column_indexes_seq
```

If only-keys has been set, obtain the corresponding column indexes:

```
  \seq_if_empty:NF \l__datatool_only_keys_seq
  {
    \seq_map_inline:Nn \l__datatool_only_keys_seq
     {
       \tl_if_exist:cTF { dtl@ci@\dtldbname @ ##1 }
       {
         \seq_put_right:Nv \l__datatool_only_columns_seq
           { dtl@ci@\dtldbname @ ##1 }
       }
       {
         \PackageWarning { datatool }
         { Ignoring ~ key ~ `##1' ~ in ~ only-keys ~ option: ~
           no ~ such ~ key ~ in ~ database ~ `\dtldbname'
         }
       }
```

```
    }
  }
```
If omit-keys has been set, obtain the corresponding column indexes:
```
  \seq_if_empty:NF \l__datatool_omit_keys_seq
  {
    \seq_map_inline:Nn \l__datatool_omit_keys_seq
     {
       \tl_if_exist:cTF { dtl@ci@\dtldbname @ ##1 }
       {
         \seq_put_right:Nv \l__datatool_omit_columns_seq
           { dtl@ci@\dtldbname @ ##1 }
       }
       {
         \PackageWarning { datatool }
         { Ignoring ~ key ~ `##1' ~ in ~ omit-keys ~ option: ~
           no ~ such ~ key ~ in ~ database ~ `\dtldbname'
         }
       }
     }
  }
  \seq_if_empty:NTF \l__datatool_only_columns_seq
  {
```
No inclusion list supplied. Columns will be ordered by column index, skipping omitted columns.
```
    \exp_args:NNv \int_set:Nn \l__datatool_max_cols_int { dtlcols@\dtldbname }
    \int_step_inline:nn { \l__datatool_max_cols_int }
     {
       \seq_if_in:NnF \l__datatool_omit_columns_seq { ##1 }
       {
         \seq_put_right:Nn \l__datatool_column_indexes_seq { ##1 }
       }
     }
  }
  {
```
Inclusion list supplied:
```
      \seq_set_eq:NN \l__datatool_column_indexes_seq
        \l__datatool_only_columns_seq
  }
```
Total number of columns to be shown:
```
  \int_set:Nn \l__datatool_max_cols_int
    { \seq_count:N \l__datatool_column_indexes_seq }
```
Make sure there are 1 or more columns.
```
  \int_compare:nNnTF { \l__datatool_max_cols_int } > { \c_zero_int }
  {
```
Get the alignment specifications.
```
    \int_zero:N \dtlcolumnnum
```

```
\tl_if_empty:NF \l__datatool_user_align_tl
{
  \tl_set_eq:NN \l__datatool_align_tl \l__datatool_user_align_tl
}
\tl_if_empty:NTF \l__datatool_user_header_tl
{
  \bool_if:NF \l_datatool_include_header_bool
  {
```

If no header, locally set this to a token list containing a single empty value to skip the default header formation, which is redundant in this case.

```
    \tl_set:Nn \l__datatool_user_header_tl { \c_empty_tl }
  }
}
{
  \tl_set_eq:NN \l__datatool_row_tl \l__datatool_user_header_tl
}
\bool_lazy_or:nnT
  { \tl_if_empty_p:N \l__datatool_user_align_tl }
  { \tl_if_empty_p:N \l__datatool_user_header_tl }
{
  \seq_map_function:NN \l__datatool_column_indexes_seq
    \__datatool_display_db_metadata_fn:n
}
```

If there should be multiple database rows per tabular row, add copies.

```
\int_compare:nNnT
  { \l_datatool_display_per_row_int } > { \c_one_int }
{
  \bool_if:NT \l_datatool_include_header_bool
   {
     \tl_set:Nx \l__datatool_row_tl
      {
        \exp_args:Ne \tl_tail:n
         {
           \prg_replicate:nn { \l_datatool_display_per_row_int }
            { & \exp_not:V \l__datatool_row_tl }
         }
      }
   }
  \tl_set:Nx \l__datatool_align_tl
   {
     \prg_replicate:nn { \l_datatool_display_per_row_int }
      { \exp_not:V \l__datatool_align_tl }
   }
}
```

Calculate database row count divided by replicate count. Note that this doesn't take into account any filtering.

```
\int_compare:nNnTF
   { \l_datatool_display_per_row_int } > { \c_one_int }
```

```
      {
        \int_set:Nn \l_datatool_display_tab_rows_int
          {
            \int_div_truncate:nn
             { \DTLrowcount { \dtldbname } }
             { \l_datatool_display_per_row_int }
          }
        \int_if_zero:nF
          {
            \int_mod:nn
              { \DTLrowcount { \dtldbname } }
              { \l_datatool_display_per_row_int }
          }
         {
           \int_incr:N \l_datatool_display_tab_rows_int
         }
      }
      {
        \int_set:Nn \l_datatool_display_tab_rows_int
          { \DTLrowcount { \dtldbname } }
      }
```

Initialisation successful, do the main body:

```
     #1
   }
   {
```

Trigger an error or warning if no columns. Only a warning is issued if the database is empty (to allow for saving a database at the end of one run to be loaded at the next).

```
     \@DTLifdbempty { \dtldbname }
      {
        \PackageWarning { datatool }
        { Can't ~ display ~ database ~ `\dtldbname': ~ database ~ empty }
      }
      {
        \PackageError { datatool }
        { Can't ~ display ~ database ~ `\dtldbname': ~ no ~ columns ~ available }
        {
          Either ~ the ~ database ~ is ~ empty ~ or ~ all ~ columns ~
          are ~ in ~ the ~ omit ~ list
        }
      }
  }
}
```

Handler for iterating over column data:

```
\cs_new:Nn \__datatool_display_db_metadata_fn:n
{
  \int_incr:N \dtlcolumnnum
  \exp_args:NV \__datatool_get_col_type_header:nn \l__datatool_keydata_tl
   { #1 }
```

```
  \tl_if_empty:NT \l__datatool_user_align_tl
  {
    \dtladdalign \l__datatool_align_tl
     { \l__datatool_item_type_int }
     { \dtlcolumnnum }
     { \l__datatool_max_cols_int }
  }
  \tl_if_empty:NT \l__datatool_user_header_tl
  {
    \tl_if_empty:NF \l__datatool_row_tl
    {
      \tl_put_right:Nn \l__datatool_row_tl { & }
    }
    \tl_clear:N \l__datatool_tmpb_tl
    \dtladdheaderalign \l__datatool_tmpb_tl
     { \l__datatool_item_type_int }
     { \dtlcolumnnum }
     { \l__datatool_max_cols_int }
    \tl_put_right:Nn \l__datatool_row_tl { \dtlcolumnheader }
    \tl_put_right:Nx \l__datatool_row_tl
     { { \exp_not:V \l__datatool_tmpb_tl } }
    \tl_put_right:Nx \l__datatool_row_tl
      { { \exp_not:o { \l__datatool_item_head_tl } } }
  }
}
```

Row handler:

```
\cs_new:Nn \__datatool_display_db_row_fn:n
{
```

Get the current row from the given row index. This sets the `\dtlcurrentrow` token register so that the condition can lookup values in the current row if required.

```
  \exp_args:Nv \@dtlgetrow { dtldb@\dtldbname } { #1 }
  \__datatool_if_display_row:NnT \l__datatool_content_tl { #1 }
  {
    \__datatool_display_db_row:Nn \l__datatool_content_tl { #1 }
  }
}
\cs_new:Nn \__datatool_display_db_row:Nn
{
  \int_compare:nNnTF { \l__datatool_row_idx_int } = { -1 }
  {
    \PackageWarning { datatool }
    { Omitting ~ row ~ #2: ~ not ~ found ~ in ~ database ~ `\dtldbname' }
  }
  {
    \int_incr:N \dtlrownum
    \int_compare:nNnT { \dtlrownum } > { \c_one_int }
     {
       \int_compare:nNnTF
         { \l_datatool_display_per_row_int } > { \c_one_int }
```

```
        {
          \int_compare:nNnTF
            { \int_mod:nn { \dtlrownum } { \l_datatool_display_per_row_int } }
             = { \c_one_int }
           {
             \tl_put_right:NV #1 \dtldisplaycr
             \tl_put_right:NV #1 \dtldisplaystartrow
           }
           {
             \tl_put_right:Nn #1 { & }
           }
        }
        {
          \tl_put_right:NV #1 \dtldisplaycr
          \tl_put_right:NV #1 \dtldisplaystartrow
        }
      }
    \int_zero:N \l__datatool_col_idx_int
    \seq_map_function:NN \l__datatool_column_indexes_seq
      \__datatool_display_db_col_fn:n
    \__datatool_display_post_row:Nn \l__datatool_content_tl { #2 }
  }
}
```

Function handler when iterating over columns within a row.

```
\cs_new:Nn \__datatool_display_db_col_fn:n
{
  \int_compare:nNnT { \l__datatool_col_idx_int } > { 0 }
  {
     \tl_put_right:Nn \l__datatool_content_tl { & }
  }
```

This may not be the same as `#1` if columns are missing or not in sequential order. Use `#1` to reference the column.

```
  \int_incr:N \l__datatool_col_idx_int
```

Get the entry for the current column and row.

```
  \dtlgetentryfromcurrentrow { \l__datatool_item_value_tl } { #1 }
  \exp_args:NV \__datatool_get_col_type:nn \l__datatool_keydata_tl
   { #1 }
```

Ensure that any null values match their data type.

```
  \int_case:nnF { \l__datatool_item_type_int }
  {
    { \c_datatool_string_int }
      {
        \tl_set:Nn \l__datatool_tmpb_tl { \dtlstringformat }
        \datatool_if_null:NT \l__datatool_item_value_tl
         {
           \tl_set_eq:NN \l__datatool_item_value_tl \DTLstringnull
         }
```

```
    }
  { \c_datatool_integer_int }
    {
      \tl_set:Nn \l__datatool_tmpb_tl { \dtlintformat }
      \datatool_if_null:NT \l__datatool_item_value_tl
       {
         \tl_set_eq:NN \l__datatool_item_value_tl \DTLnumbernull
       }
    }
  { \c_datatool_decimal_int }
    {
      \tl_set:Nn \l__datatool_tmpb_tl { \dtlrealformat }
      \datatool_if_null:NT \l__datatool_item_value_tl
       {
         \tl_set_eq:NN \l__datatool_item_value_tl \DTLnumbernull
       }
    }
  { \c_datatool_currency_int }
    {
      \tl_set:Nn \l__datatool_tmpb_tl { \dtlcurrencyformat }
      \datatool_if_null:NT \l__datatool_item_value_tl
       {
         \tl_set_eq:NN \l__datatool_item_value_tl \DTLnumbernull
       }
    }
  { \c_datatool_datetime_int }
    {
      \tl_set:Nn \l__datatool_tmpb_tl { \dtldatetimeformat }
      \datatool_if_null:NT \l__datatool_item_value_tl
       {
         \tl_set_eq:NN \l__datatool_item_value_tl \DTLnumbernull
       }
    }
  { \c_datatool_date_int }
    {
      \tl_set:Nn \l__datatool_tmpb_tl { \dtldateformat }
      \datatool_if_null:NT \l__datatool_item_value_tl
       {
         \tl_set_eq:NN \l__datatool_item_value_tl \DTLnumbernull
       }
    }
  { \c_datatool_time_int }
    {
      \tl_set:Nn \l__datatool_tmpb_tl { \dtltimeformat }
      \datatool_if_null:NT \l__datatool_item_value_tl
       {
         \tl_set_eq:NN \l__datatool_item_value_tl \DTLnumbernull
       }
    }
}
```

```
  {
    \tl_set:Nn \l__datatool_tmpb_tl { \dtlstringformat }
    \datatool_if_null:NT \l__datatool_item_value_tl
     {
       \tl_set_eq:NN \l__datatool_item_value_tl \DTLstringnull
     }
  }
```

Add the item to the content list variable:

```
  \__datatool_display_add_item:NVVVVVVn
    \l__datatool_content_tl
    \l__datatool_item_value_tl  % item
    \l__datatool_tmpb_tl % fmt-cs
    \l__datatool_item_type_int % type
    \dtlrownum  % row num
    \l__datatool_row_idx_int % row idx
    \l__datatool_col_idx_int % col num
    { #1 } % col idx
}
```

\DTLdisplaydbAddItem

\DTLdisplaydbAddItem⟨*content-tl*⟩{⟨*item*⟩}{⟨*fmt-cs*⟩}{⟨*type*⟩}{⟨*row*⟩}{⟨*row idx*⟩}{⟨*col*⟩}{⟨*col idx*⟩}

Add the column item:

```
\NewDocumentCommand \DTLdisplaydbAddItem { m m m m m m m m }
{
  \tl_put_right:Nn #1 { #3 { #2 } }
}
\cs_new:Nn \__datatool_display_add_item:Nnnnnnnn
{
  \DTLdisplaydbAddItem #1 { #2 } { #3 } { #4 } { #5 } { #6 } { #7 } { #8 }
}
\cs_generate_variant:Nn \__datatool_display_add_item:Nnnnnnnn
 { NVVVVVVn }
```

\dtldisplaylongdbenv

```
\newcommand{\dtldisplaylongdbenv}{longtable}
```

DTLdisplaylongdbAddBegin

\DTLdisplaydblongAddBegin{⟨*content tl*⟩}{⟨*align token list*⟩}{⟨*header token list*⟩}

```
\NewDocumentCommand \DTLdisplaylongdbAddBegin { m m m }
{
  \tl_put_right:Nx #1
   {
     \exp_not:N \begin { \dtldisplaylongdbenv }
```

```
    }
  \tl_put_right:Nn #1
    {
      { #2 }
    }
```
Add the header row. Is there a caption?
```
  \tl_if_eq:NnTF \l_datatool_caption_tl { \c_novalue_tl }
  {
```
No caption.
```
      \bool_if:NT \l_datatool_include_header_bool
      {
        \tl_put_right:NV #1 \dtldisplaystarttab
        \tl_put_right:Nn #1 { #3 }
        \tl_put_right:NV #1 \l_datatool_post_head_tl
        \tl_if_empty:NF \dtldisplayafterhead
        {
          \tl_put_right:NV #1 \dtldisplaycr
          \tl_put_right:NV #1 \dtldisplayafterhead
        }
        \tl_put_right:Nn #1 { \endhead }
      }
  }
  {
```
Caption. Is there a short caption?
```
      \tl_put_right:Nn #1 { \caption }
      \tl_if_eq:NnF \l_datatool_short_caption_tl { \c_novalue_tl }
      {
        \tl_put_right:Nn #1 { [ }
        \tl_put_right:NV #1 \l_datatool_short_caption_tl
        \tl_put_right:Nn #1 { ] }
      }
      \tl_put_right:Nx #1 { { \exp_not:o { \l_datatool_caption_tl } } }
```
Is there a label?
```
      \tl_if_eq:NnF \l_datatool_label_tl { \c_novalue_tl }
      {
        \tl_put_right:Nx #1
         {
           \exp_not:N \label { \l_datatool_label_tl }
         }
      }
      \bool_if:NT \l_datatool_include_header_bool
      {
        \tl_put_right:NV #1 \dtldisplaycr
        \tl_put_right:NV #1 \dtldisplaystarttab
        \tl_put_right:Nn #1 { #3 }
        \tl_put_right:NV #1 \l_datatool_post_head_tl
        \tl_if_empty:NF \dtldisplayafterhead
```

```
      {
        \tl_put_right:NV #1 \dtldisplaycr
        \tl_put_right:NV #1 \dtldisplayafterhead
      }
    }
    \tl_put_right:Nn #1 { \endfirsthead }
```

Caption. Is there a continuation caption?

```
    \tl_put_right:Nn #1 { \caption [] }
    \tl_if_eq:NnTF \l_datatool_cont_caption_tl { \c_novalue_tl }
    {
      \tl_put_right:Nx #1 { { \exp_not:o { \l_datatool_caption_tl } } }
    }
    {
      \tl_put_right:Nx #1 { { \exp_not:o { \l_datatool_cont_caption_tl } } }
    }
    \bool_if:NT \l_datatool_include_header_bool
    {
      \tl_put_right:NV #1 \dtldisplaycr
      \tl_put_right:NV #1 \dtldisplaystarttab
      \tl_put_right:Nn #1 { #3 }
      \tl_put_right:NV #1 \l_datatool_post_head_tl
      \tl_if_empty:NF \dtldisplayafterhead
      {
        \tl_put_right:NV #1 \dtldisplaycr
        \tl_put_right:NV #1 \dtldisplayafterhead
      }
    }
    \tl_put_right:Nn #1 { \endhead }
  }
```

Is there a footer?

```
  \tl_if_eq:NnF \l_datatool_foot_tl { \c_novalue_tl }
  {
    \tl_put_right:NV #1 \l_datatool_foot_tl
    \tl_put_right:Nn #1 { \endfoot }
  }
  \tl_if_eq:NnF \l_datatool_last_foot_tl { \c_novalue_tl }
  {
    \tl_put_right:NV #1 \l_datatool_last_foot_tl
    \tl_put_right:Nn #1 { \endlastfoot }
  }
}
```

\DTLdisplaylongdbAddEnd

\DTLdisplaylongdbAddEnd{⟨*content tl*⟩}

```
\NewDocumentCommand \DTLdisplaylongdbAddEnd { m }
{
```

```
  \tl_put_right:NV #1 \dtldisplayendtab
  \tl_put_right:Nx #1 { \exp_not:N \end { \dtldisplaylongdbenv } }
}
```

\DTLdisplaylongdb

\DTLdisplaylongdb[⟨*options*⟩]{⟨*db*⟩}

Displays the database ⟨*db*⟩ in a longtable environment. (User needs to load longtable).

```
\NewDocumentCommand \DTLdisplaylongdb { o m }
{
  \DTLifdbexists { #2 }
  {
    \group_begin:
    \IfValueT { #1 }
    {
      \keys_set_filter:nnnN { datatool/display } { tabular } { #1 }
        \l__datatool_tmpb_tl
      \tl_if_empty:NF \l__datatool_tmpb_tl
      {
        \PackageWarning { datatool }
        {
          Ignoring ~ unsupported ~ \token_to_str:N\DTLdisplaylongdb
          \c_space_tl option(s): ~ \exp_not:o { \l__datatool_tmpb_tl }
        }
      }
    }
    \tl_set:Nx \dtldbname { #2 }
    \__datatool_display_long_db:
    \group_end:
  }
  {
    \PackageError { datatool }
    { Database ~ `#2' ~ doesn't ~ exist }
    { }
  }
}
```

Inner command:

```
\cs_new:Nn \__datatool_display_long_db:
{
```

Initialise and check supplied options make sense:

```
  \__datatool_if_display_init:T
  {
    \l__datatool_init_display_tl
```

Start constructing the content token list. Add the begin part and header row:

```
    \exp_args:NNVV \DTLdisplaylongdbAddBegin \l__datatool_content_tl
      \l__datatool_align_tl \l__datatool_row_tl
```

The rest is much the same as for \DTLdisplaydb:

```
\int_zero:N \dtlrownum
\int_zero:N \l__datatool_row_idx_int
```

Iterate over all rows:

```
\__datatool_display_loop:
```

Add the end part:

```
\DTLdisplaylongdbAddEnd \l__datatool_content_tl
```

Do the content:

```
    \l__datatool_pre_display_tl
    \l__datatool_content_tl
  }
}
```

12.8 Editing Databases

\@dtl@toksA

```
\newtoks\@dtl@toksA
```

\@dtl@toksB

```
\newtoks\@dtl@toksB
```

\dtlswaprows

\dtlswaprows{⟨*db*⟩}{⟨*row1 idx*⟩}{⟨*row2 idx*⟩}

Swaps the rows with indices ⟨*row1 idx*⟩ and ⟨*row2 idx*⟩ in the database ⟨*db*⟩. (Doesn't check if data base exists or if indices are out of bounds.)

```
\newrobustcmd*{\dtlswaprows}[3]{%
  \ifnum#2=#3\relax
```

Attempt to swap row with itself: do nothing.

```
  \else
```

Let row A be the row with the lower index and row B be the row with ther higher index.

```
    \ifnum#2<#3\relax
      \edef\@dtl@rowAidx{\number#2}%
      \edef\@dtl@rowBidx{\number#3}%
    \else
      \edef\@dtl@rowAidx{\number#3}%
      \edef\@dtl@rowBidx{\number#2}%
    \fi
```

Split the database around row A.

```
    \edef\@dtl@dosplit{\noexpand\dtlgetrow{#1}{\@dtl@rowAidx}}%
    \@dtl@dosplit
```

Store first part of database in \@dtl@firstpart.

```
    \expandafter\def\expandafter\@dtl@firstpart\expandafter
      {\the\dtlbeforerow}%
```

Store row A in `\@dtl@toksA`.

```
\@dtl@toksA=\dtlcurrentrow
```

Split the second part (everything after row A).

```
\edef\@dtl@dosplit{\noexpand\@dtlgetrow
  {\the\dtlafterrow}{\@dtl@rowBidx}}%
\@dtl@dosplit
```

Store the mid part (everything between row A and row B)

```
\expandafter\def\expandafter\@dtl@secondpart\expandafter
  {\the\dtlbeforerow}%
```

Store row B in `\@dtl@toksB`.

```
\@dtl@toksB=\dtlcurrentrow
```

Store the last part (everything after row B).

```
\expandafter\def\expandafter\@dtl@thirdpart\expandafter
  {\the\dtlafterrow}%
```

Reconstruct database: store first part in `\toks@`

```
\toks@=\expandafter{\@dtl@firstpart}%
```

Store mid part in `\dtl@toks`

```
\@dtl@toks=\expandafter{\@dtl@secondpart}%
```

Format data for first part, row B and mid part.

```
\edef\@dtl@tmp{\the\toks@
  \__datatool_row_markup:VV
     \@dtl@rowAidx
     \@dtl@toksB
  \the\@dtl@toks}%
```

Store data so far in `\toks@`.

```
\toks@=\expandafter{\@dtl@tmp}%
```

Store last part in `\dtl@toks`.

```
\@dtl@toks=\expandafter{\@dtl@thirdpart}%
```

Format row A and end part.

```
\edef\@dtl@tmp{\the\toks@
 \__datatool_row_markup:VV
   \@dtl@rowBidx
   \@dtl@toksA
 \the\@dtl@toks
}%
```

Update the database according to global setting.

```
    \__datatool_dtldb_set:nV { #1 } \@dtl@tmp
  \fi
}
```

`\dtl@decrementrows`

`\dtl@decrementrows{`⟨*toks*⟩`}{`⟨*n*⟩`}`

decrement by 1 all rows in ⟨*toks*⟩ with row index above ⟨*n*⟩. NB this doesn't change the database row count.

```
\newcommand*{\dtl@decrementrows}[2]{%
  \def\@dtl@newlist{}%
  \edef\@dtl@min{\number#2}%
  \expandafter\@dtl@decrementrows\the#1%
    \db@row@elt@w%
      \db@row@id@w \@nil\db@row@id@end@%
      \db@row@id@w \@nil\db@row@id@end@%
    \db@row@elt@end@%
    \@nil
  #1=\expandafter{\@dtl@newlist}%
}
```

\@dtl@decrementrows

```
\def\@dtl@decrementrows\db@row@elt@w\db@row@id@w #1\db@row@id@end@%
#2\db@row@id@w #3\db@row@id@end@\db@row@elt@end@#4\@nil{%
  \def\@dtl@thisrow{#1}%
  \ifx\@dtl@thisrow\@nnil
    \let\@dtl@donextdec=\@dtl@gobbletonil
  \else
    \ifnum\@dtl@thisrow>\@dtl@min
      \@dtl@tmpcount=\@dtl@thisrow\relax
      \int_decr:N \@dtl@tmpcount
      \toks@{#2}%
      \@dtl@toks=\expandafter{\@dtl@newlist}%
      \edef\@dtl@newlist{\the\@dtl@toks
        \__datatool_row_markup:VV
          \@dtl@tmpcount
          \toks@
      }%
    \else
      \toks@{#2}%
      \@dtl@toks=\expandafter{\@dtl@newlist}%
      \edef\@dtl@newlist{\the\@dtl@toks
        \__datatool_row_markup:nV
        { #1 }
        \toks@
      }%
    \fi
    \let\@dtl@donextdec=\@dtl@decrementrows
  \fi
  \@dtl@donextdec#4\@nil
}
```

```
\ExplSyntaxOff
```

\DTLremoverow

```
\DTLremoverow{⟨db⟩}{⟨row index⟩}
```

Remove row with given index from database named ⟨*db*⟩.

```
\newcommand*{\DTLremoverow}[2]{%
```

Check database exists

```
  \DTLifdbexists{#1}%
  {%
```

Check index if index is out of bounds

```
    \ifnum#2>0\relax
```

Check if data base has at least ⟨*row index*⟩ rows

```
      \expandafter\ifnum\csname dtlrows@#1\endcsname<#2\relax
        \expandafter\ifnum\csname dtlrows@#1\endcsname=1\relax
          \PackageError{datatool}{Can't remove row `\number#2' from
            database `#1': no such row}{Database `#1' only has
            1 row}%
        \else
          \PackageError{datatool}{Can't remove row `\number#2' from
            database `#1': no such row}{Database `#1' only has
            \expandafter\number\csname dtlrows@#1\endcsname\space
            rows}%
        \fi
      \else
        \@DTLremoverow{#1}{#2}%
      \fi
    \else
      \PackageError{datatool}{Can't remove row \number#2: index
        out of bounds}{Row indices start at 1}%
    \fi
  }%
  {%
    \PackageError{datatool}{Can't remove row: database `#1' doesn't
      exist}{}%
  }%
}

\ExplSyntaxOn
```

\@DTLremoverow

```
\@DTLremoverow{⟨db⟩}{⟨row index⟩}
```

Doesn't perform any checks for the existence of the database or if the index is in range.

```
\newcommand*{\@DTLremoverow}[2]{%
```

Get row from data base

```
      \edef\dtl@dogetrow{\noexpand\dtlgetrow{#1}{\number#2}}%
      \dtl@dogetrow
```

Update the row indices

```
      \expandafter\dtl@decrementrows\expandafter
```

```
      {\dtlbeforerow}{#2}%
    \expandafter\dtl@decrementrows\expandafter
      {\dtlafterrow}{#2}%
```

Reconstruct database according to global setting.

```
\__datatool_dtldb_set:nx { #1 }
 {
   \exp_not:V \dtlbeforerow
   \exp_not:V \dtlafterrow
 }
```

Decrement row counter.

```
\bool_if:NTF \l__datatool_db_global_bool
 {
   \int_gdecr:c { dtlrows@ #1 }
 }
 {
   \int_decr:c { dtlrows@ #1 }
 }
}
```

12.9 Database Functions

\DTLsumforkeys

\DTLsumforkeys[⟨*condition*⟩][⟨*assign list*⟩]{⟨*db list*⟩}{⟨*key list*⟩}{⟨*cmd*⟩}

Sums all entries for key ⟨*key*⟩ over all databases listed in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first argument ⟨*condition*⟩ is the same as that for `\DTLforeach`. The second optional argument provides an assignment list to pass to `\DTLforeach` in case extra information is need by ⟨*condition*⟩. Version 3.0: switched to using `\DTLmapdata`

```
\NewDocumentCommand \DTLsumforkeys { o o m m m }
{
```

Scope to prevent problems if nested.

```
  \group_begin:
  \fp_zero:N \l__datatool_total_fp
```

Initialise.

```
  \int_set_eq:NN
     \@dtl@datatype
     \c_datatool_unknown_int
  \tl_clear:N \l__datatool_datum_currency_tl
  \IfValueTF { #1 }
   {
     \cs_set:Nn \__datatool_filter:n
      {
        \ifthenelse { #1 } { ##1 } { }
      }
```

```
    }
    {
      \cs_set_eq:NN \__datatool_filter:n \use:n
    }
```
Iterate over all the listed data bases:
```
   \clist_map_inline:nn { #3 }
    {
```
Iterate over the current database (read only):
```
      \DTLmapdata [ name = { ##1 }, read-only ]
      {
        \IfValueT { #2 }
         {
           \DTLmapgetvalues { #2 }
         }
        \__datatool_filter:n
         {
```
Iterate through key list.
```
           \clist_map_variable:nNn
            { #4 }
            \l__datatool_item_key_tl
           {
             \DTLmapget
              {
                key = \l__datatool_item_key_tl,
                return = \l__datatool_item_value_tl
              }
             \datatool_if_null:NF \l__datatool_item_value_tl
              {
                \__datatool_parse:N
                  \l__datatool_item_value_tl
```
Check that the value is numerical.
```
                \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
                {
```
Save previous settings.
```
                  \tl_set_eq:NN
                    \l__datatool_tmp_currency_tl
                    \l__datatool_datum_currency_tl
                  \int_set_eq:NN
                    \l__datatool_tmp_datatype_int
                    \@dtl@datatype
```
Obtain this item’s numeric value as a floating point variable.
```
                  \datatool_set_fp:NV \l__datatool_datum_value_fp
                    \l__datatool_item_value_tl
                  \fp_add:Nn \l__datatool_total_fp
                    { \l__datatool_datum_value_fp }
```

Update data type, if applicable.

```
                    \__datatool_update_datatype:
                  }
                }
              }
            }
          }
        }
      }
```

Convert floating point variable and format result.

```
   \tl_set:Ne \l__datatool_result_tl
    { \fp_to_decimal:N \l__datatool_total_fp }
   \__datatool_assign_result:N #5
```

Expand to the formatted value if store-datum = false.

```
   \bool_if:NF \l__datatool_db_store_datum_bool
    { \tl_set:Nx #5 { #5 } }
```

End scope.

```
   \exp_args:NNNV
    \group_end: \tl_set:Nn #5 #5
 }
```

`\@dtlsumforkeys` Version 3.0: removed `\@dtlsumforkeys`.

`\DTLsumcolumn`

`\DTLsumcolumn{`⟨*db*⟩`}{`⟨*key*⟩`}{`⟨*cmd*⟩`}`

Quicker version of `\DTLsumforkeys` that just sums over one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩. Version 3.0: switched to using `\DTLmapdata`

```
 \NewDocumentCommand \DTLsumcolumn { m m m }
 {
```

Scope to prevent problems if nested.

```
   \group_begin:
   \fp_zero:N \l__datatool_total_fp
```

Initialise.

```
   \int_set_eq:NN
      \@dtl@datatype
      \c_datatool_unknown_int
   \tl_clear:N \l__datatool_datum_currency_tl
```

Iterate over the database (read only):

```
   \DTLmapdata [ name = { #1 }, read-only ]
    {
      \DTLmapget
       {
         key = { #2 },
         return = \l__datatool_item_value_tl
```

```
      }
    \datatool_if_null:NF \l__datatool_item_value_tl
     {
       \__datatool_parse:N
         \l__datatool_item_value_tl
```

Check that the value is numerical.

```
       \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
       {
```

Save previous settings.

```
         \tl_set_eq:NN
           \l__datatool_tmp_currency_tl
           \l__datatool_datum_currency_tl
         \int_set_eq:NN
           \l__datatool_tmp_datatype_int
           \@dtl@datatype
```

Obtain this item's numeric value as a floating point variable.

```
         \datatool_set_fp:NV \l__datatool_datum_value_fp
           \l__datatool_item_value_tl
         \fp_add:Nn \l__datatool_total_fp
           { \l__datatool_datum_value_fp }
```

Update data type, if applicable.

```
         \__datatool_update_datatype:
       }
     }
  }
```

Convert floating point variable and format result.

```
 \tl_set:Ne \l__datatool_result_tl
  { \fp_to_decimal:N \l__datatool_total_fp }
 \__datatool_assign_result:N #3
```

Expand to the formatted value if store-datum = false.

```
 \bool_if:NF \l__datatool_db_store_datum_bool
  { \tl_set:Nx #3 { #3 } }
```

End scope.

```
 \exp_args:NNNV
  \group_end: \tl_set:Nn #3 #3
}
```

`\@dtl@elements` Count register to keep track of number of elements. TODO remove.

```
\newcount\@dtl@elements
```

`\DTLmeanforkeys`

```
\DTLmeanforkeys[⟨condition⟩][⟨assign list⟩]{⟨db list⟩}
{⟨key list⟩}{⟨cmd⟩}
```

Computes the arithmetic mean of all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first argument ⟨*condition*⟩ is the same as that for `\DTLforeach`. The second optional argument allows an assignment list to be passed to `\DTLmapgetvalues`. Version 3.0: now uses `\DTLmapdata`.

```
\NewDocumentCommand \DTLmeanforkeys { o o m m m }
{%
```

Scope to prevent problems if nested.

```
  \group_begin:
  \int_zero:N \l__datatool_count_int
  \fp_zero:N \l__datatool_total_fp
```

Initialise.

```
  \int_set_eq:NN
     \@dtl@datatype
     \c_datatool_unknown_int
  \tl_clear:N \l__datatool_datum_currency_tl
  \seq_clear:N \l__datatool_tmpb_seq
  \IfValueTF { #1 }
   {
     \cs_set:Nn \__datatool_filter:n
      {
        \ifthenelse { #1 } { ##1 } { }
      }
   }
   {
     \cs_set_eq:NN \__datatool_filter:n \use:n
   }
   \__datatool_filtered_calc_mean:nnn { #2 } { #3 } { #4 }
```

Convert floating point variable and format result.

```
  \tl_set:Ne \l__datatool_result_tl
   { \fp_to_decimal:N \l__datatool_mean_fp }
  \__datatool_assign_result:N #5
```

Expand to the formatted value if store-datum = false.

```
  \bool_if:NF \l__datatool_db_store_datum_bool
   { \tl_set:Nx #5 { #5 } }
```

End scope.

```
  \exp_args:NNNV
   \group_end: \tl_set:Nn #5 #5
}

\cs_new:Nn \__datatool_filtered_calc_mean:nnn
{
```

Iterate over all the listed data bases:

```
  \clist_map_inline:nn { #2 }
   {
```

Iterate over the current database (read only):

```
\DTLmapdata [ name = { ##1 }, read-only ]
{
  \IfValueT { #1 }
   {
     \DTLmapgetvalues { #1 }
   }
  \__datatool_filter:n
   {
```

Iterate through key list.

```
     \clist_map_variable:nNn
      { #3 }
      \l__datatool_item_key_tl
     {
       \DTLmapget
        {
          key = \l__datatool_item_key_tl,
          return = \l__datatool_item_value_tl
        }
       \datatool_if_null:NF \l__datatool_item_value_tl
        {
          \__datatool_parse:N
            \l__datatool_item_value_tl
```

Check that the value is numerical.

```
          \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
          {
```

Save previous settings.

```
            \tl_set_eq:NN
              \l__datatool_tmp_currency_tl
              \l__datatool_datum_currency_tl
            \int_set_eq:NN
              \l__datatool_tmp_datatype_int
              \@dtl@datatype
```

Increment item count.

```
            \int_incr:N \l__datatool_count_int
```

Obtain this item's numeric value as a floating point variable.

```
            \datatool_set_fp:NV \l__datatool_datum_value_fp
              \l__datatool_item_value_tl
            \fp_add:Nn \l__datatool_total_fp
              { \l__datatool_datum_value_fp }
            \seq_put_right:Nx \l__datatool_tmpb_seq
              { \DTLdatumvalue { \l__datatool_item_value_tl } }
```

Update data type, if applicable.

```
            \__datatool_update_datatype:
          }
        }
```

```
            }
          }
        }
      }
    \int_if_zero:nTF { \l__datatool_count_int }
    {
      \fp_zero:N \l__datatool_mean_fp
      \PackageError
      { datatool }
      {
        no ~ numeric ~ data ~ found ~ in ~ column ~ key ~ set ~ `#3' ~
        for ~ database ~ set ~ `#2'
      }
      { }
    }
    {
      \fp_set:Nn
        \l__datatool_mean_fp
        { \l__datatool_total_fp / \l__datatool_count_int }
    }
  }
```

`\@dtlmeanforkeys` Version 3.0: removed `\@dtlmeanforkeys`.

`\DTLmeanforcolumn`

`\DTLmeanforcolumn{`⟨*db*⟩`}{`⟨*key*⟩`}{`⟨*cmd*⟩`}`

Quicker version of `\DTLmeanforkeys` that just computes the mean over one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩. Version 3.0: switched to `\DTLmapdata`.

```
\NewDocumentCommand \DTLmeanforcolumn { m m m }
{
```

Check data base exists

```
  \DTLifdbexists { #1 }
  {
```

Scope to prevent problems if nested.

```
    \group_begin:
    \int_zero:N \l__datatool_count_int
    \fp_zero:N \l__datatool_total_fp
```

Initialise.

```
    \int_set_eq:NN
       \@dtl@datatype
       \c_datatool_unknown_int
    \tl_clear:N \l__datatool_datum_currency_tl
    \seq_clear:N \l__datatool_tmpb_seq
    \__datatool_column_calc_mean:nn { #1 } { #2 }
```

Convert floating point variable and format result.

```
      \tl_set:Ne \l__datatool_result_tl
       { \fp_to_decimal:N \l__datatool_mean_fp }
      \__datatool_assign_result:N #3
```

Expand to the formatted value if store-datum = false.

```
      \bool_if:NF \l__datatool_db_store_datum_bool
       { \tl_set:Nx #3 { #3 } }
```

End scope.

```
      \exp_args:NNNV
       \group_end: \tl_set:Nn #3 #3
   }
   {
     \PackageError { datatool }
     {
       Database ~ `#1' ~ isn't ~ defined
     }
     { }
   }
}

\cs_new:Nn \__datatool_column_calc_mean:nn
{
  \DTLmapdata [ name = { #1 }, read-only ]
  {
    \DTLmapget
     {
       key = { #2 },
       return = \l__datatool_item_value_tl
     }
    \datatool_if_null:NF \l__datatool_item_value_tl
     {
       \__datatool_parse:N
         \l__datatool_item_value_tl
```

Check that the value is numerical.

```
        \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
        {
```

Save previous settings.

```
          \tl_set_eq:NN
            \l__datatool_tmp_currency_tl
            \l__datatool_datum_currency_tl
          \int_set_eq:NN
            \l__datatool_tmp_datatype_int
            \@dtl@datatype
```

Increment item count.

```
          \int_incr:N \l__datatool_count_int
```

Obtain this item's numeric value as a floating point variable.

```
          \datatool_set_fp:NV \l__datatool_datum_value_fp
            \l__datatool_item_value_tl
          \fp_add:Nn \l__datatool_total_fp
            { \l__datatool_datum_value_fp }
          \seq_put_right:Nx \l__datatool_tmpb_seq
            { \DTLdatumvalue { \l__datatool_item_value_tl } }
```

Update data type, if applicable.

```
          \__datatool_update_datatype:
        }
      }
    }
    \int_if_zero:nTF { \l__datatool_count_int }
    {
      \PackageError
      { datatool }
      {
        no ~ numeric ~ data ~ found ~ in ~ column ~ labelled ~ `#2' ~
        in ~ database ~ `#1'
      }
      { }
      \fp_zero:N \l__datatool_mean_fp
    }
    {
      \fp_set:Nn
        \l__datatool_mean_fp
        { \l__datatool_total_fp / \l__datatool_count_int }
    }
  }
```

\DTLvarianceforkeys

\DTLvarianceforkeys[⟨*condition*⟩][⟨*assign list*⟩]{⟨*db list*⟩}{⟨*key list*⟩}{⟨*cmd*⟩}

Computes the variance of all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first optional argument ⟨*condition*⟩ is the same as that for `\DTLforeach`. The second optional argument is an assignment list to pass to `\DTLmapgetvalues` in case it is required for the condition. Version 3.0: changed to use `\DTLmapdata`

```
\NewDocumentCommand \DTLvarianceforkeys
  { o o m m m }
{
```

Scope to prevent problems if nested.

```
  \group_begin:
  \int_zero:N \l__datatool_count_int
  \fp_zero:N \l__datatool_total_fp
```

Initialise.

```
   \int_set_eq:NN
      \@dtl@datatype
      \c_datatool_unknown_int
   \tl_clear:N \l__datatool_datum_currency_tl
   \seq_clear:N \l__datatool_tmpb_seq
   \IfValueTF { #1 }
    {
      \cs_set:Nn \__datatool_filter:n
       {
         \ifthenelse { #1 } { ##1 } { }
       }
    }
    {
      \cs_set_eq:NN \__datatool_filter:n \use:n
    }
```

The mean needs to be calculated first.

```
    \__datatool_filtered_calc_mean:nnn { #2 } { #3 } { #4 }
   \int_if_zero:nF { \l__datatool_count_int }
   {
```

Calculate the variance.

```
     \fp_zero:N \l__datatool_tmpa_fp
     \seq_map_inline:Nn \l__datatool_tmpb_seq
      {
        \fp_set:Nn \l__datatool_tmpb_fp
         {
           ##1 - \l__datatool_mean_fp
         }
        \fp_add:Nn \l__datatool_tmpa_fp
         {
            \l__datatool_tmpb_fp * \l__datatool_tmpb_fp
         }
      }
     \fp_set:Nn \l__datatool_tmpa_fp
        { \l__datatool_tmpa_fp / \l__datatool_count_int }
```

Convert floating point variable and format result.

```
     \tl_set:Ne \l__datatool_result_tl
      { \fp_to_decimal:N \l__datatool_tmpa_fp }
     \__datatool_assign_result:N #5
```

Expand to the formatted value if store-datum = false.

```
     \bool_if:NF \l__datatool_db_store_datum_bool
      { \tl_set:Nx #5 { #5 } }
   }
```

End scope.

```
   \exp_args:NNNV
    \group_end: \tl_set:Nn #5 #5
 }
```

\@dtlvarianceforkeys Version 3.0: removed.

\DTLvarianceforcolumn

\DTLvarianceforcolumn{⟨*db*⟩}{⟨*key*⟩}{⟨*cmd*⟩}

Quicker version of \DTLvarianceforkeys that just computes the variance over one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩. Version 3.0: switched to \DTLmapdata.

```
\NewDocumentCommand \DTLvarianceforcolumn { m m m }
{
```

Check data base exists

```
   \DTLifdbexists { #1 }
   {
```

Scope to prevent problems if nested.

```
      \group_begin:
      \int_zero:N \l__datatool_count_int
      \fp_zero:N \l__datatool_total_fp
```

Initialise.

```
      \int_set_eq:NN
         \@dtl@datatype
         \c_datatool_unknown_int
      \tl_clear:N \l__datatool_datum_currency_tl
```

The mean needs to be calculated first.

```
      \__datatool_column_calc_mean:nn { #1 } { #2 }
      \int_if_zero:nF { \l__datatool_count_int }
      {
```

Calculate the variance.

```
        \fp_zero:N \l__datatool_tmpa_fp
        \seq_map_inline:Nn \l__datatool_tmpb_seq
         {
           \fp_set:Nn \l__datatool_tmpb_fp
            {
              ##1 - \l__datatool_mean_fp
            }
           \fp_add:Nn \l__datatool_tmpa_fp
            {
               \l__datatool_tmpb_fp * \l__datatool_tmpb_fp
            }
         }
        \fp_set:Nn \l__datatool_tmpa_fp
         { \l__datatool_tmpa_fp / \l__datatool_count_int }
```

Convert floating point variable and format result.

```
        \tl_set:Ne \l__datatool_result_tl
         { \fp_to_decimal:N \l__datatool_tmpa_fp }
        \__datatool_assign_result:N #3
```

Expand to the formatted value if store-datum = false.

```
      \bool_if:NF \l__datatool_db_store_datum_bool
       { \tl_set:Nx #3 { #3 } }
    }
```

End scope.

```
    \exp_args:NNNV
     \group_end: \tl_set:Nn #3 #3
  }
  {
    \PackageError { datatool }
    {
      Database ~ `#1' ~ isn't ~ defined
    }
    { }
  }
}
```

\DTLsdforkeys

\DTLsdforkeys[⟨*condition*⟩][⟨*assign list*⟩]{⟨*db list*⟩}{⟨*key list*⟩}{⟨*cmd*⟩}

Computes the standard deviation of all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first optional argument ⟨*condition*⟩ is the same as that for `\DTLmapgetvalues`. The second optional argument is an assignment list for `\DTLforeach` in case it is needed for the condition. Version 3.0: now uses `\DTLmapdata`.

```
\NewDocumentCommand \DTLsdforkeys
  { o o m m m }
{
```

Scope to prevent problems if nested.

```
  \group_begin:
  \int_zero:N \l__datatool_count_int
  \fp_zero:N \l__datatool_total_fp
```

Initialise.

```
  \int_set_eq:NN
     \@dtl@datatype
     \c_datatool_unknown_int
  \tl_clear:N \l__datatool_datum_currency_tl
  \seq_clear:N \l__datatool_tmpb_seq
  \IfValueTF { #1 }
   {
     \cs_set:Nn \__datatool_filter:n
      {
        \ifthenelse { #1 } { ##1 } { }
      }
   }
   {
```

```
      \cs_set_eq:NN \__datatool_filter:n \use:n
    }
```

The mean needs to be calculated first.

```
    \__datatool_filtered_calc_mean:nnn { #2 } { #3 } { #4 }
   \int_if_zero:nF { \l__datatool_count_int }
   {
```

Calculate the variance.

```
     \fp_zero:N \l__datatool_tmpa_fp
     \seq_map_inline:Nn \l__datatool_tmpb_seq
      {
        \fp_set:Nn \l__datatool_tmpb_fp
         {
           ##1 - \l__datatool_mean_fp
         }
        \fp_add:Nn \l__datatool_tmpa_fp
         {
            \l__datatool_tmpb_fp * \l__datatool_tmpb_fp
         }
      }
     \fp_set:Nn \l__datatool_tmpa_fp
        { sqrt ( \l__datatool_tmpa_fp / \l__datatool_count_int ) }
```

Convert floating point variable and format result.

```
     \tl_set:Ne \l__datatool_result_tl
      { \fp_to_decimal:N \l__datatool_tmpa_fp }
     \__datatool_assign_result:N #5
```

Expand to the formatted value if store-datum = false.

```
     \bool_if:NF \l__datatool_db_store_datum_bool
      { \tl_set:Nx #5 { #5 } }
   }
```

End scope.

```
   \exp_args:NNNV
    \group_end: \tl_set:Nn #5 #5
 }
```

`\@dtlsdforkeys` Version 3.0: removed.

`\DTLsdforcolumn`

`\DTLsdforcolumn{`⟨*db*⟩`}{`⟨*key*⟩`}{`⟨*cmd*⟩`}`

Quicker version of `\DTLsdforkeys` that just computes the standard deviation over one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩. Version 3.0: now uses `\DTLmapdata`.

```
 \NewDocumentCommand \DTLsdforcolumn { m m m }
 {
```

Check data base exists

```
\DTLifdbexists { #1 }
{
```

Scope to prevent problems if nested.

```
  \group_begin:
  \int_zero:N \l__datatool_count_int
  \fp_zero:N \l__datatool_total_fp
```

Initialise.

```
  \int_set_eq:NN
     \@dtl@datatype
     \c_datatool_unknown_int
  \tl_clear:N \l__datatool_datum_currency_tl
```

The mean needs to be calculated first.

```
  \__datatool_column_calc_mean:nn { #1 } { #2 }
  \int_if_zero:nF { \l__datatool_count_int }
  {
```

Calculate the variance.

```
    \fp_zero:N \l__datatool_tmpa_fp
    \seq_map_inline:Nn \l__datatool_tmpb_seq
     {
       \fp_set:Nn \l__datatool_tmpb_fp
        {
          ##1 - \l__datatool_mean_fp
        }
       \fp_add:Nn \l__datatool_tmpa_fp
        {
           \l__datatool_tmpb_fp * \l__datatool_tmpb_fp
        }
     }
    \fp_set:Nn \l__datatool_tmpa_fp
     { sqrt ( \l__datatool_tmpa_fp / \l__datatool_count_int ) }
```

Convert floating point variable and format result.

```
    \tl_set:Ne \l__datatool_result_tl
     { \fp_to_decimal:N \l__datatool_tmpa_fp }
    \__datatool_assign_result:N #3
```

Expand to the formatted value if store-datum = false.

```
    \bool_if:NF \l__datatool_db_store_datum_bool
     { \tl_set:Nx #3 { #3 } }
  }
```

End scope.

```
  \exp_args:NNNV
   \group_end: \tl_set:Nn #3 #3
}
{
  \PackageError { datatool }
```

```
      {
        Database ~ `#1' ~ isn't ~ defined
      }
      { }
    }
  }
```

\DTLminforkeys

\DTLminforkeys[⟨*condition*⟩][⟨*assign list*⟩]{⟨*db list*⟩}{⟨*key list*⟩}{⟨*cmd*⟩}

Determines the minimum over all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first optional argument ⟨*condition*⟩ is the same as that for `\DTLforeach`. The second optional argument is an assignment list for `\DTLforeach` in the event that extra information is need for the condition. Version 3.0: changed to use `\DTLmapdata`

```
\NewDocumentCommand \DTLminforkeys { o o m m m }
{
```

Scope to prevent problems if nested.

```
  \group_begin:
  \fp_set_eq:NN \l__datatool_min_fp \c_inf_fp
  \tl_clear:N #5
  \IfValueTF { #1 }
   {
     \cs_set:Nn \__datatool_filter:n
      {
        \ifthenelse { #1 } { ##1 } { }
      }
   }
   {
     \cs_set_eq:NN \__datatool_filter:n \use:n
   }
```

Iterate over all the listed data bases:

```
  \clist_map_inline:nn { #3 }
   {
```

Iterate over the current database (read only):

```
      \DTLmapdata [ name = { ##1 }, read-only ]
      {
        \IfValueT { #2 }
         {
           \DTLmapgetvalues { #2 }
         }
        \__datatool_filter:n
         {
```

Iterate through key list.

```
           \clist_map_variable:nNn
```

```
          { #4 }
          \l__datatool_item_key_tl
         {
           \DTLmapget
            {
              key = \l__datatool_item_key_tl,
              return = \l__datatool_item_value_tl
            }
           \datatool_if_null:NF \l__datatool_item_value_tl
            {
              \__datatool_parse:N
                \l__datatool_item_value_tl
```

Check that the value is numerical.

```
                \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
                {
                  \datatool_set_fp:NV \l__datatool_datum_value_fp
                    \l__datatool_item_value_tl
                  \fp_compare:nNnT
                    { \l__datatool_min_fp }
                     >
                    { \l__datatool_datum_value_fp }
                  {
                    \fp_set_eq:NN
                      \l__datatool_min_fp
                      \l__datatool_datum_value_fp
                    \tl_set_eq:NN #5 \l__datatool_item_value_tl
                  }
                }
            }
        }
      }
    }
  \tl_if_empty:NTF #5
   {
     \PackageError { datatool }
     {
       no ~ numeric ~ data ~ found ~ in ~ column ~ key ~ set ~ `#3' ~
      for ~ database ~ set ~ `#2'
     }
     { }
   }
   {
     \bool_if:NF \l__datatool_db_store_datum_bool
      {
        \tl_set:Nx #5 { #5 }
      }
   }
```

End scope.

```
    \exp_args:NNNV
     \group_end: \tl_set:Nn #5 #5
  }
```

`\@dtlminforkeys` Version 3.0: removed.

`\DTLminforcolumn`

`\DTLminforcolumn{`*⟨db⟩*`}{`*⟨key⟩*`}{`*⟨cmd⟩*`}`

Quicker version of `\DTLminforkeys` that just finds the minimum value in one column (specified by *⟨key⟩*) for a single database (specified by *⟨db⟩*) and stores the result in *⟨cmd⟩*. Version 3.0: switched to using `\DTLmapdata`

```
\NewDocumentCommand \DTLminforcolumn { m m m }
{
```

Check data base exists

```
  \DTLifdbexists{#1}%
  {
```

Scope to prevent problems if nested.

```
  \group_begin:
  \fp_set_eq:NN \l__datatool_min_fp \c_inf_fp
  \tl_clear:N #3
```

Iterate over the database (read only):

```
  \DTLmapdata [ name = { #1 }, read-only ]
   {
     \DTLmapget
      {
        key = { #2 },
        return = \l__datatool_item_value_tl
      }
     \datatool_if_null:NF \l__datatool_item_value_tl
      {
        \__datatool_parse:N
          \l__datatool_item_value_tl
```

Check that the value is numerical.

```
        \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
        {
          \datatool_set_fp:NV \l__datatool_datum_value_fp
            \l__datatool_item_value_tl
          \fp_compare:nNnT
            { \l__datatool_min_fp }
             >
            { \l__datatool_datum_value_fp }
          {
            \fp_set_eq:NN
              \l__datatool_min_fp
```

```
                \l__datatool_datum_value_fp
              \tl_set_eq:NN #3 \l__datatool_item_value_tl
            }
          }
        }
      }
    \tl_if_empty:NTF #3
      {
        \PackageError { datatool }
        {
          no ~ numeric ~ data ~ found ~ in ~ column ~ `#2' ~
         for ~ database ~ `#1'
        }
        { }
      }
      {
        \bool_if:NF \l__datatool_db_store_datum_bool
         {
           \tl_set:Nx #3 { #3 }
         }
      }
```

End scope.

```
    \exp_args:NNNV
     \group_end: \tl_set:Nn #3 #3
    }
```

data base doesn't exist

```
    {
      \PackageError {datatool}
      { Database ~ `#1' ~ doesn't ~ exist}
      {}
    }
  }
```

\DTLmaxforkeys

\DTLmaxforkeys[⟨*condition*⟩][⟨*assign list*⟩]{⟨*db list*⟩}{⟨*key list*⟩}{⟨*cmd*⟩}

Determines the maximum over all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first optional argument ⟨*condition*⟩ is the same as that for \DTLforeach. The second optional argument is an assignment list to pass to \DTLforeach in the event that extra information is required in the condition. Version 3.0: changed to use \DTLmapdata

```
 \NewDocumentCommand \DTLmaxforkeys { o o m m m }
 {
```

Scope to prevent problems if nested.

```
    \group_begin:
    \fp_set_eq:NN \l__datatool_max_fp \c_minus_inf_fp
```

```
\tl_clear:N #5
\IfValueTF { #1 }
 {
   \cs_set:Nn \__datatool_filter:n
    {
      \ifthenelse { #1 } { ##1 } { }
    }
 }
 {
   \cs_set_eq:NN \__datatool_filter:n \use:n
 }
```

Iterate over all the listed data bases:

```
\clist_map_inline:nn { #3 }
 {
```

Iterate over the current database (read only):

```
    \DTLmapdata [ name = { ##1 }, read-only ]
    {
      \IfValueT { #2 }
       {
         \DTLmapgetvalues { #2 }
       }
      \__datatool_filter:n
       {
```

Iterate through key list.

```
          \clist_map_variable:nNn
           { #4 }
           \l__datatool_item_key_tl
          {
            \DTLmapget
             {
               key = \l__datatool_item_key_tl,
               return = \l__datatool_item_value_tl
             }
            \datatool_if_null:NF \l__datatool_item_value_tl
             {
               \__datatool_parse:N
                 \l__datatool_item_value_tl
```

Check that the value is numerical.

```
               \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
               {
                 \datatool_set_fp:NV \l__datatool_datum_value_fp
                   \l__datatool_item_value_tl
                 \fp_compare:nNnT
                   { \l__datatool_max_fp }
                    <
                   { \l__datatool_datum_value_fp }
                 {
```

```
                    \fp_set_eq:NN
                      \l__datatool_max_fp
                      \l__datatool_datum_value_fp
                    \tl_set_eq:NN #5 \l__datatool_item_value_tl
                  }
                }
              }
            }
          }
        }
      }
    }
   \tl_if_empty:NTF #5
    {
      \PackageError { datatool }
      {
        no ~ numeric ~ data ~ found ~ in ~ column ~ key ~ set ~ `#3' ~
       for ~ database ~ set ~ `#2'
      }
      { }
    }
    {
      \bool_if:NF \l__datatool_db_store_datum_bool
       {
         \tl_set:Nx #5 { #5 }
       }
    }
```

End scope.

```
   \exp_args:NNNV
    \group_end: \tl_set:Nn #5 #5
}
```

`\@dtlmaxforkeys` Version 3.0: removed.

`\DTLmaxforcolumn`

`\DTLmaxforcolumn{`⟨*db*⟩`}{`⟨*key*⟩`}{`⟨*cmd*⟩`}`

Quicker version of `\DTLmaxforkeys` that just finds the maximum value in one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩. Version 3.0: switched to using `\DTLmapdata`

```
\NewDocumentCommand \DTLmaxforcolumn { m m m }
{
```

Check data base exists

```
   \DTLifdbexists{#1}%
   {
```

Scope to prevent problems if nested.

```
   \group_begin:
   \fp_set_eq:NN \l__datatool_max_fp \c_minus_inf_fp
   \tl_clear:N #3
```

Iterate over the database (read only):

```
\DTLmapdata [ name = { #1 }, read-only ]
 {
   \DTLmapget
    {
      key = { #2 },
      return = \l__datatool_item_value_tl
    }
   \datatool_if_null:NF \l__datatool_item_value_tl
    {
      \__datatool_parse:N
        \l__datatool_item_value_tl
```

Check that the value is numerical.

```
        \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
        {
          \datatool_set_fp:NV \l__datatool_datum_value_fp
            \l__datatool_item_value_tl
          \fp_compare:nNnT
            { \l__datatool_max_fp }
             <
            { \l__datatool_datum_value_fp }
          {
            \fp_set_eq:NN
              \l__datatool_max_fp
              \l__datatool_datum_value_fp
            \tl_set_eq:NN #3 \l__datatool_item_value_tl
          }
        }
    }
 }
\tl_if_empty:NTF #3
 {
   \PackageError { datatool }
   {
     no ~ numeric ~ data ~ found ~ in ~ column ~ `#2' ~
    for ~ database ~ `#1'
   }
   { }
 }
 {
   \bool_if:NF \l__datatool_db_store_datum_bool
    {
      \tl_set:Nx #3 { #3 }
    }
 }
```

End scope.

```
\exp_args:NNNV
 \group_end: \tl_set:Nn #3 #3
```

```
    }
```
data base doesn't exist
```
    {
      \PackageError {datatool}
      { Database ~ `#1' ~ doesn't ~ exist}
      {}
    }
  }
```

\DTLcomputebounds[⟨*condition*⟩]{⟨*db list*⟩}{⟨*x key*⟩}{⟨*y key*⟩}{⟨*minX cmd*⟩}{⟨*minY cmd*⟩}{⟨*maxX cmd*⟩}{⟨*maxY cmd*⟩}

\DTLcomputebounds

Computes the maximum and minimum x and y values over all the databases listed in ⟨*db list*⟩ where the x value is given by ⟨*x key*⟩ and the y value is given by ⟨*y key*⟩. The results are stored in ⟨*minX cmd*⟩, ⟨*minY cmd*⟩, ⟨*maxX cmd*⟩ and ⟨*maxY cmd*⟩ in plain decimal format. Version 3.0: switched to using `\DTLmapdata`

```
\NewDocumentCommand \DTLcomputebounds { o m m m m m m m }
{
```
Scope to prevent problems if nested.
```
  \group_begin:
  \fp_set_eq:NN \l__datatool_min_fp \c_inf_fp
  \fp_set_eq:NN \l__datatool_min_ii_fp \c_inf_fp
  \fp_set_eq:NN \l__datatool_max_fp \c_minus_inf_fp
  \fp_set_eq:NN \l__datatool_max_ii_fp \c_minus_inf_fp
  \tl_clear:N #5
  \tl_clear:N #6
  \tl_clear:N #7
  \tl_clear:N #8
  \IfValueTF { #1 }
   {
     \cs_set:Nn \__datatool_filter:n
      {
        \ifthenelse { #1 } { ##1 } { }
      }
   }
   {
     \cs_set_eq:NN \__datatool_filter:n \use:n
   }
```
Iterate over all the listed data bases:
```
  \clist_map_inline:nn { #2 }
   {
```
Iterate over the current database (read only):
```
      \DTLmapdata [ name = { ##1 }, read-only ]
      {
        \DTLmapgetvalues
```

```
    {
      \DTLthisX = #3, \DTLthisY = #4
    }
  \__datatool_filter:n
    {
```

x key:

```
      \tl_set_eq:NN
        \l__datatool_item_value_tl \DTLthisX
      \datatool_if_null:NF \l__datatool_item_value_tl
       {
         \__datatool_parse:N
           \l__datatool_item_value_tl
```

Check that the value is numerical.

```
         \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
         {
           \datatool_set_fp:NV \l__datatool_datum_value_fp
             \l__datatool_item_value_tl
           \fp_compare:nNnT
             { \l__datatool_min_fp }
              >
             { \l__datatool_datum_value_fp }
           {
             \fp_set_eq:NN
               \l__datatool_min_fp
               \l__datatool_datum_value_fp
             \tl_set_eq:Nx #5
             { \DTLdatumvalue { \l__datatool_item_value_tl } }
           }
           \fp_compare:nNnT
             { \l__datatool_max_fp }
              <
             { \l__datatool_datum_value_fp }
           {
             \fp_set_eq:NN
               \l__datatool_max_fp
               \l__datatool_datum_value_fp
             \tl_set_eq:Nx #7
             { \DTLdatumvalue { \l__datatool_item_value_tl } }
           }
         }
       }
```

y key:

```
      \tl_set_eq:NN
        \l__datatool_item_value_tl \DTLthisY
      \datatool_if_null:NF \l__datatool_item_value_tl
       {
         \__datatool_parse:N
           \l__datatool_item_value_tl
```

Check that the value is numerical.

```
                \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
                {
                  \datatool_set_fp:NV \l__datatool_datum_value_fp
                    \l__datatool_item_value_tl
                  \fp_compare:nNnT
                    { \l__datatool_min_ii_fp }
                     >
                    { \l__datatool_datum_value_fp }
                  {
                    \fp_set_eq:NN
                      \l__datatool_min_ii_fp
                      \l__datatool_datum_value_fp
                    \tl_set_eq:Nx #6
                    { \DTLdatumvalue { \l__datatool_item_value_tl } }
                  }
                  \fp_compare:nNnT
                    { \l__datatool_max_ii_fp }
                     <
                    { \l__datatool_datum_value_fp }
                  {
                    \fp_set_eq:NN
                      \l__datatool_max_ii_fp
                      \l__datatool_datum_value_fp
                    \tl_set_eq:Nx #8
                    { \DTLdatumvalue { \l__datatool_item_value_tl } }
                  }
                }
              }
            }
          }
        }
      \tl_if_empty:NTF #5
       {
         \PackageError { datatool }
         {
           no ~ numeric ~ data ~ found ~ in ~ column ~ `#3' ~
          for ~ database ~ set ~ `#2'
         }
         { }
       }
```

End scope.

```
      \tl_set:Nx \l__datatool_tmpa_tl
        {
          \exp_not:N \group_end:
          \exp_not:N \tl_set:Nn \exp_not:N #5 { #5 }
          \exp_not:N \tl_set:Nn \exp_not:N #6 { #6 }
          \exp_not:N \tl_set:Nn \exp_not:N #7 { #7 }
          \exp_not:N \tl_set:Nn \exp_not:N #8 { #8 }
```

```
    }
  \l__datatool_tmpa_tl
}
```

Map over columns in the current row.

\dtlmapcurrentrow

`\dtlmapcurrentrow{⟨cs⟩}{⟨body⟩}`

```
\NewDocumentCommand \dtlmapcurrentrow { m m }
{
  \tl_if_empty:VTF \dtlcurrentrow
  {
    \PackageError { datatool }
    {
      Unable ~ to ~ map ~ current ~ row ~ (missing or empty)
    }
    {}
  }
  {
    \__datatool_map_current_row:Nn #1 { #2 }
  }
}
\cs_new:Nn \__datatool_map_current_row:Nn
{
  \cs_set:Nn \__datatool_map_row_loop_body:n
   {
     \tl_set:Nn #1 { ##1 }
     \exp_args:NV \tl_if_head_eq_meaning:nNT
        #1 \__datatool_datum:w
      {
        \__datatool_to_datum:N #1
      }
     #2
   }
  \exp_last_unbraced:NV \__datatool_map_row:w
    \dtlcurrentrow
    \db@col@id@w \q_recursion_tail \db@col@id@end@
    \db@col@elt@w \db@col@elt@end@
    \db@col@id@w \q_recursion_tail \db@col@id@end@
  \q_recursion_stop
  \prg_break_point:Nn \__datatool_map_row_break: { }
}
\ExplSyntaxOff
```

\DTLgetvalueforkey

`\DTLgetvalueforkey{⟨cmd⟩}{⟨key⟩}{⟨db name⟩}{⟨ref key⟩}{⟨ref value⟩}`

This (globally) sets ⟨*cmd*⟩ (a control sequence) to the value of the key specified by ⟨*key*⟩ in the first row of the database called ⟨*db name*⟩ which contains the key ⟨*ref key*⟩ which has the value ⟨*value*⟩. TODO

```
\newcommand*{\DTLgetvalueforkey}[5]{%
```

Get row containing referenced (key,value) pair

```
  \DTLgetrowforkey{\@dtl@row}{#3}{#4}{#5}%
```

Get column number for ⟨*key*⟩

```
  \@sdtl@getcolumnindex{\@dtl@col}{#3}{#2}%
```

Get value for given column

```
  {%
    \dtlcurrentrow=\expandafter{\@dtl@row}%
    \edef\@dtl@dogetval{\noexpand\dtlgetentryfromcurrentrow
      {\noexpand\@dtl@val}{\@dtl@col}}%
    \@dtl@dogetval
    \global\let#1=\@dtl@val
  }%
}
```

\DTLgetrowforkey

```
\DTLgetrowforkey{⟨cmd⟩}{⟨db name⟩}{⟨ref key⟩}{⟨ref value⟩}
```

This (globally) sets ⟨*cmd*⟩ (a control sequence) to the first row of the database called ⟨*db name*⟩ which contains the key ⟨*ref key*⟩ that has the value ⟨*value*⟩.

```
\newcommand*{\DTLgetrowforkey}[4]{%
  \global\let#1=\@empty
  \@sDTLforeach{#2}{\dtl@refvalue=#3}{%
    \DTLifnull{\dtl@refvalue}%
    {}%
    {%
      \ifthenelse{\equal{\dtl@refvalue}{#4}}%
      {%
        \xdef#1{\the\dtlcurrentrow}%
        \dtlbreak
      }%
      {}%
    }%
  }%
}
```

12.10 Sorting Databases

The sorting commands have been rewritten in v3.0 to use l3seq.

```
\ExplSyntaxOn
```

The older `\dtlsort` has the following arguments:

- Replacement list (optional): comma-separated list of column keys to use as a replacement if the given value is null.
- Sort criteria: comma-separated list of ⟨*key*⟩=⟨*order*⟩ items, where ⟨*order*⟩ may be `ascending` or `descending`. The order may be omitted, in which case ascending is assumed.
- Handler function: as for `\dtlsortlist`

The newer `\DTLsortdata` command works using a similar principle to `\DTLsortwordlist` but the items in the word list sequence need to include the row specs so the database can be reconstructed afterwards.

The sort criteria argument has a different syntax for the list where each criteria in the list is given by ⟨*key*⟩=`{`⟨*options*⟩`}` where ⟨*key*⟩ is the column key. The options may be: ascending, descending, and replacements=`{`⟨*key list*⟩`}`. This means that a different set of replacements in the event of null can be supplied for each key.

The handler function is the same as for `\DTLsortword list`. Need a sequence to keep track of order options: true indicates ascending and false descending.

```
\seq_new:N \l__datatool_sort_order_seq
```

A sequence to keep track of whether the current sort criteria is for a numeric column. True for a numeric column, false otherwise.

```
\seq_new:N \l__datatool_sort_numeric_seq
```

A sequence to keep track of the replacement lists. Each item in the sequence will be the csv list.

```
\seq_new:N \l__datatool_replacement_indexes_seq
```

```
\bool_new:N \l__datatool_sort_order_bool
\clist_new:N \l__datatool_sort_replacements_clist
\keys_define:nn { datatool / sortdata / criteria }
{
  ascending .bool_set:N = \l__datatool_sort_order_bool ,
  descending .bool_set_inverse:N = \l__datatool_sort_order_bool ,
  asc .code:n =
   {
     \bool_set_true:N \l__datatool_sort_order_bool
   } ,
  asc .value_forbidden:n = true ,
  desc .code:n =
   {
     \bool_set_false:N \l__datatool_sort_order_bool
   } ,
  desc .value_forbidden:n = true ,
  replacements .code:n =
   {
     \clist_set:No
       \l__datatool_sort_replacements_clist
        { #1 }
   } ,
}
```

Map function used for obtaining indexes of replacement columns:

```
\cs_new:Nn \__datatool_replacement_handler:n
{
 \tl_if_exist:cTF
  {
    dtl@ci
    @ \l__datatool_default_dbname_tl
    @ #1
  }
  {
    \clist_put_right:Nv \l__datatool_tmpa_clist
     {
       dtl@ci
       @ \l__datatool_default_dbname_tl
       @ #1
     }
  }
  {
   \__datatool_sortdata_missing_column:nn
    {
      \token_to_str:N \DTLsortdata : ~ No ~ column ~ with ~ key ~
      ` #1 ' ~ in ~ database ~
      ` \l__datatool_default_dbname_tl ' ~
      in ~ replacement ~ list ~ for ~
      column ~ key ` #1 '. ~
      Ignoring ~ ` \l__datatool_item_key_tl '
    }
    {
      Check ~ that ~ you ~ have ~ correctly ~ spelt ~ the ~
      column ~ key ~ and ~ database ~ name ~
    }
  }
}
```

Parse the sort criteria list. This will set three sequences that correspond to each item in the list, so the $\langle i \rangle$th item in each sequence corresponds to the $\langle i \rangle$th item in the criteria list.

```
\cs_new:Nn \__datatool_parse_sort_criteria_list:n
{
  \seq_clear:N \l__datatool_sort_order_seq
  \seq_clear:N \l__datatool_sort_numeric_seq
  \seq_clear:N \l__datatool_column_indexes_seq
  \seq_clear:N \l__datatool_replacement_indexes_seq
  \clist_map_inline:nn { #1 }
   {
```

Parse optional criteria:

```
      \keyval_parse:NNn
        \__datatool_parse_sort_criteria:n
        \__datatool_parse_sort_criteria:nn
```

```
    { ##1 }
  \tl_if_exist:cTF
   {
     dtl@ci
      @ \l__datatool_default_dbname_tl
      @ \l__datatool_item_key_tl
   }
   {
     \exp_args:NNc
       \int_set:Nn \l__datatool_col_idx_int
        {
           dtl@ci
           @ \l__datatool_default_dbname_tl
           @ \l__datatool_item_key_tl
        }
```

Set the column index for this key.

```
  \seq_put_right:Nx \l__datatool_column_indexes_seq
    { \int_use:N \l__datatool_col_idx_int }
```

Set the order for this key.

```
  \bool_if:NTF \l__datatool_sort_order_bool
   {
     \seq_put_right:Nn \l__datatool_sort_order_seq
       { \c_true_bool }
   }
   {
     \seq_put_right:Nn \l__datatool_sort_order_seq
       { \c_false_bool }
   }
```

Get the column type.

```
  \__datatool_get_col_type:vV
    { dtlkeys @ \l__datatool_default_dbname_tl }
     \l__datatool_col_idx_int
```

Set the numeric flag for this key.

```
  \datatool_if_numeric_datum_type:nTF
     { \l__datatool_item_type_int }
   {
```

Numeric column.

```
     \seq_put_right:Nn \l__datatool_sort_numeric_seq
       { \c_true_bool }
   }
   {
```

Not numeric column.

```
     \seq_put_right:Nn \l__datatool_sort_numeric_seq
       { \c_false_bool }
   }
```

Indexes of replacement columns.

```
        \clist_clear:N \l__datatool_tmpa_clist
        \clist_map_function:NN
           \l__datatool_sort_replacements_clist
           \__datatool_replacement_handler:n
        \seq_put_right:Nx \l__datatool_replacement_indexes_seq
          { \l__datatool_tmpa_clist }
      }
      {
        \__datatool_sortdata_missing_column:nn
        {
          \token_to_str:N \DTLsortdata : ~ No ~ column ~ with ~ key ~
          ` \l__datatool_item_key_tl ' ~ in ~ database ~
          ` \l__datatool_default_dbname_tl ' . ~
          Ignoring ~ ` \l__datatool_item_key_tl '
        }
        {
          Check ~ that ~ you ~ have ~ correctly ~ spelt ~ the ~
          column ~ key ~ and ~ database ~ name ~
        }
      }
   }
}
```

Parse the sort criteria where only the column key is provided.

```
\cs_new:Nn \__datatool_parse_sort_criteria:n
{
```

Set defaults:

```
  \bool_set_true:N \l__datatool_sort_order_bool
  \clist_clear:N \l__datatool_sort_replacements_clist
  \tl_set:Nn \l__datatool_item_key_tl { #1 }
}
```

Parse the sort criteria where the column key and option list are provided.

```
\cs_new:Nn \__datatool_parse_sort_criteria:nn
{
  \__datatool_parse_sort_criteria:n { #1 }
```

Parse the sort criteria options.

```
  \keys_set:nn { datatool / sortdata / criteria } { #2 }
}
```

Syntax: ⟨*handler-cs*⟩`{`⟨*numeric bool var*⟩`}{`⟨*col-idx*⟩`}{`⟨*row-spec*⟩`}`⟨*replacement list*⟩
The column index ⟨*col-idx*⟩ needs to be fully expanded. This will set `\l__datatool_content_tl` to the sort value, using the fallback list in the event of a null value.

```
\cs_new:Nn \__datatool_set_sort_value:NNnnN
{
```

Get the entry from the row specs.

```
  \__datatool_get_entry_from_row:Nnn
    \l__datatool_item_value_tl { #3 } { #4 }
```

Parse the value.

```
\__datatool_if_replace:NTF \l__datatool_item_value_tl
 {
```

No value. Try column in the replacement list if available.

```
      \tl_set_eq:NN \l__datatool_content_tl \dtlnovalue
      \clist_if_empty:NF #5
       {
         \clist_pop:NN #5 \l__datatool_item_col_tl
         \__datatool_set_sort_value:NNVnN
            #1 #2 \l__datatool_item_col_tl
            { #4 } #5
       }
  }
  {
    \cs_if_eq:NNF \__datatool_encap_sort:nnn \use_i:nnn
     {
       \tl_set:Ne \l__datatool_item_value_tl
         {
           \__datatool_encap_sort:onn
             { \l__datatool_item_value_tl }
             { #3 } { \l__datatool_default_dbname_tl }
         }
     }
    \__datatool_parse:N \l__datatool_item_value_tl
    \bool_if:NTF #2
     {
```

Numeric column. Set sort to the numeric value.

```
       \datatool_if_numeric_datum_type:nTF { \@dtl@datatype }
        {
          \tl_set_eq:NN \l__datatool_content_tl
            \l__datatool_datum_value_tl
        }
        {
```

If the column type is numeric but the value isn't, expand fully and try again.

```
          \tl_set:Nx \l__datatool_item_value_tl
            { \l__datatool_datum_original_value_tl }
          \__datatool_parse:N \l__datatool_item_value_tl
          \datatool_if_numeric_datum_type:nTF { \@dtl@datatype }
          {
            \tl_set_eq:NN \l__datatool_content_tl
              \l__datatool_datum_value_tl
          }
          {
```

Value still isn't numeric so set to zero.

```
            \tl_set:Nn \l__datatool_content_tl { 0 }
            \tl_set:Nx \l__datatool_item_value_tl
             {
```

```
                \exp_not:N \__datatool_datum:nnnn
                 { \l__datatool_item_value_tl }
                 { 0 } { }
                 { \c_datatool_integer_int }
               }
             }
           }
         }
         {
```

Not numeric column. Convert to byte sequence.

```
          \exp_args:NV #1
            \l__datatool_datum_original_value_tl
            \l__datatool_content_tl
          \tl_set:Nx \l__datatool_item_value_tl
            {
              \exp_not:N \__datatool_datum:nnnn
               { \l__datatool_item_value_tl }
               {  } { }
               { \c_datatool_string_int }
            }
         }
    }
```

If null, treat as empty for strings and 0 for numbers.

```
   \datatool_if_null:NT \l__datatool_content_tl
    {
      \bool_if:NTF #2
       {
         \tl_set:Nn \l__datatool_content_tl { 0 }
       }
       {
         \tl_clear:N \l__datatool_content_tl
       }
      \tl_set_eq:NN \l__datatool_item_value_tl \dtlnovalue
    }
```

The sort value should now be in `\l__datatool_content_tl` and the actual value (or null) in `\l__datatool_item_value_tl`.

```
 }
 \cs_generate_variant:Nn
   \__datatool_set_sort_value:NNnnN
   { NNnVN , NNVnN }
```

Options for `\DTLsortdata`. Handler function:

```
 \cs_new:Nn \__datatool_default_sort_data_fn:nN
 {
   \DTLsortwordhandler { #1 } #2
 }
 \cs_set_eq:NN
   \__datatool_sort_data_fn:nN
   \__datatool_default_sort_data_fn:nN
```

If the sort value should be saved in a column. Index or key:

```
\int_new:N \__datatool_sort_data_sortcol_int
\tl_new:N \__datatool_sort_data_sortcol_tl
```

If the letter group should be saved in a column. Index or key:

```
\int_new:N \__datatool_sort_data_grpcol_int
\tl_new:N \__datatool_sort_data_grpcol_tl
```

Determines whether or not the value in the given token list variable should be replaced.

```
\cs_set_eq:NN
  \__datatool_if_replace:NTF
  \datatool_if_null_or_empty:NTF
```

Encapsulate non-numeric sort values:

```
\cs_new:Nn \__datatool_encap_sort:nnn { #1 }
\cs_generate_variant:Nn \__datatool_encap_sort:nnn { onn }
```

Allow user to choose between error, warning or ignore missing columns.

```
\cs_new:Nn \__datatool_sortdata_missing_column_err:nn
{
  \PackageError { datatool } { #1 } { #2 }
}
\cs_new:Nn \__datatool_sortdata_missing_column_warn:nn
{
  \PackageWarning { datatool } { #1 }
}
\cs_set_eq:NN
 \__datatool_sortdata_missing_column:nn
 \__datatool_sortdata_missing_column_err:nn
```

Define keys.

```
\keys_define:nn { datatool / sortdata }
{
  missing-column-action .choice: ,
  missing-column-action / error .code:n =
   {
     \cs_set_eq:NN
      \__datatool_sortdata_missing_column:nn
      \__datatool_sortdata_missing_column_err:nn
   } ,
  missing-column-action / warn .code:n =
   {
     \cs_set_eq:NN
      \__datatool_sortdata_missing_column:nn
      \__datatool_sortdata_missing_column_warn:nn
   } ,
  missing-column-action / ignore .code:n =
   {
     \cs_set_eq:NN
      \__datatool_sortdata_missing_column:nn
      \use_none:nn
```

```
 } ,
function .code:n =
 {
   \cs_set_eq:NN \__datatool_sort_data_fn:nN #1
 } ,
encap .code:n =
 {
   \tl_if_empty:nTF { #1 }
    {
      \cs_set_eq:NN \__datatool_encap_sort:nnn \use_i:nnn
    }
    {
      \cs_set:Nn \__datatool_encap_sort:nnn
       { #1 { ##1 } { ##2 } { ##3 } }
    }
 },
replace .choice: ,
replace / null .code:n =
 {
   \cs_set_eq:NN
      \__datatool_if_replace:NTF
       \datatool_if_null:NTF
 } ,
replace / null ~ or ~ empty .code:n =
 {
   \cs_set_eq:NN
      \__datatool_if_replace:NTF
       \datatool_if_null_or_empty:NTF
 } ,
save-sort .code:n =
 {
   \tl_set:Nn \__datatool_sort_data_sortcol_tl { sort }
   \int_zero:N \__datatool_sort_data_sortcol_int
 } ,
save-sort .value_forbidden:n = true ,
save-group .code:n =
 {
   \tl_set:Nn \__datatool_sort_data_grpcol_tl { group }
   \int_zero:N \__datatool_sort_data_grpcol_int
 } ,
save-group .value_forbidden:n = true ,
save-sort-column .code:n =
 {
   \int_set:Nn \__datatool_sort_data_sortcol_int { #1 }
   \tl_clear:N \__datatool_sort_data_sortcol_tl
 },
save-sort-column .value_required:n = true ,
save-sort-key .code:n =
 {
   \tl_set:Nn \__datatool_sort_data_sortcol_tl { #1 }
```

```
      \int_zero:N \__datatool_sort_data_sortcol_int
    },
  save-sort-key .value_required:n = true ,
  save-group-column .code:n =
    {
      \int_set:Nn \__datatool_sort_data_grpcol_int { #1 }
      \tl_clear:N \__datatool_sort_data_grpcol_tl
    },
  save-group-column .value_required:n = true ,
  save-group-key .code:n =
    {
      \tl_set:Nn \__datatool_sort_data_grpcol_tl { #1 }
      \int_zero:N \__datatool_sort_data_grpcol_int
    },
  save-group-key .value_required:n = true ,
}
```

\DTLsortdata

\DTLsortdata[⟨*options*⟩]{⟨*db-name*⟩}{⟨*sort criteria*⟩}

```
\NewDocumentCommand \DTLsortdata
  { o m m }
{
  \__datatool_db_sort:nnn { #1 } { #2 } { #3 }
}
```

This just makes it easier for the sort action to expand the arguments without the inconvenience of the optional syntax.

```
\cs_new:Nn \__datatool_db_sort:nnn
{
```

Initialise:

```
  \cs_set_eq:NN
    \__datatool_sort_data_fn:nN
    \__datatool_default_sort_data_fn:nN
  \cs_set_eq:NN
    \__datatool_if_replace:NTF
    \datatool_if_null_or_empty:NTF
  \cs_set_eq:NN \__datatool_encap_sort:nnn \use_i:nnn
  \int_zero:N \__datatool_sort_data_sortcol_int
  \tl_clear:N \__datatool_sort_data_sortcol_tl
  \int_zero:N \__datatool_sort_data_grpcol_int
  \tl_clear:N \__datatool_sort_data_grpcol_tl
  \IfValueT { #1 }
   {
     \keys_set:nn { datatool / sortdata } { #1 }
   }
  \tl_if_empty:nTF { #2 }
  {
    \exp_args:No \__datatool_db_sort:nn
```

```
      { \l__datatool_default_dbname_tl }
      { #3 }
   }
   {
     \exp_args:Nx \__datatool_db_sort:nn
      { \text_purify:n { #2 } } { #3 }
   }
 }
 \cs_generate_variant:Nn \__datatool_db_sort:nnn
  { VVV }
```

Syntax: {⟨*db-name*⟩}{⟨*sort criteria*⟩}

```
 \cs_new:Nn \__datatool_db_sort:nn
 {
  \datatool_db_state:nnnn { #1 }
  {
   \__datatool_sort_db_check_opts:n { #1 }
```

Scope to localise the effect of the hook.

```
   \group_begin:
   \tl_set:Nx \l__datatool_default_dbname_tl { #1 }
   \int_compare:nNnTF
      { \DTLrowcount {  \l__datatool_default_dbname_tl } }
       >
      { 200 }
    {
      \typeout
       {
         Sorting ~ database ~
        ` \l__datatool_default_dbname_tl' ~ - ~
        this ~ may ~ take ~ a ~ while.
       }
    }
    {
      \dtl@message { Sorting ~ database ~ ` \l__datatool_default_dbname_tl ' }
    }
   \dtl@SortWordCommands@hook
   \seq_clear:N \l__datatool_wordlist_seq
```

Parse the sort criteria. This will set the sequence variables: `\l__datatool_sort_order_seq` `\l__datatool_sort_numeric_seq` `\l__datatool_column_indexes_seq`

```
   \__datatool_parse_sort_criteria_list:n { #2 }
   \seq_if_empty:NTF \l__datatool_column_indexes_seq
   {
     \PackageError { datatool }
      {
        \token_to_str:N \DTLsortdata : ~ missing ~ sort ~
        criteria
      }
      {
```

```
        The ~ final ~ argument ~ of ~ \token_to_str:N \DTLsortdata
        \c_space_tl ~ must ~ have ~ at ~ least ~ one ~ column ~ key
      }
    }
    {
```

For each row in the database, set the list of sort values for each specified column.

```
      \bool_set_true:N \l__datatool_sort_datum_bool
      \DTLmapdata [ read-only ]
      {
        \seq_clear:N \l__datatool_sorta_seq
        \seq_map_indexed_inline:Nn
          \l__datatool_column_indexes_seq
         {
           \exp_args:NNx
             \bool_set:Nn \l__datatool_sort_numeric_bool
              {
                \seq_item:Nn
                  \l__datatool_sort_numeric_seq { ##1 }
              }
           \tl_set:Nx
             \l__datatool_sort_replacements_clist
              {
                \seq_item:Nn
                  \l__datatool_replacement_indexes_seq { ##1 }
              }
```

This will set `\l__datatool_content_tl` to the sort value, and the original value (or null) will be in `\l__datatool_item_value_tl`.

```
           \__datatool_set_sort_value:NNnVN
             \__datatool_sort_data_fn:nN
              \l__datatool_sort_numeric_bool
              { ##2 }
              \l__datatool_map_data_row_tl
             \l__datatool_sort_replacements_clist
          \bool_if:NTF \l__datatool_sort_numeric_bool
          {
            \dtl@message{ Row ~ \int_use:N \dtlrownum \c_space_tl ~
              numeric ~ sort ~ value ~ for ~ column ~ ##2 : ~
              \l__datatool_content_tl }
          }
          {
            \dtl@message{ Row ~ \int_use:N \dtlrownum \c_space_tl ~
              byte ~ sort ~ value ~ for ~ column ~ ##2 : ~
              \l__datatool_content_tl }
          }
```

Append to the word list:

```
           \seq_put_right:Nx
             \l__datatool_sorta_seq
              {
```

```
              { \exp_not:V \l__datatool_content_tl }
              { \exp_not:V \l__datatool_item_value_tl }
            }
       }
      \exp_args:Noo \__datatool_sortword_append:nn
         {\l__datatool_sorta_seq }
         { \l__datatool_map_data_row_tl }
    }
  }
  \exp_args:NVV
    \__datatool_start_datasortword_list:nn
       \l__datatool_wordlist_seq
       \l__datatool_sort_order_seq
  \__datatool_finish_sort_db:n { #1 }
 }
 { }% empty
 {
   \PackageError { datatool }
   {
     \token_to_str:N \DTLsortdata : ~ database ~
     `#1' ~ doesn't ~ exist
   }
   {
     Check ~ that ~ you ~ have ~ correctly ~ spelt ~
     the ~ database ~ name ~ or ~ set ~ the ~ default ~
     name, ~ as ~ applicable
   }
 }
}
\cs_new:Nn \__datatool_start_datasortword_list:nn
{
   \group_end:
   \tl_set:Nn \l__datatool_wordlist_seq { #1 }
   \tl_set:Nn \l__datatool_sort_order_seq { #2 }
   \cs_set_eq:NN
    \__datatool_compare_sortitem:w
    \__datatool_compare_datasortitem:w
   \seq_sort:Nn \l__datatool_wordlist_seq
   {
     \__datatool_compare_sortitem:w ##1 ##2
   }
}
\seq_new:N \l__datatool_sorta_seq
\seq_new:N \l__datatool_sortb_seq
\cs_new:Npn \__datatool_compare_datasortitem:w
 #1\q_mark#2\q_stop#3\q_mark#4\q_stop
 {
   \tl_set:Nn \l__datatool_sorta_seq { #1 }
   \tl_set:Nn \l__datatool_sortb_seq { #3 }
```

```
\seq_set_eq:NN
  \l__datatool_tmp_seq
  \l__datatool_sort_order_seq
\__datatool_compare_datasortitem:
\int_compare:nNnTF
    { \dtl@sortresult } < { \c_zero_int }
 {
   \sort_return_same:
 }
 {
   \int_compare:nNnTF
      { \dtl@sortresult } > { \c_zero_int }
    {
      \sort_return_swapped:
    }
    {
      \bool_lazy_or:nnTF
        { \seq_if_empty_p:N \l__datatool_sorta_seq }
        { \seq_if_empty_p:N \l__datatool_sortb_seq }
      {
        \sort_return_same:
      }
      {
        \tl_set:Nn \l__datatool_sorta_seq { #1 }
        \tl_set:Nn \l__datatool_sortb_seq { #3 }
        \seq_get_left:NN
           \l__datatool_sorta_seq
           \l__datatool_tmpa_tl
        \seq_get_left:NN
           \l__datatool_sortb_seq
           \l__datatool_tmpb_tl
      \tl_set:Nx \l__datatool_tmpa_tl
        {
          \exp_last_unbraced:NV
            \use_ii:nn \l__datatool_tmpa_tl
        }
      \tl_set:Nx \l__datatool_tmpb_tl
        {
          \exp_last_unbraced:NV
            \use_ii:nn \l__datatool_tmpb_tl
        }
      \bool_if:NTF \l__datatool_sort_reverse_bool
       {
         \exp_args:NVV
         \__datatool_fallback_action:nnnn
           \l__datatool_tmpb_tl
           \l__datatool_tmpa_tl
          { \sort_return_swapped: }
          { \sort_return_same: }
       }
```

```
          {
            \exp_args:NVV
            \__datatool_fallback_action:nnnn
              \l__datatool_tmpa_tl
              \l__datatool_tmpb_tl
             { \sort_return_swapped: }
             { \sort_return_same: }
          }
         }
       }
    }
 }
\cs_new:Nn \__datatool_compare_datasortitem:
{
   \bool_lazy_or:nnTF
     { \seq_if_empty_p:N \l__datatool_sorta_seq }
     { \seq_if_empty_p:N \l__datatool_sortb_seq }
   {
     \int_zero:N \dtl@sortresult
   }
   {
     \seq_if_empty:NTF \l__datatool_tmp_seq
      {
        \bool_set_false:N \l__datatool_sort_reverse_bool
      }
      {
        \seq_pop_left:NN
          \l__datatool_tmp_seq
          \l__datatool_sort_reverse_bool
        \bool_set_inverse:N \l__datatool_sort_reverse_bool
      }
     \seq_pop_left:NN
        \l__datatool_sorta_seq
        \l__datatool_tmpa_tl
     \seq_pop_left:NN
        \l__datatool_sortb_seq
        \l__datatool_tmpb_tl
     \exp_args:NVV
       \__datatool_compare_sortitem:nn
         \l__datatool_tmpa_tl
         \l__datatool_tmpb_tl
       \int_if_zero:nT { \dtl@sortresult }
        {
          \__datatool_compare_datasortitem:
        }
    }
}
```

Arguments expected to be braced pairs.

```
\cs_new:Nn \__datatool_compare_sortitem:nn
```

```
{
  \__datatool_compare_sortitem:nnnn #1 #2
}
```
Check the options provided to `\DTLsortdata`
```
\cs_new:Nn \__datatool_sort_db_check_opts:n
{
```
Does the sort value or letter group need to be saved? Sort column:
```
  \tl_if_empty:NF \__datatool_sort_data_sortcol_tl
   {
     \tl_if_exist:cTF
       { dtl@ci@ #1 @ \__datatool_sort_data_sortcol_tl }
      {
        \int_set:Nn \__datatool_sort_data_sortcol_int
         {
           \use:c { dtl@ci@ #1 @ \__datatool_sort_data_sortcol_tl }
         }
      }
      {
```
Doesn't exist so a new column needs to be created.
```
        \__datatool_add_column_with_header:nxxx
          { #1 }
          { \__datatool_sort_data_sortcol_tl }
          { \int_use:N \c_datatool_string_int }
          { \__datatool_sort_data_sortcol_tl }
        \int_set:Nn \__datatool_sort_data_sortcol_int
          { \DTLcolumncount { #1 } }
      }
   }
```
Letter group column:
```
  \tl_if_empty:NF \__datatool_sort_data_grpcol_tl
   {
     \tl_if_exist:cTF
       { dtl@ci@ #1 @ \__datatool_sort_data_grpcol_tl }
      {
        \int_set:Nn \__datatool_sort_data_grpcol_int
         {
           \use:c { dtl@ci@ #1 @ \__datatool_sort_data_grpcol_tl }
         }
      }
      {
```
Doesn't exist so a new column needs to be created.
```
        \__datatool_add_column_with_header:nxxx
          { #1 }
          { \__datatool_sort_data_grpcol_tl }
          { \int_use:N \c_datatool_string_int }
          { \__datatool_sort_data_grpcol_tl }
        \int_set:Nn \__datatool_sort_data_grpcol_int
```

```
          { \DTLcolumncount { #1 } }
      }
    }
```

If the column indexes were supplied rather than the keys, check if they exist. This needs to be done after the key check in case one was referenced by key and another by index. A reference by column index must exist otherwise it gets too complicated.

```
    \int_compare:nNnT
     { \__datatool_sort_data_sortcol_int } > { \DTLcolumncount { #1 } }
    {
      \PackageError { datatool }
       {
         save-sort-column ~ value ~
         \int_use:N \__datatool_sort_data_sortcol_int
         \c_space_tl ~ out ~ of ~ range. ~ Ignoring
       }
       {
         Database ~ ` #1 ' ~ only ~ has ~ \DTLcolumncount { #1 } ~
         columns. ~ Use ~ save-sort-key ~ instead ~ if ~ you ~
         want ~ to ~ create ~ a ~ new ~ column
       }
      \int_zero:N \__datatool_sort_data_sortcol_int
    }
    \int_compare:nNnT
     { \__datatool_sort_data_grpcol_int } > { \DTLcolumncount { #1 } }
    {
      \PackageError { datatool }
       {
         save-group-column ~ value ~
         \int_use:N \__datatool_sort_data_grpcol_int
         \c_space_tl ~ out ~ of ~ range. ~ Ignoring
       }
       {
         Database ~ ` #1 ' ~ only ~ has ~ \DTLcolumncount { #1 } ~
         columns. ~ Use ~ save-group-key ~ instead ~ if ~ you ~
         want ~ to ~ create ~ a ~ new ~ column
       }
      \int_zero:N \__datatool_sort_data_grpcol_int
    }
}
```

Reconstruct database after sorting.

```
\cs_new:Nn \__datatool_finish_sort_db:n
{
   \tl_clear:N \l__datatool_content_tl
   \seq_map_indexed_inline:Nn \l__datatool_wordlist_seq
   {
     \__datatool_finish_sort_db_parse:w ##2 { ##1 }
   }
   \bool_if:NTF \l__datatool_db_global_bool
```

```
    {
     \__datatool_token_register_gset:cV
      { dtldb @ #1 }
      \l__datatool_content_tl
    }
    {
     \__datatool_token_register_set:cV
      { dtldb @ #1 }
      \l__datatool_content_tl
    }
}
```

`\dtlsortdatavalue` Used to encapsulate the sort value when returned by the save-sort... settings. The first argument is the actual byte sequence. The second is the value used to form the byte sequence.

```
\newcommand{\dtlsortdatavalue}[2]{#2}
```

Obtain the original row markup.

```
\cs_new:Npn \__datatool_finish_sort_db_parse:w
 #1 \q_mark #2 \q_stop #3
{
  \tl_set:Nn \l__datatool_tmp_seq { #1 }
  \tl_set:Nn \l__datatool_row_tl
   { #2 }
  \int_compare:nNnT
    { \__datatool_sort_data_sortcol_int } > { \c_zero_int }
   {
```

The sort value will be in the form `\dtlsortdatavalue{`⟨*sort*⟩`}{`⟨*datum*⟩`}`

```
      \tl_clear:N \l__datatool_tmpa_tl
      \seq_map_inline:Nn \l__datatool_tmp_seq
      {
        \tl_if_eq:nnF
          { ##1 } { { } { \c_datatool_nullvalue_tl } }
         {
           \tl_set:Nn \l__datatool_tmpa_tl { ##1 }
         }
       \tl_if_empty:NF \l__datatool_tmpa_tl
         {
           \seq_map_break:
         }
      }
      \tl_if_empty:NF \l__datatool_tmpa_tl
      {
        \tl_put_left:Nn \l__datatool_tmpa_tl
          { \dtlsortdatavalue }
        \exp_args:NVV
         \__datatool_if_split_row:nnNNTF
         \l__datatool_row_tl
         \__datatool_sort_data_sortcol_int
```

```
          \l__datatool_map_data_edit_before_tl
          \l__datatool_map_data_edit_after_tl
        {
```

Something already exists in this column, so replace it.

```
            \tl_set:Nx \l__datatool_row_tl
             {
               \exp_not:V \l__datatool_map_data_edit_before_tl
               \__datatool_row_element_markup:VV
                  \__datatool_sort_data_sortcol_int
                  \l__datatool_tmpa_tl
               \exp_not:V \l__datatool_map_data_edit_after_tl
             }
          }
          {
            \tl_put_right:Nx \l__datatool_row_tl
             {
               \__datatool_row_element_markup:VV
                  \__datatool_sort_data_sortcol_int
                  \l__datatool_tmpa_tl
             }
          }
       }
    }
   \int_compare:nNnT
    { \__datatool_sort_data_grpcol_int } > { \c_zero_int }
    {
```

Fetch the letter group corresponding to the first sort item.

```
      \tl_clear:N \l__datatool_tmpa_tl
      \seq_map_inline:Nn \l__datatool_tmp_seq
      {
        \tl_if_eq:nnF
          { ##1 } { { } { \c_datatool_nullvalue_tl } }
         {
           \tl_set:Nn \l__datatool_tmpa_tl { ##1 }
         }
       \tl_if_empty:NF \l__datatool_tmpa_tl
         {
           \seq_map_break:
         }
      }
      \__datatool_sort_db_fetch_grp:N \l__datatool_tmpa_tl
```

Use hook to post-process the letter group token variable.

```
      \datatool_post_process_lettergroup:N
        \l__datatool_tmpa_tl
      \exp_args:NVV
       \__datatool_if_split_row:nnNNTF
       \l__datatool_row_tl
       \__datatool_sort_data_grpcol_int
```

```
        \l__datatool_map_data_edit_before_tl
        \l__datatool_map_data_edit_after_tl
      {
```

Something already exists in this column, so replace it.

```
          \tl_set:Nx \l__datatool_row_tl
          {
            \exp_not:V \l__datatool_map_data_edit_before_tl
            \__datatool_row_element_markup:VV
               \__datatool_sort_data_grpcol_int
               \l__datatool_tmpa_tl
            \exp_not:V \l__datatool_map_data_edit_after_tl
          }
      }
      {
        \tl_put_right:Nx \l__datatool_row_tl
          {
            \__datatool_row_element_markup:VV
               \__datatool_sort_data_grpcol_int
               \l__datatool_tmpa_tl
          }
      }
  }
```

Add in the start and end row markers.

```
  \tl_put_right:Nx \l__datatool_content_tl
   {
     \__datatool_row_markup:nV
       { #3 } \l__datatool_row_tl
   }
}
```

Syntax: ⟨*grp tl var*⟩ The current row specs should be in `\l__datatool_row_tl`

```
\cs_new:Nn \__datatool_sort_db_fetch_grp:N
{
  \int_compare:nNnTF
    { \tl_count:N #1 } = { 2 }
   {
     \exp_after:wN \__datatool_sort_db_fetch_grp:nnN #1 #1
   }
   {
     \tl_clear:N #1
   }
}
\cs_new:Nn \__datatool_sort_db_fetch_grp:nnN
{
  \tl_clear:N #3
  \DTLassignlettergroup { #2 } { #1 } #3
}
```

Hook to post-process the letter group token variable.

```
\cs_new:Nn \datatool_post_process_lettergroup:N
{
}
```

`\@dtl@list` Version 3.0: removed `\@dtl@list`. Token register to store data when sorting.

`\DTLsort` `\DTLsort[⟨replacement keys⟩]{⟨sort criteria⟩}{⟨db name⟩}`

Sorts database ⟨*db name*⟩ according to `{`⟨*sort criteria*⟩`}`, which must be a comma separated list of keys, and optionally `=`⟨*order*⟩, where ⟨*order*⟩ is either `ascending` or `descending`. The optional argument is a list of keys to uses if the given key has a null value. The starred version uses a case insensitive string comparison.

```
\newcommand*{\DTLsort}{\@ifstar\@sDTLsort\@DTLsort}
```

`\@DTLsort` Unstarred (case sensitive) version.

```
\newcommand{\@DTLsort}[3][]{%
  \dtlsort[#1]{#2}{#3}{\dtlcompare}%
}
```

`\@sDTLsort` Starred (case insensitive) version.

```
\newcommand*{\@sDTLsort}[3][]{%
  \dtlsort[#1]{#2}{#3}{\dtlicompare}%
}
```

The `\dtlsort` macro uses some of the same sequences as `\DTLsortdata`: `\l__datatool_replacement_indexes_seq` (list of replacement columns), `\l__datatool_sort_numeric_seq` (boolean sort sequence), `\l__datatool_column_indexes_seq` (list of columns to sort by) and `\l__datatool_sort_order_seq` (sort order). The database rows will be temporarily stored in the following sequence, which will be sorted according to the handler.

```
\seq_new:N \l__datatool_dtlsort_rows_seq
```

Temporary sort values.

```
\tl_new:N \l__datatool_sorta_tl
\tl_new:N \l__datatool_sortb_tl
```

`\dtlsort` `\dtlsort[⟨replacement keys⟩]{⟨sort criteria⟩}{⟨db name⟩}{⟨handler⟩}`

More general version where user supplies a handler for the comparison.

```
\NewDocumentCommand \dtlsort { o m m m }
{
```

Check the database exists

```
  \DTLifdbexists { #3 }
  {
```

Scope.

```
\group_begin:
\tl_set:Nx \dtldbname { #3 }
```

List of replacement column indexes.

```
\seq_clear:N \l__datatool_replacement_indexes_seq
\IfValueT { #1 }
 {
   \clist_map_inline:nn { #1 }
    {
      \datatool_if_has_key:nnTF { #3 } { ##1 }
       {
         \seq_put_right:Nx \l__datatool_replacement_indexes_seq
          { \dtlcolumnindex { #3 } { ##1 } }
       }
       {
         \PackageWarning { datatool }
          { Ignoring ~ unknown ~ replacement ~ column ~ ##1 }
       }
    }
 }
```

List of sort column indexes. This also needs to keep track of the direction.

```
\seq_clear:N \l__datatool_column_indexes_seq
\seq_clear:N \l__datatool_sort_order_seq
\seq_clear:N \l__datatool_sort_numeric_seq
\keyval_parse:NNn
  \__datatool_dtlsort_col:n
  \__datatool_dtlsort_col:nn
  { #2 }
\seq_if_empty:NTF \l__datatool_column_indexes_seq
 {
   \PackageError { datatool }
    {
      \token_to_str:N \dtlsort : ~ No ~ sort ~ columns
    }
    {
      Either ~ the ~ criteria ~ argument ~ is ~ empty ~ or ~
      all ~ listed ~ column ~ keys ~ are ~ undefined
    }
 }
 {
   \seq_clear:N \l__datatool_dtlsort_rows_seq
   \int_step_inline:nn { \DTLrowcount { #3 } }
    {
      \__datatool_get_row:vnN
       { dtldb@ #3 } { ##1 } \l__datatool_sorta_tl
      \seq_put_right:NV \l__datatool_dtlsort_rows_seq
        \l__datatool_sorta_tl
    }
```

```
      \seq_sort:Nn \l__datatool_dtlsort_rows_seq
       {
         \__datatool_dtlsort_compare:Nnn #4 { ##1 } { ##2 }
       }
     }
    \exp_args:NNV
    \group_end:
     \__datatool_dtlsort_reconstruct:nn
       \l__datatool_dtlsort_rows_seq { #3 }
  }
  {
    \PackageError { datatool }
     { Database ~ ` #3 ' ~ doesn't ~ exist} { }
  }
}

\cs_new:Nn \__datatool_dtlsort_reconstruct:nn
{
  \tl_set:Nn \l__datatool_dtlsort_rows_seq { #1 }
  \__datatool_token_register_set:cn { dtldb@ #2 } { }
  \seq_map_indexed_inline:Nn \l__datatool_dtlsort_rows_seq
   {
     \bool_if:NTF \l__datatool_db_global_bool
      {
        \__datatool_token_register_gput_right:cx
         { dtldb@ #2 } { \__datatool_row_markup:nn { ##1 } { ##2 } }
      }
      {
        \__datatool_token_register_put_right:cx
         { dtldb@ #2 } { \__datatool_row_markup:nn { ##1 } { ##2 } }
      }
   }
}

\cs_new:Nn \__datatool_dtlsort_compare:Nnn
{
  \seq_map_indexed_inline:Nn \l__datatool_column_indexes_seq
   {
     \bool_set:Nn \l__datatool_sort_order_bool
      {
        \seq_item:Nn \l__datatool_sort_order_seq { ##1 }
      }
     \exp_args:NNx
       \bool_set:Nn \l__datatool_sort_numeric_bool
        {
          \seq_item:Nn
            \l__datatool_sort_numeric_seq { ##1 }
        }
     \__datatool_dtlsort_get_value:Nnn
       \l__datatool_sorta_tl { ##2 } { #2 }
     \__datatool_dtlsort_get_value:Nnn
```

```
      \l__datatool_sortb_tl { ##2 } { #3 }
    \bool_if:NTF \l__datatool_sort_numeric_bool
     {
       \DTLnumcompare \dtl@sortresult { \l__datatool_sorta_tl } { \l__datatool_sortb_tl
       \int_if_zero:nT { \dtl@sortresult }
        {
          \exp_args:NNVV #1 \dtl@sortresult \l__datatool_sorta_tl \l__datatool_sortb_tl
        }
     }
     {
       \exp_args:NNVV #1 \dtl@sortresult \l__datatool_sorta_tl \l__datatool_sortb_tl
     }
    \ifdtlverbose
      \dtl@message
       {
         `\exp_not:V \l__datatool_sorta_tl' ~
         <=> ~ `\exp_not:V \l__datatool_sortb_tl' ~ = ~ \int_use:N \dtl@sortresult
       }
    \fi
    \bool_if:NF \l__datatool_sort_order_bool
     {
       \int_set:Nn \dtl@sortresult { - \dtl@sortresult }
     }
    \int_if_zero:nTF { \dtl@sortresult }
     {
       \int_compare:nNnT
          { ##1 } = { \seq_count:N \l__datatool_column_indexes_seq }
        {
          \sort_return_same:
        }
     }
     {
       \int_compare:nNnTF { \dtl@sortresult } > { \c_zero_int }
        { \seq_map_break:n { \sort_return_swapped: } }
        { \seq_map_break:n { \sort_return_same: } }
     }
  }
}
\cs_new:Nn \__datatool_dtlsort_get_value:Nnn
{
  \__datatool_get_entry_from_row:Nnn
    #1 { #2 } { #3 }
  \bool_lazy_and:nnT
     { \tl_if_eq_p:NN #1 \dtlnovalue }
     { \bool_not_p:n { \seq_if_empty_p:N \l__datatool_replacement_indexes_seq } }
   {
     \seq_map_inline:Nn \l__datatool_replacement_indexes_seq
      {
        \__datatool_get_entry_from_row:Nnn
```

```
        #1 { ##1 } { #3 }
       \tl_if_eq:NNF #1 \dtlnovalue
        { \seq_map_break: }
     }
   }
  \bool_if:NT \l__datatool_sort_numeric_bool
   {
     \tl_if_eq:NNTF #1 \dtlnovalue
      {
        \DTLsetintegerdatum #1 { \c_datatool_nullvalue_tl } { 0 }
      }
      {
        \__datatool_parse:N #1
        \int_compare:nNnTF
           { \@dtl@datatype }
             <
           { \c_datatool_integer_int }
         {
           \bool_set_false:N \l__datatool_sort_numeric_bool
         }
      }
   }
}
\cs_new:Nn \__datatool_dtlsort_col:nN
{
  \datatool_if_has_key:nnTF { \dtldbname } { #1 }
   {
     \int_set:Nn \l__datatool_col_idx_int
      { \dtlcolumnindex { \dtldbname } { #1 } }
     \seq_put_right:NV \l__datatool_column_indexes_seq
      \l__datatool_col_idx_int
     \seq_put_right:Nn \l__datatool_sort_order_seq { #2 }
     \__datatool_get_col_type:vV
       { dtlkeys @ \dtldbname }
        \l__datatool_col_idx_int
     \datatool_if_numeric_datum_type:nTF
       { \l__datatool_item_type_int }
      {
```

Numeric column.

```
        \seq_put_right:Nn \l__datatool_sort_numeric_seq
          { \c_true_bool }
      }
      {
```

Not numeric column.

```
        \seq_put_right:Nn \l__datatool_sort_numeric_seq
          { \c_false_bool }
      }
   }
```

```
    {
      \PackageWarning { datatool }
        { Ignoring ~ unknown ~ sort ~ column ~ #1 }
    }
}
\cs_new:Nn \__datatool_dtlsort_col:n
{
  \__datatool_dtlsort_col:nN { #1 } \c_true_bool
}
\cs_new:Nn \__datatool_dtlsort_col:nn
{
  \tl_if_empty:nTF { #2 }
    {
      \__datatool_dtlsort_col:nN { #1 } \c_true_bool
    }
    {
      \tl_if_eq:nnTF { #2 } { descending }
       {
         \__datatool_dtlsort_col:nN { #1 } \c_false_bool
       }
       {
         \tl_if_eq:nnTF { #2 } { ascending }
          {
            \__datatool_dtlsort_col:nN { #1 } \c_true_bool
          }
          {
            \PackageError { datatool }
             {
               \token_to_str:N \dtlsort : ~ Invalid ~ sort ~ order ~ `#2' ~
               for ~ column ~ `#1'
             }
             {
               The ~ sort ~ order ~ may ~ be ~ either ~ `ascending' ~
               or ~ `descending'. ~ You ~ may ~ omit ~ the ~ order ~
               for ~ the ~ default ~ `ascending'
             }
          }
       }
    }
}
```

`\@dtl@rowa` Token register to store first row when sorting. Version 3.0: removed.

`\@dtl@rowb` Token register to store comparison row when sorting. Version 3.0: removed.

`\dtl@sortdata` `\dtl@sortdata{⟨db⟩}`

Sorts the data in named database using an insertion sort algorithm. `\@dtl@replacementkeys`, `\@dtl@sortorder` and `\@dtl@comparecs` must be set prior to use. Version 3.0: removed.

`\@dtl@sortcriteria`

```
\@dtl@sortcriteria{⟨row a toks⟩}{⟨row b toks⟩}
```

`\@dtl@dbname` and `\@dtl@sortorder` must be set before use `\@dtl@sortorder` is a comma separated list of either just keys or ⟨*key*⟩=⟨*direction*⟩. (Check keys are valid before use.) Version 3.0: removed.

`\@dtl@getsortdirection` Get the direction from either ⟨*key*⟩ or ⟨*key*⟩=⟨*direction*⟩. Sets `\@dtl@sortdirection` to either -1 (ascending) or 1 (descending). Version 3.0: removed.

`\@dtl@get@sortdirection` Get direction (trims trailing = sign) Version 3.0: removed.

`\dtl@compare`

```
\dtl@compare{⟨key⟩}{⟨a toks⟩}{⟨b toks⟩}
```

Compares two values according to ⟨*key*⟩ of database given by `\@dtl@dbname`. Sets `\dtl@sortresult`. `\@dtl@comparecs` must be set to the required comparison macro. Version 3.0: removed.

`\dtl@compare@`

```
\dtl@compare@{⟨keyA⟩}{⟨keyB⟩}{⟨A toks⟩}{⟨B toks⟩}
```

Compare ⟨*A*⟩ and ⟨*B*⟩ according ⟨*keyA*⟩ and ⟨*keyB*⟩ for database given by `\@dtl@dbname`. Sets `\dtl@sortresult`. `\@dtl@comparecs` must be set before use. Version 3.0: removed.

12.11 Saving a database to an external file

Version 3.0 provides a single command to save to the different types of files with an optional argument to specify the format.

```
\ior_new:N \g__datatool_in_stream
\iow_new:N \g__datatool_out_stream
```

Control whether or not the data should be expanded as it's written. The default doesn't expand.

```
\cs_new:Nn \__datatool_data:n { \exp_not:n { #1 } }
```

`\dtlspecialvalue` Elements starting with the special value marker will always be expanded.

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

\cs_new:Nn \__datatool_data_value:n
{
  \tl_if_head_eq_meaning:nNTF { #1 } \dtlspecialvalue
```

```
    { #1 }
    { \__datatool_data:n { #1 } }
 }
 \cs_new:Nn \__datatool_save_data_no_expand:Nn
  {
    \tl_set:Nx #1 { \tl_to_str:n { #2 } }
    \__datatool_save_data_postprocess:N #1
  }
 \cs_new:Nn \__datatool_save_data_protected_expand:Nn
  {
    \protected@edef #1 { #2 }
    \@onelevel@sanitize #1
    \__datatool_save_data_postprocess:N #1
  }
 \cs_new:Nn \__datatool_save_data_full_expand:Nn
  {
    \exp_args:NNx \__datatool_save_data_no_expand:Nn #1 { #2 }
  }
 \cs_set_eq:NN \__datatool_save_data:Nn \__datatool_save_data_no_expand:Nn
```

Escape delimiters:

```
 \cs_new:Nn \__datatool_save_escape_delim_bksl:N
  {
    \regex_replace_all:NnN \l__datatool_escape_delim_bksl_regex { \\\1 } #1
  }
 \cs_new:Nn \__datatool_save_escape_delim:N
  {
    \regex_replace_all:NnN \l__datatool_escape_delim_regex { \\\1 } #1
  }
 \cs_new:Nn \__datatool_save_double_delim:N
  {
    \regex_replace_all:NnN \l__datatool_escape_delim_regex { \1\1 } #1
  }
 \cs_set_eq:NN
   \__datatool_save_data_postprocess:N
   \__datatool_save_double_delim:N
```

Similarly unescape for reading data:

```
 \cs_new:Nn \__datatool_load_unescape_delim_bksl:N
  {
    \bool_if:NTF \l__datatool_csv_literal_content_bool
    {
      \regex_replace_all:NnN \l__datatool_unescape_str_delim_bksl_regex { \1 } #1
    }
    {
      \regex_replace_all:NnN \l__datatool_unescape_cs_delim_regex
         { \u{@dtl@delimiter} } #1
      \regex_replace_all:nnN
         { \c{ \x{5c} } ( [[:^alpha:]] | [a-zA-Z]+ ) }
         { \c{ \1 } }
```

```
        #1
    }
  }
\cs_new:Nn \__datatool_load_unescape_delim:N
  {
    \bool_if:NTF \l__datatool_csv_literal_content_bool
    {
      \regex_replace_all:NnN \l__datatool_unescape_str_delim_regex { \1 } #1
    }
    {
      \regex_replace_all:NnN \l__datatool_unescape_cs_delim_regex
         { \u{@dtl@delimiter} } #1
    }
  }
\cs_new:Nn \__datatool_load_unescape_double_delim:N
  {
    \bool_if:NTF \l__datatool_csv_literal_content_bool
    {
      \regex_replace_all:NnN \l__datatool_unescape_str_double_delim_regex { \1 } #1
    }
    {
      \regex_replace_all:NnN \l__datatool_unescape_cs_double_delim_regex
         { \1 } #1
    }
  }
\cs_set_eq:NN
  \__datatool_load_data_postprocess:N
  \__datatool_load_unescape_double_delim:N
```

The default format to save is CSV.

```
\cs_new:Nn \__datatool_save:n { \__datatool_save_csv:n { #1 } }
```

The default file extension.

```
\tl_new:N \l__datatool_default_ext_tl
\tl_set:Nn \l__datatool_default_ext_tl { .csv }
```

The default format to load is CSV.

```
\cs_new:Nn \__datatool_load: { \__datatool_load_csv: }
```

File overwrite.

```
\cs_new:Nn \__datatool_save_error_overwrite:nn
  {
    \PackageError { datatool }
    { Can't ~ save ~ to ~ file ~ `#1': ~ overwrite ~ forbidden}
    {
      The ~ current ~ overwrite ~ setting ~ doesn't ~ allow ~ an ~
      existing ~ file ~ to ~ be ~ overwritten. ~ Use ~ `overwrite=allow' ~
      option ~ to ~ allow ~ overwrite.
    }
  }
\cs_new:Nn \__datatool_save_warn_overwrite:nn
```

```
  {
    \PackageWarning { datatool }
    { File ~ `#1' ~ already ~ exists. ~ Skipping ~ save}
  }
 \cs_new:Nn \__datatool_save_allow_overwrite:nn
  {
    \dtl@message { Overwriting ~ `#1' }
    #2
  }
 \cs_set_eq:NN \__datatool_overwrite_action:nn
   \__datatool_save_error_overwrite:nn
```

Hook to run at the start of save/load CSV to allow for local catcode change for TSV format.

```
\tl_new:N \l__datatool_io_csv_init_tl
```

`\ifdtlnoheader` The `noheader` option indicates that the file doesn't have a header row.

```
\newif\ifdtlnoheader
```

`\ifdtlautokeys` Assign the default keys even if a header row is supplied (input only).

```
\newif\ifdtlautokeys
\dtlautokeysfalse
```

`\dtldefaultkey` Default key to use if none specified (column index will be appended).

```
\newcommand*{\dtldefaultkey}{Column}
```

`\ifDTLnewdbonload` [Deprecated] Governs whether or not the database should be defined by `\DTLloaddb` and `\DTLloadrawdb`.

```
\newif\ifDTLnewdbonload
```

Ensure compatibility with previous versions:

```
\DTLnewdbonloadtrue
```

Conditional set by format to indicate whether or not appending to a database is allowed.

```
\bool_new:N \l__datatool_append_allowed_bool
\bool_set_true:N  \l__datatool_append_allowed_bool
```

```
\cs_new:Nn \__datatool_load_init_old_style:n
 {
   \ifDTLnewdbonload
     \__datatool_load_init_create:n { #1 }
   \else
     \__datatool_load_init_append:n { #1 }
   \fi
 }
```

Use old method as the default for backward-compatibility.

```
\cs_set_eq:NN \__datatool_load_init:n
  \__datatool_load_init_old_style:n
```

Error if database already exists with create setting.

```
\cs_new:Nn \__datatool_load_db_exists_error:n
{
  \PackageError { datatool }
   {
     \token_to_str:N \DTLread : ~ Database ~ `#1' ~
     already ~ exists
   }
   {
     Check ~ the ~ `name' ~ setting ~ in ~ the ~ optional ~
     argument ~ of ~ \token_to_str:N \DTLread . ~
     DBTEX ~ v2.0 ~ and ~ DTLTEX ~ v2.0 ~ formats ~ don't ~ allow ~ the ~
     database ~ name ~ to ~ be ~ changed ~ if ~ it's ~
     hard-coded ~ in ~ the ~ file. ~ DBTEX ~ v3.0 ~ and ~
     DTLTEX ~ v3.0 ~ require ~ the ~ new ~ name ~ to ~
     be ~ provided ~ with ~ the ~ `name' ~ option, ~ not ~
     with ~ the ~ `default-name' ~ package-wide ~ setting
   }
}
```

Create database.

```
\cs_new:Nn \__datatool_load_init_create:n
 {
   \cs_set_eq:NN \__datatool_csv_check_key:
     \__datatool_csv_check_key_create:
   \cs_set:Nn \__datatool_csv_check_col_idx: { }
   \cs_set_eq:NN \__datatool_csv_check_col_count:
      \__datatool_csv_check_col_count_create:
   \DTLifdbexists { #1 }
   {
     \__datatool_load_db_exists_error:n { #1 }
   }
   {
    \__datatool_new_db:n { #1 }
   }
 }
\cs_new:Nn \__datatool_load_init_append:n
 {
   \bool_if:NTF \l__datatool_append_allowed_bool
   {
     \cs_set_eq:NN \__datatool_csv_check_key:
       \__datatool_csv_check_key_append:
     \cs_set_eq:NN \__datatool_csv_check_col_idx:
       \__datatool_csv_check_col_idx_append:
     \cs_set:Nn \__datatool_csv_check_col_count: { }
     \dtl@message { Appending ~ to ~  `#1' }
     \int_set:Nn \dtlrownum { \DTLrowcount { #1 } }
   }
   {
     \PackageError {datatool}
```

```
      { Appending ~ not ~ supported ~ for ~ the ~ given ~ format. ~
        Unexpected ~ results ~ may ~ occur}
      {}
    }
  }
 \cs_new:Nn \__datatool_load_init_overwrite:n
  {
    \cs_set_eq:NN \__datatool_csv_check_key:
      \__datatool_csv_check_key_create:
    \cs_set:Nn \__datatool_csv_check_col_idx: { }
    \cs_set_eq:NN \__datatool_csv_check_col_count:
       \__datatool_csv_check_col_count_create:
    \DTLifdbexists { #1 }
    {
      \DTLgcleardb { #1 }
    }
    { \DTLnewdb { #1 } }
  }
 \cs_new:Nn \__datatool_load_init_append_or_create:n
  {
    \DTLifdbexists { #1 }
    {
       \__datatool_load_init_append:n { #1 }
    }
    {
       \__datatool_load_init_create:n { #1 }
    }
  }
```

When appending, need to map the column in the new data being parse to the corresponding column in the database to which the new data is being appended.

```
 \prop_new:N \l__datatool_csv_colnew_to_colold_prop
```

```
 \cs_new:Nn \__datatool_csv_check_key_create:
  {
    \@sDTLifhaskey \l__datatool_io_name_tl \l__datatool_item_key_tl
    {
      \PackageError { datatool }
      { Duplicate ~ key ~ `\l__datatool_item_key_tl' ~ found ~ found }
      { Did ~ you ~ want ~ to ~ append ~ to ~ an ~ existing ~ database? }
    }
    { }
  }
```

Appending so set up map.

```
 \cs_new:Nn \__datatool_csv_check_key_append:
  {
    \@sDTLifhaskey \l__datatool_io_name_tl \l__datatool_item_key_tl
    {
```

Get the existing column index.

```
      \int_set:Nn \l__datatool_tmpa_int
          { \dtlcolumnindex \l__datatool_io_name_tl \l__datatool_item_key_tl }
      \prop_put:NVV
         \l__datatool_csv_colnew_to_colold_prop
         \l__datatool_item_col_tl
         \l__datatool_tmpa_int
    }
    {
       \int_gincr:c { dtlcols@ \l__datatool_io_name_tl }
       \prop_put:NVx
         \l__datatool_csv_colnew_to_colold_prop
         \l__datatool_item_col_tl
         { \int_use:c { dtlcols@ \l__datatool_io_name_tl } }
    }
  }
```

With load-action=create, any extra columns can be detected if `\dtlcolumn` is greater than the column count for the database.

```
\cs_new:Nn \__datatool_csv_check_col_count_create:
{
  \int_compare:nNnT
    { \int_use:c { dtlcols@ \l__datatool_io_name_tl } } < { \dtlcolumnnum}
  {
    \int_gset_eq:cN
       { dtlcols@ \l__datatool_io_name_tl }
       \dtlcolumnnum
  }
}
```

With load-action=append, there's no need to do anything as the index check will increment the column count.

Update column index. Nothing needs to be done with load-action=create.

```
\cs_new:Nn \__datatool_csv_check_col_idx_append:
{
  \prop_get:NVN
    \l__datatool_csv_colnew_to_colold_prop
    \l__datatool_item_col_tl
    \l__datatool_tmpb_tl
  \quark_if_no_value:NTF \l__datatool_tmpb_tl
  {
    \int_gincr:c { dtlcols@ \l__datatool_io_name_tl }
    \prop_put:NVx
      \l__datatool_csv_colnew_to_colold_prop
      \l__datatool_item_col_tl
      { \int_use:c { dtlcols@ \l__datatool_io_name_tl } }
    \tl_set:Nv \l__datatool_item_col_tl
      { \int_use:c { dtlcols@ \l__datatool_io_name_tl } }
  }
  {
    \tl_set_eq:NN \l__datatool_item_col_tl \l__datatool_tmpb_tl
```

```
  }
}
```

Default is for load-action=create

```
\cs_set_eq:NN \__datatool_csv_check_key: \__datatool_csv_check_key_create:
\cs_set_eq:NN \__datatool_csv_check_col_count:
  \__datatool_csv_check_col_count_create:
\cs_new:Nn \__datatool_csv_check_col_idx: { }
```

```
\tl_new:N \l__datatool_io_name_tl
```

`\dtllastloadeddb` This should be set at the end of DTLTEX v2.0 and DBTEX v2.0 files. Initialise to empty in case v1.0 file format loaded.

```
\tl_new:N \dtllastloadeddb
```

Does CSV/TSV source contain LaTeX commands or should TeX characters be considered literally?

```
\bool_new:N \l__datatool_csv_literal_content_bool
\bool_set_true:N \l__datatool_csv_literal_content_bool
```

Version 3.2: Does CSV/TSV source contain plain numbers or should content be parsed?

```
\bool_new:N \l__datatool_csv_plain_content_bool
\bool_set_false:N \l__datatool_csv_plain_content_bool
```

Default keys when reading CSV/TSV files.

```
\prop_new:N \l__datatool_csv_keys_prop
\prop_new:N \l__datatool_csv_headers_prop
\prop_new:N \l__datatool_csv_types_prop
```

Version 3.4: List of column indexes that shouldn't be parsed. (Columns explicitly identified as string in data-types shouldn't have items parsed.)

```
\seq_new:N \l__datatool_no_parse_cols_seq
```

```
\bool_new:N \l__datatool_csv_convert_bool
\bool_set_false:N \l__datatool_csv_convert_bool
```

`\dtl@omitlines` TODO replace with l3int variable?

```
\newcount{\dtl@omitlines}
```

```
\seq_new:N \l__datatool_csv_only_auto_reformat_seq
\cs_new:Nn \__datatool_csv_set_auto_reformat:
 {
  \seq_if_empty:NF \l__datatool_csv_only_auto_reformat_seq
   {
     \seq_if_in:NVTF
       \l__datatool_csv_only_auto_reformat_seq
       \l__datatool_item_col_tl
      {
        \bool_set_true:N \l__datatool_reformat_numeric_bool
        \bool_set_true:N \l__datatool_reformat_datetime_bool
      }
      {
```

```
      \bool_set_false:N \l__datatool_reformat_numeric_bool
      \bool_set_false:N \l__datatool_reformat_datetime_bool
    }
  }
}
```

Options:

```
\keys_define:nn { datatool/io }
{
  keys .code:n =
    {
      \prop_clear_new:N \l__datatool_csv_keys_prop
      \clist_set:Nn \l__datatool_tmpa_clist { #1 }
      \int_zero:N \l__datatool_tmpa_int
      \clist_map_inline:Nn \l__datatool_tmpa_clist
       {
         \int_incr:N \l__datatool_tmpa_int
         \tl_if_empty:nF { ##1 }
         {
           \prop_put:NVn \l__datatool_csv_keys_prop
              \l__datatool_tmpa_int  { ##1 }
         }
       }
      \prop_if_empty:NF \l__datatool_csv_keys_prop
       {
         \dtlautokeysfalse
       }
    },
  headers .code:n =
    {
      \prop_clear_new:N \l__datatool_csv_headers_prop
      \clist_set:Nn \l__datatool_tmpa_clist { #1 }
      \int_zero:N \l__datatool_tmpa_int
      \clist_map_inline:Nn \l__datatool_tmpa_clist
       {
         \int_incr:N \l__datatool_tmpa_int
         \prop_put:NVn \l__datatool_csv_headers_prop
            \l__datatool_tmpa_int  { ##1 }
       }
    },
  data-types .code:n =
    {
      \seq_clear:N \l__datatool_no_parse_cols_seq
      \prop_clear_new:N \l__datatool_csv_types_prop
      \clist_set:Nn \l__datatool_tmpa_clist { #1 }
      \int_zero:N \l__datatool_tmpa_int
      \clist_map_inline:Nn \l__datatool_tmpa_clist
       {
         \int_incr:N \l__datatool_tmpa_int
         \int_if_exist:cTF { c_datatool_ ##1 _int }
```

```
        {
          \prop_put:NVv \l__datatool_csv_types_prop
           \l__datatool_tmpa_int  { c_datatool_ ##1 _int }
          \int_compare:nNnT
            { \int_use:c { c_datatool_ ##1 _int } } = { \c_datatool_string_int }
           {
             \seq_put_right:NV \l__datatool_no_parse_cols_seq
              \l__datatool_tmpa_int
           }
        }
        {
          \PackageError { datatool }
           {
              unknown ~ data ~ type ~ identifier ~ ` ##1 '
           }
           {
              data-types ~ value ~ should ~ be ~ a ~
              comma-separated ~ list ~ of ~ data ~ type ~ identifiers ~
              such ~ as ~ `string' ~ for ~ strings or
              `decimal` for decimals
           }
        }
     }
   },
only-reformat-columns .code:n =
  {
   \seq_set_from_clist:Nn
     \l__datatool_csv_only_auto_reformat_seq { #1 }
  } ,
separator .code:n =
  {
    \DTLsetseparator { #1 }
    \tl_clear:N \l__datatool_io_csv_init_tl
  },
separator .value_required:n = true,
delimiter .code:n =
  {
     \DTLsetdelimiter { #1 }
  },
delimiter .value_required:n = true,
convert-numbers .bool_set:N = \l__datatool_csv_convert_bool ,
trim .bool_set:N = \l__datatool_new_element_trim_bool ,
strict-quotes .choice:,
strict-quotes / true .code:n =
  {
     \PackageWarning { datatool }
      {
        ignoring ~ option ~ `strict-quotes'. ~ This ~ setting ~
        is ~ only ~ supported ~ with ~ datatooltk ~ not ~ with ~
        datatool.sty
```

```
      }
  },
strict-quotes / false .code:n = { },
strict-quotes .default:n = true ,
expand .choice:,
expand .default:n = protected,
expand / none .code:n =
   {
     \cs_set_eq:NN
       \__datatool_save_data:Nn
       \__datatool_save_data_no_expand:Nn
     \cs_set_eq:NN \__datatool_data:n \tl_to_str:n
     \dtlnoexpandnewvalue
   },
expand / protected .code:n =
   {
     \cs_set_eq:NN
       \__datatool_save_data:Nn
       \__datatool_save_data_protected_expand:Nn
     \cs_set_eq:NN \__datatool_data:n \tl_to_str:n
     \dtlexpandnewvalue
   },
expand / full .code:n =
   {
     \cs_set_eq:NN
       \__datatool_save_data:Nn
       \__datatool_save_data_full_expand:Nn
     \cs_set_eq:NN \__datatool_data:n \use:n
     \dtlexpandnewvalue
   },
format .choice:,
format .value_required:n = true,
format / csv .code:n =
   {
     \tl_clear:N \l__datatool_io_csv_init_tl
     \cs_set_eq:NN \__datatool_save:n \__datatool_save_csv:n
     \cs_set_eq:NN \__datatool_load: \__datatool_load_csv:
     \tl_set:Nn \l__datatool_default_ext_tl { .csv }
     \bool_set_true:N  \l__datatool_append_allowed_bool
   },
format / tsv .code:n =
   {
     \tl_set:Nn \l__datatool_io_csv_init_tl
       { \DTLsettabseparator }
     \cs_set_eq:NN \__datatool_save:n \__datatool_save_csv:n
     \cs_set_eq:NN \__datatool_load: \__datatool_load_csv:
     \tl_set:Nn \l__datatool_default_ext_tl { .tsv }
     \bool_set_true:N  \l__datatool_append_allowed_bool
   },
format / dtltex-2 .code:n =
```

```
   {
     \cs_set_eq:NN \__datatool_save:n \__datatool_save_dtltex_ii:n
     \cs_set_eq:NN \__datatool_load: \__datatool_load_tex:
     \tl_set:Nn \l__datatool_default_ext_tl { .dtltex }
     \bool_set_false:N  \l__datatool_append_allowed_bool
   },
format / dtltex-3 .code:n =
   {
     \cs_set_eq:NN \__datatool_save:n \__datatool_save_dtltex_iii:n
     \cs_set_eq:NN \__datatool_load: \__datatool_load_tex:
     \tl_set:Nn \l__datatool_default_ext_tl { .dtltex }
     \bool_set_true:N  \l__datatool_append_allowed_bool
   },
format / dtltex .code:n =
   {
     \cs_set_eq:NN \__datatool_save:n \__datatool_save_dtltex_iii:n
     \cs_set_eq:NN \__datatool_load: \__datatool_load_tex:
     \tl_set:Nn \l__datatool_default_ext_tl { .dtltex }
     \bool_set_true:N  \l__datatool_append_allowed_bool
   },
format / dbtex-2 .code:n =
   {
     \cs_set_eq:NN \__datatool_save:n \__datatool_save_db_ii:n
     \cs_set_eq:NN \__datatool_load: \__datatool_load_tex:
     \tl_set:Nn \l__datatool_default_ext_tl { .dbtex }
     \bool_set_false:N  \l__datatool_append_allowed_bool
   },
format / dbtex-3 .code:n =
   {
     \cs_set_eq:NN \__datatool_save:n \__datatool_save_db_iii:n
     \cs_set_eq:NN \__datatool_load: \__datatool_load_tex:
     \tl_set:Nn \l__datatool_default_ext_tl { .dbtex }
     \bool_set_false:N  \l__datatool_append_allowed_bool
   },
format / dbtex .code:n =
   {
     \cs_set_eq:NN \__datatool_save:n \__datatool_save_db_iii:n
     \cs_set_eq:NN \__datatool_load: \__datatool_load_tex:
     \tl_set:Nn \l__datatool_default_ext_tl { .dbtex }
     \bool_set_false:N  \l__datatool_append_allowed_bool
   },
add-delimiter .choice:,
add-delimiter .value_required:n = true,
add-delimiter / always .code:n =
   {
     \cs_set_eq:NN \__datatool_append_csv_item:n
       \__datatool_append_csv_item_always_delim:n
   },
add-delimiter / detect .code:n =
   {
```

```
      \cs_set_eq:NN \__datatool_append_csv_item:n
        \__datatool_append_csv_item_detect_sep:n
    },
  add-delimiter / never .code:n =
    {
      \cs_set_eq:NN \__datatool_append_csv_item:n
        \__datatool_append_csv_item_no_delim:n
    },
  csv-escape-chars .choice:,
  csv-escape-chars .value_required:n = true,
  csv-escape-chars / none .code:n =
    {
      \cs_set:Nn \__datatool_save_data_postprocess:N { }
      \cs_set:Nn \__datatool_load_data_postprocess:N { }
    },
  csv-escape-chars / delim .code:n =
    {
      \cs_set_eq:NN \__datatool_save_data_postprocess:N
         \__datatool_save_escape_delim:N
      \cs_set_eq:NN \__datatool_load_data_postprocess:N
         \__datatool_load_unescape_delim:N
    },
  csv-escape-chars / delim + bksl .code:n =
    {
      \cs_set_eq:NN \__datatool_save_data_postprocess:N
         \__datatool_save_escape_delim_bksl:N
      \cs_set_eq:NN \__datatool_load_data_postprocess:N
         \__datatool_load_unescape_delim_bksl:N
    },
  csv-escape-chars / double-delim .code:n =
    {
      \cs_set_eq:NN \__datatool_save_data_postprocess:N
        \__datatool_save_double_delim:N
      \cs_set_eq:NN \__datatool_load_data_postprocess:N
         \__datatool_load_unescape_double_delim:N
    },
  csv-content .choice:,
  csv-content / literal .code:n =
    {
      \bool_set_true:N \l__datatool_csv_literal_content_bool
      \bool_set_false:N \l__datatool_csv_plain_content_bool
    },
  csv-content / tex .code:n =
    {
      \bool_set_false:N \l__datatool_csv_literal_content_bool
      \bool_set_false:N \l__datatool_csv_plain_content_bool
    },
```

Version 3.2: lines and cells are read as per csv-content=tex but the cell contents aren't parsed but are assumed to match the data type identified in the data-type option. Any

numerical columns must have plain numbers (no currency symbol, no number group and a decimal point, regardless of localisation).

```
    csv-content / no-parse .code:n =
       {
         \bool_set_false:N \l__datatool_csv_literal_content_bool
         \bool_set_true:N \l__datatool_csv_plain_content_bool
       },
    csv-skip-lines .code:n =
       {
          \tl_if_eq:nnTF { #1 } { false }
          { \int_zero:N \dtl@omitlines }
          {
            \int_compare:nNnTF { #1 } < { 0 }
            {
               \PackageError {datatool}
                {csv-skip-lines ~ value ~ can't ~ be ~ negative}{}
            }
            {
               \int_set:Nn \dtl@omitlines { #1 }
            }
          }
       },
    omitlines .int_set:N = \dtl@omitlines,
    csv-blank .choice:,
    csv-blank / ignore .code:n =
       { \cs_set:Nn \__datatool_csv_blank: { } },
    csv-blank / empty-row .code:n =
       { \cs_set:Nn \__datatool_csv_blank:
          { \int_incr:N \dtlrownum \DTLdbNewRow }
       },
      csv-blank / end .code:n =
       { \cs_set:Nn \__datatool_csv_blank: { \ior_map_break: } },
    overwrite .choice:,
    overwrite .value_required:n = true,
    overwrite / error .code:n =
       {
         \cs_set_eq:NN \__datatool_overwrite_action:nn
            \__datatool_save_error_overwrite:nn
       },
    overwrite / warn .code:n =
       {
         \cs_set_eq:NN \__datatool_overwrite_action:nn
            \__datatool_save_warn_overwrite:nn
       },
    overwrite / allow .code:n =
       {
         \cs_set_eq:NN \__datatool_overwrite_action:nn
            \__datatool_save_allow_overwrite:nn
       },
```

```
    noheader .legacy_if_set:n = dtlnoheader,
    no-header .legacy_if_set:n = dtlnoheader,
    autokeys .legacy_if_set:n = dtlautokeys,
    auto-keys .legacy_if_set:n = dtlautokeys,
    load-action .choice:,
    load-action / create .code:n =
       {
         \cs_set_eq:NN \__datatool_load_init:n
           \__datatool_load_init_create:n
       },
    load-action / append .code:n =
       {
         \cs_set_eq:NN \__datatool_load_init:n
           \__datatool_load_init_append:n
       },
    load-action / overwrite .code:n =
       {
         \cs_set_eq:NN \__datatool_load_init:n
           \__datatool_load_init_overwrite:n
       },
    load-action / detect .code:n =
       {
         \cs_set_eq:NN \__datatool_load_init:n
           \__datatool_load_init_append_or_create:n
       },
    load-action / old-style .code:n =
       {
         \cs_set_eq:NN \__datatool_load_init:n
           \__datatool_load_init_old_style:n
       },
    name .str_set_x:N = \l__datatool_io_name_tl,
  }
```

Allow these options to be set in `\DTLsetup` and remove the separator and delimiter package options.

```
\keys_define:nn { datatool }
{
   io .code:n = { \keys_set:nn { datatool/io } { #1 } },
   display .code:n = { \keys_set:nn { datatool/display } { #1 } },
   separator .undefine:,
   delimiter .undefine:
}
```

`\DTLwrite`

`\DTLwrite[`⟨*options*⟩`]{`⟨*filename*⟩`}`

```
\NewDocumentCommand \DTLwrite { o m }
{
  \group_begin:
```

```
    \IfValueT { #1 } { \keys_set:nn { datatool/io } { #1 } }
    \tl_if_empty:NT \l__datatool_io_name_tl
    {
      \tl_set_eq:NN \l__datatool_io_name_tl \l__datatool_default_dbname_tl
    }
    \DTLifdbexists { \l__datatool_io_name_tl }
    {
      \__datatool_get_filename:n { #2 }
      \file_if_exist:nTF { \l__datatool_name_tl }
      {
         \__datatool_overwrite_action:nn { \l__datatool_name_tl }
         {
           \dtl@message { Writing ~ \l__datatool_name_tl }
           \iow_open:Nn \g__datatool_out_stream { \l__datatool_name_tl }
           \__datatool_save:n { \l__datatool_io_name_tl }
           \iow_close:N \g__datatool_out_stream
         }
      }
      {
         \iow_open:Nn \g__datatool_out_stream { \l__datatool_name_tl }
         \dtl@message { Writing ~ \l__datatool_name_tl }
         \__datatool_save:n { \l__datatool_io_name_tl }
         \iow_close:N \g__datatool_out_stream
      }
    }
    {
      \PackageError {datatool}
       { Can't ~ save ~ database ~ `\l__datatool_io_name_tl': ~
         no ~ such ~ database }
       {}
    }
  \group_end:
}
```

Append default extension if omitted.

```
\tl_new:N \l__datatool_dir_tl
\tl_new:N \l__datatool_name_tl
\tl_new:N \l__datatool_ext_tl
\cs_new:Nn \__datatool_get_filename:n
{
  \file_parse_full_name:nNNN { #1 }
   \l__datatool_dir_tl \l__datatool_name_tl \l__datatool_ext_tl
  \tl_if_empty:NT \l__datatool_ext_tl
   {
     \tl_set_eq:NN \l__datatool_ext_tl \l__datatool_default_ext_tl
   }
  \tl_put_right:NV \l__datatool_name_tl \l__datatool_ext_tl
  \tl_if_empty:NF \l__datatool_dir_tl
   {
     \tl_if_eq:NnTF \l__datatool_dir_tl { / }
```

```
      {
        \tl_put_left:Nn \l__datatool_name_tl { / }
      }
      {
        \tl_put_left:Nx \l__datatool_name_tl { \l__datatool_dir_tl / }
      }
   }
}
```

Wrapper function.

```
\cs_new:Nn \__datatool_write:n
{
  \iow_now:Nn \g__datatool_out_stream { #1 }
}
\cs_generate_variant:Nn \__datatool_write:n { x }
\cs_new:Nn \__datatool_write_wrap:n
{
  \iow_wrap:nnnN
   { #1 }
   { }
   { \dtl@init@write@wrap }
   \__datatool_write:n
}
\cs_generate_variant:Nn \__datatool_write_wrap:n { x }
```

`\dtl@init@write@wrap`

```
\ExplSyntaxOff
\newcommand{\dtl@init@write@wrap}{%
  \def\%{\string\%}%
  \def\\{\string\\}%
  \def\#{\string\#}%
  \def\{{\string\{}%
  \def\}{\string\}}%
  \def\~{\string\~}%
  \def\ {\string\ }%
}
\ExplSyntaxOn
```

Save to CSV file.

```
\cs_new:Nn \__datatool_save_csv:n
{
  \l__datatool_io_csv_init_tl
  \tl_clear:N \l__datatool_row_tl
  \ifdtlnoheader
  \else
    \int_zero:N \l__datatool_col_idx_int
    \dtlforeachkey
     (
       \l__datatool_item_key_tl,
       \l__datatool_item_col_tl,
```

```
        \l__datatool_item_type_tl,
        \l__datatool_item_head_tl
      )
       \in { #1 } \do
     {
       \int_incr:N \l__datatool_col_idx_int
       \tl_if_empty:NTF \l__datatool_item_head_tl
        { \exp_args:NV \__datatool_append_csv_item:n \l__datatool_item_key_tl }
        { \exp_args:NV \__datatool_append_csv_item:n \l__datatool_item_head_tl }
     }
     \__datatool_write:x { \l__datatool_row_tl }
   \fi
   \int_step_inline:nn { \DTLrowcount { #1 } }
    {
      \tl_clear:N \l__datatool_row_tl
      \dtlgetrow { #1 } { ##1 }
      \int_zero:N \l__datatool_col_idx_int
      \dtlforeachkey
       (
         \l__datatool_item_key_tl,
         \l__datatool_item_col_tl,
         \l__datatool_item_type_tl,
         \l__datatool_item_head_tl
       )
       \in { #1 } \do
       {
         \int_incr:N \l__datatool_col_idx_int
         \exp_args:NNx \dtlgetentryfromcurrentrow
           { \l__datatool_item_value_tl }
           { \l__datatool_item_col_tl }
         \__datatool_rm_datum:N \l__datatool_item_value_tl
         \datatool_if_null:NT \l__datatool_item_value_tl
          {
            \tl_clear:N \l__datatool_item_value_tl
          }
         \exp_args:NV \__datatool_append_csv_item:n \l__datatool_item_value_tl
       }
       \__datatool_write:x { \l__datatool_row_tl }
    }
}
```

Append CSV entry to row token list. Only add delimiter if separator detected.

```
\cs_new:Nn \__datatool_append_csv_item_detect_sep:n
{
  \int_compare:nNnF
     { \l__datatool_col_idx_int } = { \c_one_int }
   { \tl_put_right:Nx \l__datatool_row_tl { \@dtl@separator } }
  \__datatool_save_data:Nn \l__datatool_tmpa_tl { #1 }
  \exp_args:NNx
    \tl_if_in:NnTF \l__datatool_tmpa_tl { \token_to_str:N \@dtl@separator }
```

```
    {
      \tl_put_right:Nx \l__datatool_row_tl
        {
          \@dtl@delimiter \l__datatool_tmpa_tl \@dtl@delimiter
        }
    }
    {
      \tl_put_right:Nx \l__datatool_row_tl
        { \l__datatool_tmpa_tl }
    }
}
```

Always add delimiters.

```
\cs_new:Nn \__datatool_append_csv_item_always_delim:n
{
  \int_compare:nNnF
     { \l__datatool_col_idx_int } = { \c_one_int }
    { \tl_put_right:Nx \l__datatool_row_tl { \@dtl@separator } }
  \__datatool_save_data:Nn \l__datatool_tmpa_tl { #1 }
  \tl_put_right:Nx \l__datatool_row_tl
    {
      \@dtl@delimiter \l__datatool_tmpa_tl \@dtl@delimiter
    }
}
```

Don’t add delimiters.

```
\cs_new:Nn \__datatool_append_csv_item_no_delim:n
{
  \int_compare:nNnF
     { \l__datatool_col_idx_int } = { \c_one_int }
    { \tl_put_right:Nx \l__datatool_row_tl { \@dtl@separator } }
  \__datatool_save_data:Nn \l__datatool_tmpa_tl { #1 }
  \tl_put_right:Nx \l__datatool_row_tl
    { \l__datatool_tmpa_tl }
}
```

Default:

```
\cs_set_eq:NN \__datatool_append_csv_item:n
 \__datatool_append_csv_item_detect_sep:n
```

```
\tl_new:N \l__datatool_save_version_comment_tl
\tl_set:Nn \l__datatool_save_version_comment_tl
 {
    Created ~ by ~ datatool ~ \use:c { ver@datatool.sty } ~ on ~
    \cs_if_exist:NTF \DTMnow { \DTMnow } { \today }
 }
```

Save to dtltex v2 file.

```
\cs_new:Nn \__datatool_save_dtltex_ii:n
{
  \__datatool_write:x
    { \c_percent_str \c_space_tl DTLTEX ~ 2.0 ~ \TrackLangEncodingName }
```

```
\__datatool_write:x
  {
    \c_percent_str \c_space_tl \l__datatool_save_version_comment_tl
  }
\__datatool_write:x
  { \token_to_str:N \DTLnewdb { #1 } \c_percent_str }
```

Content:

```
\int_step_inline:nn { \DTLrowcount { #1 } }
 {
   \__datatool_write:x
    { \token_to_str:N \DTLnewrow { #1 } \c_percent_str }
   \dtlgetrow { #1 } { ##1 }
   \dtlforeachkey
    (
      \l__datatool_item_key_tl,
      \l__datatool_item_col_tl,
      \l__datatool_item_type_tl,
      \l__datatool_item_head_tl
    )
    \in { #1 } \do
    {
      \exp_args:NNx \dtlgetentryfromcurrentrow
        { \l__datatool_item_value_tl }
        { \l__datatool_item_col_tl }
      \__datatool_rm_datum:N \l__datatool_item_value_tl
      \datatool_if_null:NF \l__datatool_item_value_tl
       {
         \exp_args:NNo \__datatool_save_data:Nn
            \l__datatool_item_value_tl
           { \l__datatool_item_value_tl }
         \__datatool_write_wrap:x
          {
            \token_to_str:N \DTLnewdbentry { #1 }
             { \l__datatool_item_key_tl }
             { \l__datatool_item_value_tl } \c_percent_str
          }
       }
    }
 }
```

Headers (no check for headers option):

```
\dtlforeachkey
 (
   \l__datatool_item_key_tl,
   \l__datatool_item_col_tl,
   \l__datatool_item_type_tl,
   \l__datatool_item_head_tl
 )
 \in { #1 } \do
 {
```

```
      \exp_args:NNo \__datatool_save_data:Nn \l__datatool_item_head_tl
          { \l__datatool_item_head_tl }
      \__datatool_write:x
       {
          \token_to_str:N \DTLsetheader { #1 }
          { \l__datatool_item_key_tl }
          { \l__datatool_item_head_tl }
          \c_percent_str
       }
    }
```

Trailer:

```
   \__datatool_write:x
     {
        \token_to_str:N \def
        \token_to_str:N \dtllastloadeddb { #1 }
        \c_percent_str
     }
}
```

Save to dtltex v3 file.

```
\cs_new:Nn \__datatool_save_dtltex_iii:n
{
  \__datatool_write:x
    { \c_percent_str \c_space_tl DTLTEX ~ 3.0 ~ \TrackLangEncodingName }
  \__datatool_write:x
    {
      \c_percent_str \c_space_tl \l__datatool_save_version_comment_tl
    }
  \__datatool_write:x
    { \token_to_str:N \DTLdbProvideData { #1 } \c_percent_str }
```

Content:

```
   \int_step_inline:nn { \DTLrowcount { #1 } }
    {
      \__datatool_write:x
       { \token_to_str:N \DTLdbNewRow  }
      \dtlgetrow { #1 } { ##1 }
      \dtlforeachkey
       (
         \l__datatool_item_key_tl,
         \l__datatool_item_col_tl,
         \l__datatool_item_type_tl,
         \l__datatool_item_head_tl
       )
       \in { #1 } \do
       {
         \exp_args:NNx \dtlgetentryfromcurrentrow
           { \l__datatool_item_value_tl }
           { \l__datatool_item_col_tl }
         \__datatool_rm_datum:N \l__datatool_item_value_tl
```

```
        \datatool_if_null:NF \l__datatool_item_value_tl
         {
           \exp_args:NNo \__datatool_save_data:Nn
              \l__datatool_item_value_tl
             { \l__datatool_item_value_tl }
           \__datatool_write_wrap:x
            {
              \token_to_str:N \DTLdbNewEntry
               { \l__datatool_item_key_tl }
               { \l__datatool_item_value_tl } \c_percent_str
            }
         }
      }
   }
```

Headers:

```
  \ifdtlnoheader
  \else
    \dtlforeachkey
     (
       \l__datatool_item_key_tl,
       \l__datatool_item_col_tl,
       \l__datatool_item_type_tl,
       \l__datatool_item_head_tl
     )
     \in { #1 } \do
     {
       \exp_args:NNo \__datatool_save_data:Nn \l__datatool_item_head_tl
           { \l__datatool_item_head_tl }
       \__datatool_write:x
        {
           \token_to_str:N \DTLdbSetHeader
           { \l__datatool_item_key_tl }
           { \l__datatool_item_head_tl }
           \c_percent_str
        }
     }
  \fi
}
```

Save to dbtex v2 file.

```
\cs_new:Nn \__datatool_save_db_ii:n
{
```

Initialise commands:

```
  \cs_set_eq:NN \__datatool_dbtex_row:nn \__datatool_dbtex_row_ii:nn
  \cs_set_eq:NN \__datatool_dbtex_value:nn \__datatool_dbtex_value_ii:nn
  \cs_set_eq:NN \__datatool_dbtex_datum:nnnn \__datatool_dbtex_datum_ii:nnnn
  \cs_set_eq:NN \__datatool_datum:w \__datatool_datum_use_i:w
  \cs_set_eq:NN \__datatool_dbkeys_block:nnnn \__datatool_dbkeys_block_ii:nnnn
  \__datatool_init_dbwrite:
```

Header:

```
\__datatool_write:x
  { \c_percent_str \c_space_tl DBTEX ~ 2.0 ~ \TrackLangEncodingName }
\__datatool_write:x
  {
    \c_percent_str \c_space_tl \l__datatool_save_version_comment_tl
    \iow_newline:
    \token_to_str:N \DTLifdbexists { #1 } \c_percent_str \iow_newline:
    {
       \token_to_str:N \PackageError { datatool }
       { Database ~ `#1' ~ already ~ exists } { }
         \c_percent_str \iow_newline:
       \token_to_str:N \aftergroup
       \token_to_str:N \endinput
    }
    {} \c_percent_str \iow_newline:
    \token_to_str:N \bgroup
    \token_to_str:N \makeatletter
    \iow_newline:
    \token_to_str:N \dtl@message
     { Reconstructing ~ database ~ `#1' }
    \c_percent_str
  }
```

Database Header:

```
\__datatool_write:x
  {
    \token_to_str:N \expandafter
    \token_to_str:N \global
    \token_to_str:N \expandafter
    \iow_newline:
    \token_to_str:N \newtoks
    \token_to_str:N \csname \c_space_tl
      dtlkeys@#1
    \token_to_str:N \endcsname
    \iow_newline:
    \token_to_str:N \expandafter
    \token_to_str:N \global
    \iow_newline:
    \token_to_str:N \csname \c_space_tl
      dtlkeys@#1
    \token_to_str:N \endcsname =
    \iow_char:N \{ \c_percent_str
  }
\exp_args:Nv \__datatool_write:x { dtlkeys@#1 }
\__datatool_write:x { \iow_char:N \} \c_percent_str }
```

Database Content:

```
\__datatool_write:x
  {
```

```
      \token_to_str:N \expandafter
      \token_to_str:N \global
      \token_to_str:N \expandafter
      \iow_newline:
      \token_to_str:N \newtoks
      \token_to_str:N \csname \c_space_tl
        dtldb@#1
      \token_to_str:N \endcsname
      \iow_newline:
      \token_to_str:N \expandafter
      \token_to_str:N \global
      \iow_newline:
      \token_to_str:N \csname \c_space_tl
        dtldb@#1
      \token_to_str:N \endcsname =
      \iow_char:N \{ \c_percent_str
    }
  \exp_args:Nv \__datatool_write_wrap:x { dtldb@#1 }
  \__datatool_write:x { \iow_char:N \} \c_percent_str }
```

Row and column count registers:

```
  \__datatool_write:x
    {
```

Number of Rows :

```
      \token_to_str:N \expandafter
      \token_to_str:N \global \iow_newline: \c_space_tl
      \token_to_str:N \expandafter
      \token_to_str:N \newcount
        \token_to_str:N \csname \c_space_tl
          dtlrows@#1
        \token_to_str:N \endcsname \iow_newline:
      \token_to_str:N \expandafter
      \token_to_str:N \global \iow_newline: \c_space_tl
        \token_to_str:N \csname \c_space_tl
           dtlrows@#1
        \token_to_str:N \endcsname
         = \DTLrowcount { #1 } \token_to_str:N \relax \iow_newline:
```

Number of Columns :

```
      \token_to_str:N \expandafter
      \token_to_str:N \global \iow_newline: \c_space_tl
      \token_to_str:N \expandafter
      \token_to_str:N \newcount
        \token_to_str:N \csname \c_space_tl
          dtlcols@#1
        \token_to_str:N \endcsname \iow_newline:
      \token_to_str:N \expandafter
      \token_to_str:N \global \iow_newline: \c_space_tl
        \token_to_str:N \csname \c_space_tl
           dtlcols@#1
```

```
        \token_to_str:N \endcsname
         = \DTLcolumncount { #1 } \token_to_str:N \relax
    }
```
Column label to index assignments:
```
  \dtlforeachkey
   (
     \l__datatool_item_key_tl,
     \l__datatool_item_col_tl,
     \l__datatool_item_type_tl,
     \l__datatool_item_head_tl
   )
    \in { #1 } \do
  {
    \__datatool_write:x
     {
       \token_to_str:N \expandafter \iow_newline: \c_space_tl
       \token_to_str:N \gdef
         \token_to_str:N \csname \c_space_tl
            dtl@ci@#1@ \l__datatool_item_key_tl
         \token_to_str:N \endcsname
        { \l__datatool_item_col_tl }
       \c_percent_str
     }
  }
```
Trailer:
```
  \__datatool_write:n { \egroup }
  \__datatool_write:x
    {
       \token_to_str:N \def
       \token_to_str:N \dtllastloadeddb { #1 }
       \c_percent_str
    }
}
```
Save to dbtex v3 file.
```
\cs_new:Nn \__datatool_save_db_iii:n
{
```
Initialise commands:
```
  \cs_set_eq:NN \__datatool_dbtex_row:nn \__datatool_dbtex_row_iii:nn
  \cs_set_eq:NN \__datatool_dbtex_value:nn \__datatool_dbtex_value_iii:nn
  \cs_set_eq:NN \__datatool_dbtex_datum:nnnn \__datatool_dbtex_datum_iii:nnnn
  \cs_set_eq:NN \__datatool_datum:w \__datatool_datum_use:w
  \cs_set_eq:NN \__datatool_dbkeys_block:nnnn \__datatool_dbkeys_block_iii:nnnn
  \cs_set_eq:NN \DTLtemporalvalue \__datatool_dbtex_temporal_value:nn
  \__datatool_init_dbwrite:
```
Header:
```
  \__datatool_write:x
    { \c_percent_str \c_space_tl DBTEX ~ 3.0 ~ \TrackLangEncodingName }
```

```
  \__datatool_write:x
    {
      \c_percent_str \c_space_tl \l__datatool_save_version_comment_tl
    }
  \__datatool_write:x
    { \token_to_str:N \DTLdbProvideData { #1 } \c_percent_str }
```

Database Construction:

```
  \__datatool_write:x { \token_to_str:N \DTLreconstructdatabase  }
```

Number of rows and columns:

```
  \__datatool_write:x
    { { \DTLrowcount { #1 } } { \DTLcolumncount { #1 } } }
```

Database Header:

```
  \exp_args:Nv \__datatool_dbkeys_write_iii:n { dtlkeys@#1 }
```

Database Content:

```
  \exp_args:Nv \__datatool_dbtex_content_iii:n { dtldb@#1 }
```

Column label to index assignments:

```
  \__datatool_write:x
    { \iow_char:N \{ \c_percent_str \c_space_tl Key ~ to ~ index }
  \dtlforeachkey
   (
     \l__datatool_item_key_tl,
     \l__datatool_item_col_tl,
     \l__datatool_item_type_tl,
     \l__datatool_item_head_tl
   )
    \in { #1 } \do
  {
    \__datatool_write:x
     {
        \token_to_str:N \dtldbreconstructkeyindex
         { \l__datatool_item_key_tl } { \l__datatool_item_col_tl }
        \c_percent_str
     }
  }
  \__datatool_write:x
    { \iow_char:N \} \c_percent_str \c_space_tl End ~ of ~ key ~ to ~ index }
}
```

When saving the contents of the database in a dbtex file, the datum markers need to be hidden as the LaTeX3 catcode changes won't be available.

DBTeX v2 writes all the internal marker tokens. Row:

```
\cs_new:Nn \__datatool_dbtex_row_ii:nn
 {
    \c_percent_str \iow_newline:
    \c_percent_str \c_space_tl Start ~ of ~ Row ~ #1 \iow_newline:
    \token_to_str:N \db@row@elt@w \c_space_tl
     \c_percent_str \iow_newline:
```

```
    \token_to_str:N \db@row@id@w \c_space_tl #1
      \c_percent_str \iow_newline:
    \token_to_str:N \db@row@id@end@ \c_space_tl
      \c_percent_str \iow_newline:
     #2
    \c_percent_str \c_space_tl End ~ of ~ Row ~ #1 \iow_newline:
    \token_to_str:N \db@row@id@w \c_space_tl #1
      \c_percent_str \iow_newline:
    \token_to_str:N \db@row@id@end@ \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@row@elt@end@ \c_space_tl
  }
```

Column value:

```
 \cs_new:Nn \__datatool_dbtex_value_ii:nn
  {
    \c_percent_str \c_space_tl Column ~ #1 \iow_newline:
    \token_to_str:N \db@col@id@w \c_space_tl #1
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@id@end@ \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@elt@w \c_space_tl
      \tl_if_head_eq_meaning:nNTF { #2 } \__datatool_datum:nnnn
      { #2 }
      {
        \tl_if_head_eq_meaning:nNTF { #2 } \__datatool_datum:w
        { #2 }
        {
          \__datatool_data_value:n { #2 }
        }
      }
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@elt@end@ \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@id@w \c_space_tl #1
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@id@end@ \c_space_tl
      \c_percent_str \c_space_tl End ~ of ~ Column ~ #1 \iow_newline:
  }
```

DBTEX v2 doesn't support datum:

```
 \cs_new:Nn \__datatool_dbtex_datum_ii:nnnn
  {
    \__datatool_data_value:n { #1 }
  }
```

Header block:

```
 \cs_new:Nn \__datatool_dbkeys_block_ii:nnnn
  {
    \c_percent_str \iow_newline:
    \c_percent_str \c_space_tl Header ~ block ~ for ~ column ~ #1 \iow_newline:
```

```
    \token_to_str:N \db@plist@elt@w \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@id@w \c_space_tl #1
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@id@end@ \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@key@id@w \c_space_tl #2
      \c_percent_str \iow_newline:
    \token_to_str:N \db@key@id@end@ \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@type@id@w \c_space_tl
      \__datatool_use_datatype:n { #3 }
      \c_percent_str \iow_newline:
    \token_to_str:N \db@type@id@end@ \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@header@id@w \c_space_tl
      \__datatool_data_value:n { #4 }
      \c_percent_str \iow_newline:
    \token_to_str:N \db@header@id@end@ \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@id@w \c_space_tl #1
      \c_percent_str \iow_newline:
    \token_to_str:N \db@col@id@end@ \c_space_tl
      \c_percent_str \iow_newline:
    \token_to_str:N \db@plist@elt@end@ \c_space_tl
  }
```

DBTeX v3 hides the internal marker tokens.

```
 \cs_new:Nn \__datatool_dbtex_content_iii:n
  {
   \exp_args:Nx \__datatool_write_wrap:n
     {
       { \c_percent_str \c_space_tl Content \iow_newline:
         #1
       } \c_percent_str \c_space_tl End ~ of ~ Content
     }
  }
```

Row:

```
 \cs_new:Nn \__datatool_dbtex_row_iii:nn
  {
    \c_percent_str \c_space_tl Row ~ #1 \iow_newline:
    \token_to_str:N \dtldbrowreconstruct
    { #1 } \c_percent_str
    \iow_newline: {  \c_percent_str \c_space_tl Row ~  #1 ~ Content
    \iow_newline: #2
    }\c_percent_str \iow_newline:
  }
```

Column value:

```
 \cs_new:Nn \__datatool_dbtex_value_iii:nn
```

```
  {
    \c_space_tl \c_space_tl \token_to_str:N \dtldbcolreconstruct
      { #1 } \c_percent_str \c_space_tl Column ~ #1 \iow_newline:
    \c_space_tl \c_space_tl
    {
      \tl_if_head_eq_meaning:nNTF { #2 } \__datatool_datum:nnnn
       { #2 }
       {
         \__datatool_data_value:n { #2 }
       }
    } \c_percent_str \iow_newline:
  }
```

Datum:

```
 \cs_new:Nn \__datatool_dbtex_datum_iii:nnnn
  {
    \token_to_str:N \dtldbdatumreconstruct
     {
       \__datatool_data_value:n { #1 }
     }
     { #2 }
     { \__datatool_data_value:n { #3 } } { \number#4 }
  }
```

Temporal value:

```
 \cs_new:Nn \__datatool_dbtex_temporal_value:nn
  {
    \token_to_str:N \DTLtemporalvalue { #1 } { #2 }
  }
```

Headers:

```
 \cs_new:Nn \__datatool_dbkeys_write_iii:n
  {
   \exp_args:Nx \__datatool_write_wrap:n
     {
       { \c_percent_str \c_space_tl Header \iow_newline:
         #1
       } \c_percent_str \c_space_tl End ~ of ~ Header
     }
  }
```

Header block:

```
 \cs_new:Nn \__datatool_dbkeys_block_iii:nnnn
  {
    \token_to_str:N \dtldbheaderreconstruct
    { #1 } { #2 } { #3 }
    {
      \__datatool_data_value:n { #4 }
    } \c_percent_str \iow_newline:
  }
```

Default.

```
\cs_set_eq:NN \__datatool_dbtex_row:nn \__datatool_dbtex_row_iii:nn
\cs_set_eq:NN \__datatool_dbtex_value:nn \__datatool_dbtex_value_iii:nn
\cs_set_eq:NN \__datatool_dbtex_datum:nnnn \__datatool_dbtex_datum_iii:nnnn
\cs_set_eq:NN \__datatool_dbkeys_block:nnnn \__datatool_dbkeys_block_iii:nnnn

\cs_new:Nn \__datatool_init_dbwrite:
{
```

Header markers:

```
  \def \db@plist@elt@w
    \db@col@id@w ##1 \db@col@id@end@ % ID
    \db@key@id@w ##2 \db@key@id@end@ % label
    \db@type@id@w ##3 \db@type@id@end@ % type
    \db@header@id@w ##4 \db@header@id@end@ % title
    \db@col@id@w ##5 \db@col@id@end@ % ID
    \db@plist@elt@end@
  {
    \__datatool_dbkeys_block:nnnn { ##1 } { ##2 } { ##3 } { ##4 }
  }
```

Content markers:

```
  \def \db@row@elt@w
    \db@row@id@w ##1 \db@row@id@end@
    ##2
    \db@row@id@w ##3 \db@row@id@end@
    \db@row@elt@end@
  {
     \__datatool_dbtex_row:nn { ##1 } { ##2 }
  }
  \def \db@col@id@w ##1 \db@col@id@end@
    \db@col@elt@w ##2 \db@col@elt@end@
    \db@col@id@w ##3 \db@col@id@end@
  {
    \__datatool_dbtex_value:nn { ##1 } { ##2 }
  }
  \cs_set:Nn \__datatool_datum:nnnn
  {
    \__datatool_dbtex_datum:nnnn { ##1 } { ##2 } { ##3 } { ##4 }
  }
}
```

Reconstruction macros.

`\dtldbrowreconstruct`

```
\newcommand{\dtldbrowreconstruct}[2]{
  \__datatool_row_markup_full:nn { #1 } { #2 }
}
```

`\dtldbcolreconstruct` The second argument should be `\dtldbvaluereconstruct` or `\dtldbdatumreconstruct` in which case it can be allowed to expand, but if not prevent any further expansion.

```
\newcommand{\dtldbcolreconstruct}[2]{
```

```
                        \__datatool_row_element_markup_full:nn
                         { #1 }
                         {
                           \tl_if_head_eq_meaning:nNTF
                             { #2 } \dtldbvaluereconstruct
                           { #2 }
                           {
                             \tl_if_head_eq_meaning:nNTF
                                { #2 } \dtldbdatumreconstruct
                              { #2 }
                              { \exp_not:n { #2 } }
                           }
                         }
                      }

\dtldbdatumreconstruct
                      \newcommand{\dtldbdatumreconstruct}[4]{
                        \__datatool_weird_datum:nnnn
                          { #1 } { #2 } { #3 }
                        {
                          \int_case:nnF { #4 }
                          {
                             { \c_datatool_integer_int } { \exp_not:N \c_datatool_integer_int }
                             { \c_datatool_decimal_int } { \exp_not:N \c_datatool_decimal_int }
                             { \c_datatool_currency_int } { \exp_not:N \c_datatool_currency_int }
                             { \c_datatool_datetime_int } { \exp_not:N \c_datatool_datetime_int }
                             { \c_datatool_date_int } { \exp_not:N \c_datatool_date_int }
                             { \c_datatool_time_int } { \exp_not:N \c_datatool_time_int }
                             { \c_datatool_unknown_int } { \exp_not:N \c_datatool_unknown_int }
                          }
                          { \exp_not:N \c_datatool_string_int }
                        }
                      }

\dtldbvaluereconstruct
                      \newcommand{\dtldbvaluereconstruct}[1]
                      {
                        \exp_not:n { #1 }
                      }

\dtldbheaderreconstruct
                      \newcommand{\dtldbheaderreconstruct}[4]
                      {
                        \exp_not:N \db@plist@elt@w
                        \exp_not:N \db@col@id@w #1
                        \exp_not:N \db@col@id@end@
                        \exp_not:N \db@key@id@w #2
                        \exp_not:N \db@key@id@end@
                        \exp_not:N \db@type@id@w #3
```

```
    \exp_not:N \db@type@id@end@
    \exp_not:N \db@header@id@w \exp_not:n { #4 }
    \exp_not:N \db@header@id@end@
    \exp_not:N \db@col@id@w #1
    \exp_not:N \db@col@id@end@
    \exp_not:N \db@plist@elt@end@
  }

dtldbreconstructkeyindex

  \newcommand{\dtldbreconstructkeyindex}[2]
  {
    \csgdef { dtl@ci@\dtllastloadeddb @ #1 } { #2 }
  }
```

\dtl@reconstruct@data{⟨*header*⟩}{⟨*content*⟩}{⟨*num rows*⟩}
{⟨*num cols*⟩}
{⟨*mappings*⟩}

\dtl@reconstruct@data

```
  \newcommand{\dtl@reconstruct@data}[5]{
    \exp_args:Nxx \@dtl@reconstruct@data { #1 } {#2 }
      { #3 } { #4 } { #5 } { \dtllastloadeddb }
  }
  \newcommand{\@dtl@reconstruct@data}[6]{
```

Number of rows:

```
    \expandafter\global
     \expandafter\newcount\csname dtlrows@#6\endcsname
    \expandafter\global
     \csname dtlrows@#6\endcsname=#3\relax
```

Number of columns:

```
    \expandafter\global
     \expandafter\newcount\csname dtlcols@#6\endcsname
    \expandafter\global
     \csname dtlcols@#6\endcsname=#4\relax
```

Header:

```
    \expandafter\global\expandafter
    \newtoks\csname dtlkeys@#6\endcsname
    \__datatool_token_register_gset:cn { dtlkeys@#6 } { #1 }
```

Content:

```
    \expandafter\global\expandafter
       \newtoks\csname dtldb@#6\endcsname
    \__datatool_token_register_gset:cn { dtldb@#6 } { #2 }
```

Column label to index assignments:

```
    #5
  }

  \ExplSyntaxOff
```

12.11.1 Deprecated Save Functions

`\@dtl@write` Used with the older functions, which are now deprecated, so `\@dtl@write` should only be defined if the older functions are used.

`\@dtl@def@write`

```
\newcommand{\@dtl@def@write}{%
  \newwrite\@dtl@write
  \let\@dtl@def@write\relax
}
```

`\DTLsavedb`

```
\DTLsavedb{⟨db name⟩}{⟨filename⟩}
```

Save a database as an ASCII data file using the separator and delimiter given by `\@dtl@separator` and `\@dtl@delimiter`.

```
\newcommand*{\DTLsavedb}[2]{%
  \DTLwrite[name={#1},overwrite=warn,format=csv,expand=none,add-
delimiter=detect]{#2}%
}
```

`\DTLsavetexdb`

```
\DTLsavetexdb{⟨db name⟩}{⟨filename⟩}
```

Save a database as a LaTeX file.

```
\newcommand*{\DTLsavetexdb}[2]{%
  \DTLwrite[name={#1},overwrite=warn,format=dtltex-2,expand=full]{#2}%
}
```

`\DTLsaverawdb`

```
\newcommand*{\DTLsaverawdb}[2]{%
  \DTLwrite[name={#1},overwrite=warn,format=dbtex-2,expand=full]{#2}%
}
```

`\DTLprotectedsaverawdb`

```
\newcommand*{\DTLprotectedsaverawdb}[2]{%
  \DTLwrite[name={#1},overwrite=warn,format=dbtex-2,expand=none]{#2}%
}
```

12.12 Loading a database from an external file

```
\ExplSyntaxOn
```

`\DTLdbProvideData` Default definition changes the default-name option and defines `\dtllastloadeddb`. This is in case the file is simply `\input` instead of using `\DTLread`.

```
\newcommand{\DTLdbProvideData}[1]{%
  \tl_set:Nn \l__datatool_default_dbname_tl { #1 }
  \tl_set_eq:NN \dtllastloadeddb \l__datatool_default_dbname_tl
```

```
  \DTLifdbexists { #1 } { } { \DTLnewdb { #1 } }
}
```

`\DTLreconstructdata` Deprecated. Experimental version that changes category code of `@` which is best avoided. TODO remove when a new rollback is provided after v3.0.

```
\newcommand{\DTLreconstructdata}{%
  \PackageWarning { datatool }
  {
    Experimental ~ command ~
    \token_to_str:N \DTLreconstructdata \c_space_tl ~
    deprecated. ~ Re-save ~ file
  }
  \DTLifdbexists{\dtllastloadeddb}%
  {%
    \PackageError{datatool}{Database ~ `\dtllastloadeddb' ~ already ~ exists}{}%
    \use_none:nnnn
  }%
  {%
    \bgroup\makeatletter
    \let
     \dtl@db@header@reconstruct
     \dtldbheaderreconstruct
    \let
     \dtl@db@row@reconstruct
     \dtldbrowreconstruct
    \def
       \dtl@db@col@reconstruct
       ##1 ##2
     {
      \__datatool_row_element_markup_full:nn
       { ##1 }
       {
         \tl_if_head_eq_meaning:nNTF
           { ##2 } \dtl@db@value@reconstruct
         { ##2 }
         {
           \tl_if_head_eq_meaning:nNTF
             { ##2 } \dtl@db@datum@reconstruct
             { ##2 }
            { \exp_not:n { ##2 } }
         }
       }
     }
    \cs_set_eq:NN
     \dtl@db@datum@reconstruct
     \dtldbdatumreconstruct
    \cs_set_eq:NN
     \dtl@db@value@reconstruct
     \dtldbvaluereconstruct
    \cs_set_eq:NN
```

```
      \dtl@db@reconstruct@keyindex
      \dtldbreconstructkeyindex
    \@DTLreconstructdata
  }%
}
```

\@DTLreconstructdata

```
\newcommand{\@DTLreconstructdata}[5]{%
  \dtl@message{Reconstructing ~ database ~ `\dtllastloadeddb'}%
  \dtl@reconstruct@data{#1}{#2}{#3}{#4}{#5}%
  \egroup
}
```

\DTLreconstructdbdata

\DTLreconstructdbdata{⟨*header*⟩}{⟨*content*⟩}{⟨*rows*⟩}{⟨*cols*⟩}{⟨*index*⟩}

Deprecated. Experimental command dropped in favour of \DTLreconstructdatabase, which has the number of rows and columns at the start.

```
\newcommand{\DTLreconstructdbdata}[5]{
  \DTLifdbexists {\dtllastloadeddb}
  {
    \PackageError{datatool}{Database ~ `\dtllastloadeddb' ~ already ~ exists}{}
  }
  {
    \dtl@message{Reconstructing ~ database ~ `\dtllastloadeddb'}
    \dtl@reconstruct@data{#1}{#2}{#3}{#4}{#5}%
  }
}
```

\DTLreconstructdatabase

\DTLreconstructdatabase{⟨*rows*⟩}{⟨*cols*⟩}{⟨*header*⟩}{⟨*content*⟩}{⟨*index*⟩}

This has the number of rows and columns in the first two arguments, which makes it easier for datatooltk to parse.

```
\newcommand{\DTLreconstructdatabase}[5]{
  \DTLifdbexists {\dtllastloadeddb}
  {
    \PackageError{datatool}{Database ~ `\dtllastloadeddb' ~ already ~ exists}{}
  }
  {
    \dtl@message{Reconstructing ~ database ~ `\dtllastloadeddb'}
    \dtl@reconstruct@data{#3}{#4}{#1}{#2}{#5}%
  }
}
```

Behaviour of \DTLdbProvideData within \DTLread:

```
\cs_new:Nn \__datatool_provide_data:n
{
  \tl_if_empty:NT \l__datatool_io_name_tl
  {
    \tl_set:Nn \l__datatool_io_name_tl { #1 }
  }
  \tl_set_eq:NN \dtllastloadeddb \l__datatool_io_name_tl
  \tl_set_eq:NN \l__datatool_default_dbname_tl \l__datatool_io_name_tl
```

Don't create the database for the dbtex formats as the database internals are created either explicitly (dbtex v2.0) or via `\DTLreconstructdatabase`.

```
  \tl_if_eq:NnF \l__datatool_default_ext_tl { .dbtex }
   { \__datatool_load_init:n { \dtllastloadeddb } }
}
```

Behaviour of `\@sDTLsetheader` within `\DTLread` if noheader on:

```
\cs_new:Nn \__datatool_set_no_header:nnn
{
  \@sDTLifhaskey { #1 } { #2 }
  {
    \int_set:Nn \dtlcolumnnum { \dtlcolumnindex { #1 } { #2 } }
    \prop_get:NVN \l__datatool_csv_headers_prop \dtlcolumnnum
     \l__datatool_tmpb_tl
    \quark_if_no_value:NTF \l__datatool_tmpb_tl
    {
      \dtl@message{Ignoring ~ header ~ for ~ column ~ `#2' ~ in ~
        database ~ `#1' ~ (no-header=true ~ and ~ no ~ corresponding ~
        key ~ in ~ headers ~ option)}
    }
    {
      \tl_set_eq:NN \l__datatool_item_head_tl \l__datatool_tmpb_tl
      \exp_args:Nnnx \@dtl@setheaderforindex { #1 } { \dtlcolumnnum }
        { \exp_not:o { \l__datatool_item_head_tl } }
    }
  }
  {
    \dtl@message{Ignoring ~ header ~ for ~ column ~ `#2' ~ in ~
      database ~ `#1' ~ (no-header=true ~ and ~ no ~ column ~ with ~ that ~ key)}
  }
}
```

`\DTLread` `\DTLread[`⟨*options*⟩`]{`⟨*file*⟩`}`

Reads a database from the given file according to the format option.

```
\NewDocumentCommand \DTLread { o m }
{
  \group_begin:
    \cs_set_eq:NN \DTLdbProvideData \__datatool_provide_data:n
    \IfValueT { #1 } { \keys_set:nn { datatool/io } { #1 } }
    \bool_set_true:N \l__datatool_db_global_bool
```

```
    \ifdtlnoheader
     \cs_set_eq:NN \@sDTLsetheader \__datatool_set_no_header:nnn
    \fi
    \__datatool_get_filename:n { #2 }
    \file_if_exist:nTF { \l__datatool_name_tl }
    {
      \__datatool_load:
      \tl_if_eq:NNF
        \l__datatool_io_name_tl \dtllastloadeddb
      {
        \PackageWarning { datatool }
         { Database ~ name ~ `\l__datatool_io_name_tl' ~ requested ~
          but ~ name ~ hardcoded ~ in ~ file ~ as ~ ` \dtllastloadeddb '}
      }
    }
    {
      \PackageError {datatool}
       { Can't ~ open ~ file ~ `\l__datatool_name_tl' }
       {}
    }
    \global\let\dtllastloadeddb\dtllastloadeddb
  \group_end:
}
\int_new:N \l__datatool_line_int
```

Load CSV/TSV file:

```
\cs_new:Nn \__datatool_load_csv:
{
  \tl_if_empty:NT \l__datatool_io_name_tl
  {
    \tl_set_eq:NN \l__datatool_io_name_tl \l__datatool_default_dbname_tl
  }
  \int_zero:N \dtlrownum
  \__datatool_load_init:n { \l__datatool_io_name_tl }
  \int_zero:N \l__datatool_line_int
  \l__datatool_io_csv_init_tl
  \exp_args:NV \char_set_catcode_other:N \@dtl@delimiter
  \exp_args:NV \char_set_catcode_other:N \@dtl@separator
  \char_set_catcode_other:n { `\% }
  \ior_open:Nn \g__datatool_in_stream { \l__datatool_name_tl }
  \bool_if:NTF \l__datatool_csv_literal_content_bool
  {
    \ior_str_map_inline:Nn \g__datatool_in_stream
      {
        \__datatool_parse_csv_line:nN { ##1 }
          \__datatool_parse_str_csv_process_item:n
      }
  }
  {
    \ior_map_inline:Nn \g__datatool_in_stream
```

```
      {
        \__datatool_parse_csv_line:nN { ##1 }
          \__datatool_parse_csv_process_item:n
      }
  }
```

Headers need setting now that the data type has been found.

```
  \int_step_function:nN
   { \int_use:c { dtlcols@ \l__datatool_io_name_tl } }
    \__datatool_parse_csv_add_header:n
```

Close stream.

```
  \ior_close:N \g__datatool_in_stream
  \tl_set_eq:NN \dtllastloadeddb \l__datatool_io_name_tl
}
```

Add column header but check first if the key is already defined in case of load-action-append:

```
\cs_new:Nn \__datatool_parse_csv_add_header:n
{
  \prop_get:NnN
    \l__datatool_csv_types_prop { #1 }
    \l__datatool_tmpb_tl
  \quark_if_no_value:NTF \l__datatool_tmpb_tl
  {
    \tl_set:NV \l__datatool_item_type_tl \c_datatool_string_int
  }
  {
    \tl_set_eq:NN \l__datatool_item_type_tl \l__datatool_tmpb_tl
  }
```

Has a key has been provided for this column?

```
  \prop_get:NnN
      \l__datatool_csv_keys_prop { #1 }
      \l__datatool_tmpb_tl
  \quark_if_no_value:NTF \l__datatool_tmpb_tl
  {
```

No key has been provided for this column, so create a default.

```
     \tl_set:Nx \l__datatool_item_key_tl
        { \dtldefaultkey #1 }
  }
  {
```

A key has been provided for this column, so use that.

```
    \tl_set_eq:NN \l__datatool_item_key_tl \l__datatool_tmpb_tl
  }
  \@DTLifhaskey { \l__datatool_io_name_tl } \l__datatool_item_key_tl
  { }
  {
```

Has a header has been provided for this column?

```
    \prop_get:NnN \l__datatool_csv_headers_prop { #1 }
      \l__datatool_tmpb_tl
    \quark_if_no_value:NTF \l__datatool_tmpb_tl
    {
```
No header has been provided for this column, so use the key.
```
     \tl_set_eq:NN \l__datatool_item_head_tl \l__datatool_item_key_tl
    }
    {
      \tl_set_eq:NN \l__datatool_item_head_tl \l__datatool_tmpb_tl
    }
    \tl_if_empty:NTF \l__datatool_item_type_tl
     {
       \int_set_eq:NN \l__datatool_item_type_int
        \c_datatool_unknown_int
     }
     {
       \int_set:Nn \l__datatool_item_type_int
        { \l__datatool_item_type_tl }
     }
    \__datatool_token_register_gput_right:cx
      { dtlkeys@ \l__datatool_io_name_tl }
    {
      \__datatool_column_markup:nVVV
       { #1 }
       \l__datatool_item_key_tl
       \l__datatool_item_type_int
       \l__datatool_item_head_tl
    }
```
Define the key to column index mapping:
```
    \tl_gclear_new:c
      { dtl@ci@ \l__datatool_io_name_tl @ \l__datatool_item_key_tl }
    \tl_gset:cx
      { dtl@ci@ \l__datatool_io_name_tl @ \l__datatool_item_key_tl }
      { #1 }
  }
}
```
Action for blank line:
```
\cs_new:Nn \__datatool_csv_blank: { }
```
The literal setting is awkward as letters still need to have the letter category code (or be active UTF-8).
```
\cs_new:Nn \__datatool_parse_csv_process_item:n
{
```
TODO replace with `\__datatool_process_new_value:n` ?
```
   \@dtl@setnewvalue { #1 }
   \tl_set:NV \l__datatool_item_value_tl \@dtl@toks
   \__datatool_load_data_postprocess:N \l__datatool_item_value_tl
}
```

```
\tl_new:N \l_datatool_str_csv_regex_cases_tl
\tl_set:Nn \l_datatool_str_csv_regex_cases_tl
 {
```

replace `\n`, `\r` and `\f` with a space

```
    { \x { 5C } [ nrf ] } { ~ }
```

replace `\t` with a tab character

```
    { \x { 5C } t } { \t }
```

replace `\` with `\textbackslash␣`

```
    { \cO(\x{5C}) } { \x{ 5C } textbackslash \cS\x{20} }
```

replace special character ⟨*c*⟩ with `\`⟨*c*⟩ for # $ % & _ { and }

```
    { \cO(\x{23}|\x{24}|\x{25}|\x{26}|\x{5F}|\x{7B}|\x{7D}) } { \x{ 5C } \1 }
```

replace `^` with `\textasciicircum␣`

```
    { \cO(\x{5E}) } { \x{ 5C } textasciicircum \cS\x{20} }
```

replace `~` with `\textasciitilde␣`

```
    { \cO(\x{7E}) } { \x{ 5C } textasciitilde \cS\x{20} }
 }
\cs_new:Nn \__datatool_parse_str_csv_process_item:n
{
   \tl_set:Nn \l__datatool_item_value_tl { #1 }
   \__datatool_load_data_postprocess:N \l__datatool_item_value_tl
   \exp_args:NV \regex_replace_case_all:nN
     \l_datatool_str_csv_regex_cases_tl
     \l__datatool_item_value_tl
   \exp_args:NNnV \tl_set_rescan:Nnn \l__datatool_item_value_tl { }
     \l__datatool_item_value_tl
   \dtl@domappings \l__datatool_item_value_tl
}
```

Test for blank line. Note that a blank line may be represented as a series of separator characters without anything else. This is checked for by the regex. This command may be redefined to omit the regular expression match if not appropriate, but the regular expression variable is private as it's redefined by `\DTLsetseparator`.

```
\cs_new:Nn \datatool_if_blank_line:nTF
{
  \tl_if_blank:nTF { #1 }
  { #2 }
  {
    \tl_if_eq:nnTF { #1 } { \par }
     { #2 }
     {
       \regex_match:NnTF \l__datatool_blank_row_regex { #1 }
        { #2 } { #3 }
     }
  }
}
```

```
\cs_new:Nn \__datatool_parse_csv_line:nN
{
  \int_incr:N \l__datatool_line_int
  \int_compare:nNnT
     { \l__datatool_line_int } > { \dtl@omitlines }
  {
    \datatool_if_blank_line:nTF { #1 }
    {
      \__datatool_csv_blank:
    }
    {
      \tl_set:Nn \l__datatool_row_tl { #1 }
```

Split the line on the separator.

```
      \__datatool_split_line:
      \int_zero:N \dtlcolumnnum
      \dtlcurrentrow = { }
```

Iterate over each item in the row.

```
      \seq_map_inline:Nn \l__datatool_row_seq
      {
        \int_incr:N \dtlcolumnnum
        \tl_set:Nx \l__datatool_item_col_tl
          { \int_use:N \dtlcolumnnum }
        #2 { ##1 }
        \legacy_if:nTF { dtlnoheader }
        {
```

Row of data.

```
          \__datatool_csv_set_auto_reformat:
          \bool_if:NTF \l__datatool_csv_plain_content_bool
           {
             \__datatool_no_parse_csv_item:
           }
           {
             \__datatool_parse_csv_item:
           }
          \__datatool_csv_check_col_idx:
```

Append this item to the current row token register. This only needs a local assignment.

```
          \__datatool_token_register_put_right:Nx
            \dtlcurrentrow
            {
              \__datatool_row_element_markup:VV
                \l__datatool_item_col_tl
                \l__datatool_item_value_tl
            }
        }
        {
```

Header row. Get the column label (key).

```
          \legacy_if:nTF { dtlautokeys }
```

```
        {
```
The autokeys setting is on so generate the default key.
```
          \tl_set:Nx \l__datatool_item_key_tl
             { \dtldefaultkey \l__datatool_item_col_tl }
          \prop_put:NVV \l__datatool_csv_keys_prop
              \l__datatool_item_col_tl \l__datatool_item_key_tl
        }
        {
          \prop_get:NVN
            \l__datatool_csv_keys_prop \l__datatool_item_col_tl
            \l__datatool_tmpb_tl
          \quark_if_no_value:NTF \l__datatool_tmpb_tl
           {
```
No key has been provided and autokeys is off so use the header as the key.
```
             \tl_set:Nx \l__datatool_item_key_tl
               { \tl_to_str:V \l__datatool_item_value_tl }
             \prop_put:NVV \l__datatool_csv_keys_prop
                \l__datatool_item_col_tl \l__datatool_item_key_tl
           }
           {
             \tl_set_eq:NN
               \l__datatool_item_key_tl
               \l__datatool_tmpb_tl
           }
        }
```
If a key already exists for this column then either we're appending or there's a duplicate key.
```
        \__datatool_csv_check_key:
```
Has a header been provided?
```
        \prop_if_in:NVF
           \l__datatool_csv_headers_prop
           \l__datatool_item_col_tl
         {
```
A header has not been provided for this column so use the value found on this line of the file.
```
           \prop_put:NVV
             \l__datatool_csv_headers_prop
             \l__datatool_item_col_tl
             \l__datatool_item_value_tl
         }
```
Unless data-types has been used, no information is currently available about the data type so header information needs to be saved (in the property map) and set later.
```
      }
    }
    \legacy_if:nTF { dtlnoheader }
    {
```

Globally append this row of data to the database token register.

```
          \int_incr:N \dtlrownum
          \__datatool_token_register_gput_right:cx
            { dtldb@ \l__datatool_io_name_tl }
            {
              \__datatool_row_markup:VV
                \dtlrownum
                \dtlcurrentrow
            }
```

Globally increment the row count.

```
          \int_gincr:c
             { dtlrows@ \l__datatool_io_name_tl }
       }
       {
```

Header has been found. The next row will be data.

```
          \legacy_if_set_true:n { dtlnoheader }
       }
```

If the number of columns is greater than the current column count for the database, update it.

```
       \__datatool_csv_check_col_count:
     }
   }
 }
```

Version 3.2: function for parsing CSV item:

```
\cs_new:Nn \__datatool_parse_csv_item:
 {
```

Check if this column shouldn't be parsed:

```
    \seq_if_in:NVTF
      \l__datatool_no_parse_cols_seq
      \l__datatool_item_col_tl
     {
       \tl_if_empty:NTF \l__datatool_item_value_tl
        { \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int }
        { \int_set_eq:NN \@dtl@datatype \c_datatool_string_int }
     }
     {
```

Check if this column should be converted:

```
      \bool_if:NTF \l__datatool_csv_convert_bool
       {
```

Convert if number.

```
          \prop_get:NVN
            \l__datatool_csv_types_prop \l__datatool_item_col_tl
            \l__datatool_tmpb_tl
           \quark_if_no_value:NTF \l__datatool_tmpb_tl
            {
```

```
            \int_set_eq:NN \@dtl@datatype \c_datatool_unknown_int
          }
          {
            \int_set:Nn \@dtl@datatype { \l__datatool_tmpb_tl }
          }
         \datatool_if_temporal_datum_type:nTF { \@dtl@datatype }
          {
            \__datatool_parse:NV
             \l__datatool_tmpa_tl
             \l__datatool_item_value_tl
          }
          {
            \datatool_if_numeric_datum_type:nTF { \@dtl@datatype }
             {
               \int_compare:nNnTF
                 { \@dtl@datatype } = { \c_datatool_currency_int }
                {
                  \exp_args:NV \DTLdecimaltocurrency
                   \l__datatool_item_value_tl
                   \l__datatool_tmpa_tl
                }
                {
                  \exp_args:NV \DTLdecimaltolocale
                   \l__datatool_item_value_tl
                   \l__datatool_tmpa_tl
                }
             }
             {
               \__datatool_parse:NV
                \l__datatool_tmpa_tl
                \l__datatool_item_value_tl
             }
          }
        \int_set:Nn \@dtl@datatype { \DTLdatumtype \l__datatool_tmpa_tl }
        \bool_if:NF \l__datatool_db_store_datum_bool
         {
            \tl_set:Ne \l__datatool_item_value_tl
               { \l__datatool_tmpa_tl }
         }
      }
      {
```

Parse to determine data type.

```
        \__datatool_parse:NV
         \l__datatool_tmpa_tl
         \l__datatool_item_value_tl
      }
    }
```

If store-datum option on, set the current item to its datum representation.

```
  \bool_if:NT \l__datatool_db_store_datum_bool
```

```
    {
```
Convert to weird datum.
```
      \__datatool_to_weird_datum:N \l__datatool_tmpa_tl
      \tl_set_eq:NN
        \l__datatool_item_value_tl
        \l__datatool_tmpa_tl
    }
```
If the data type was determined (that is, the item isn't empty), update the column metadata if applicable.
```
    \int_compare:nNnF
      { \@dtl@datatype } = { \c_datatool_unknown_int }
    {
      \prop_get:NVN
        \l__datatool_csv_types_prop \l__datatool_item_col_tl
        \l__datatool_tmpb_tl
      \quark_if_no_value:NTF \l__datatool_tmpb_tl
       {
         \prop_put:NVx
           \l__datatool_csv_types_prop
           \l__datatool_item_col_tl
           { \number\@dtl@datatype }
       }
       {
         \tl_set_eq:NN
           \l__datatool_item_type_tl
           \l__datatool_tmpb_tl
         \int_compare:nNnT
           { \@dtl@datatype } > { \l__datatool_item_type_tl }
         {
           \prop_put:NVx
             \l__datatool_csv_types_prop \l__datatool_item_col_tl
             { \number\@dtl@datatype }
         }
       }
    }
  }
```
Just for csv-content=no-parse:
```
 \cs_new:Nn \__datatool_no_parse_csv_item:
  {
```
Assume string if no information provided in data-types:
```
    \int_set_eq:NN
      \@dtl@datatype
      \c_datatool_string_int
    \prop_if_empty:NTF \l__datatool_csv_types_prop
     {
       \prop_put:NVx
         \l__datatool_csv_types_prop \l__datatool_item_col_tl
```

```
      { \number\@dtl@datatype }
    }
    {
      \prop_get:NVN
        \l__datatool_csv_types_prop
        \l__datatool_item_col_tl
        \l__datatool_tmpb_tl
     \quark_if_no_value:NTF \l__datatool_tmpb_tl
      {
```

Assume same as the final data type unless this is the first column:

```
        \prop_get:NeN
          \l__datatool_csv_types_prop
          { \int_eval:n { \l__datatool_item_col_tl - \c_one_int } }
          \l__datatool_tmpb_tl
        \quark_if_no_value:NF \l__datatool_tmpb_tl
         {
           \int_set:Nn \@dtl@datatype { \l__datatool_tmpb_tl }
         }
        \prop_put:NVx
          \l__datatool_csv_types_prop \l__datatool_item_col_tl
          { \number\@dtl@datatype }
      }
      {
        \int_set:Nn \@dtl@datatype { \l__datatool_tmpb_tl }
      }
    }
  \tl_set_eq:NN
    \l__datatool_datum_original_value_tl
    \l__datatool_item_value_tl
  \tl_clear:N \l__datatool_datum_value_tl
  \tl_clear:N \l__datatool_datum_currency_tl
  \datatool_if_numeric_datum_type:nT { \@dtl@datatype }
   {
     \tl_set_eq:NN
      \l__datatool_datum_value_tl
      \l__datatool_item_value_tl
```

Column has been identified with a numeric type. Does it need converting?

```
      \bool_if:NTF \l__datatool_csv_convert_bool
       {
         \int_case:nn { \@dtl@datatype }
          {
            { \c_datatool_integer_int }
             {
               \exp_args:NV \DTLdecimaltolocale
                \l__datatool_datum_original_value_tl
                \l__datatool_item_value_tl
             }
            { \c_datatool_decimal_int }
             {
```

```
            \exp_args:NV \DTLdecimaltolocale
             \l__datatool_datum_original_value_tl
             \l__datatool_item_value_tl
           }
         { \c_datatool_currency_int }
           {
             \exp_args:NV \DTLdecimaltocurrency
              \l__datatool_datum_original_value_tl
              \l__datatool_item_value_tl
           }
         { \c_datatool_datetime_int }
           {
             \datatool_decimal_to_temporal:NnV
               \l__datatool_item_value_tl
               { \c_datatool_datetime_int }
               \l__datatool_datum_original_value_tl
           }
         { \c_datatool_date_int }
           {
             \datatool_decimal_to_temporal:NnV
               \l__datatool_item_value_tl
               { \c_datatool_date_int }
               \l__datatool_datum_original_value_tl
           }
         { \c_datatool_time_int }
           {
             \datatool_decimal_to_temporal:NnV
               \l__datatool_item_value_tl
               { \c_datatool_time_int }
               \l__datatool_datum_original_value_tl
           }
       }
```

Bit strange to want conversion but not store as datum but allow it anyway.

```
      \bool_if:NF \l__datatool_db_store_datum_bool
       {
         \exp_args:NV \tl_if_head_eq_meaning:nNF
           \l__datatool_item_value_tl
           \__datatool_datum:nnnn
         {
           \tl_set:Ne \l__datatool_item_value_tl
            { \l__datatool_item_value_tl }
         }
       }
    }
    {
      \int_compare:nNnT { \@dtl@datatype } = { \c_datatool_currency_int }
       {
```

No conversion requested but markup currency.

```
         \tl_set:Nn
```

```
              \l__datatool_datum_currency_tl
              { \DTLcurr { \DTLCurrencyCode } }
            \tl_set:Ne \l__datatool_item_value_tl
              {
                \exp_not:N \DTLcurrency
                  { \exp_not:V \l__datatool_datum_value_tl }
              }
            \tl_set_eq:NN
              \l__datatool_datum_original_value_tl
              \l__datatool_item_value_tl
          }
        }
    }
```

If store-datum option on, set the current item to its datum representation.

```
    \bool_if:NT \l__datatool_db_store_datum_bool
     {
      \exp_args:NV \tl_if_head_eq_meaning:nNF
         \l__datatool_item_value_tl
          \__datatool_datum:nnnn
       {
         \tl_set:Ne \l__datatool_item_value_tl
          {
            \exp_not:N \__datatool_datum:nnnn
            { \exp_not:V \l__datatool_datum_original_value_tl }
            { \l__datatool_datum_value_tl }
            { \exp_not:V \l__datatool_datum_currency_tl }
            { \int_use:N \@dtl@datatype }
         }
       }
```

Convert to weird datum.

```
       \__datatool_to_weird_datum_no_parse:N \l__datatool_item_value_tl
    }
 }
```

Splitting is awkward because the delimiter needs to be taken into account.

```
\cs_new:Nn \__datatool_split_line:n
{
  \tl_set:Nn \l__datatool_row_tl { #1 }
  \__datatool_split_line:
}
\seq_new:N \l__datatool_row_seq

\cs_new:Nn \__datatool_split_line:
{
  \exp_args:NNV \seq_set_split_keep_spaces:NnV \l__datatool_tmp_seq
    \@dtl@separator  \l__datatool_row_tl
  \seq_clear:N \l__datatool_row_seq
  \tl_set:Nn \l__datatool_item_value_tl { \q_no_value }
  \__datatool_map_row_seq:
```

```
  \quark_if_no_value:NF \l__datatool_item_value_tl
  {
    \seq_put_right:NV \l__datatool_row_seq \l__datatool_item_value_tl
  }
}
\cs_new:Nn \__datatool_map_row_seq:
{
  \seq_pop_left:NN \l__datatool_tmp_seq \l__datatool_tmpb_tl
  \quark_if_no_value:NF \l__datatool_tmpb_tl
  {
    \quark_if_no_value:NTF \l__datatool_item_value_tl
    {
       \regex_replace_once:NnNTF \l__datatool_delim_both_regex { \1 }
         \l__datatool_tmpb_tl
       {
          \seq_put_right:NV \l__datatool_row_seq \l__datatool_tmpb_tl
       }
       {
         \regex_replace_once:NnNTF \l__datatool_delim_left_regex { }
           \l__datatool_tmpb_tl
         {
            \tl_set_eq:NN \l__datatool_item_value_tl \l__datatool_tmpb_tl
         }
         {
            \bool_if:NT \l__datatool_new_element_trim_bool
            {
              \tl_trim_spaces:N \l__datatool_tmpb_tl
            }
            \seq_put_right:NV \l__datatool_row_seq \l__datatool_tmpb_tl
         }
       }
    }
    {
```

Version 3.3: bug fix (middle part missing if multiple separators in delimited cell). If we've reached here, the cell is under construction. Only end the cell construction if the closing separator is found.

```
       \regex_replace_once:NnNTF \l__datatool_delim_right_regex { }
         \l__datatool_tmpb_tl
       {
          \tl_put_right:NV \l__datatool_item_value_tl \@dtl@separator
          \tl_put_right:NV \l__datatool_item_value_tl \l__datatool_tmpb_tl
          \seq_put_right:NV \l__datatool_row_seq \l__datatool_item_value_tl
          \tl_set:Nn \l__datatool_item_value_tl { \q_no_value }
       }
       {
          \tl_put_right:NV \l__datatool_item_value_tl \@dtl@separator
          \tl_put_right:NV \l__datatool_item_value_tl \l__datatool_tmpb_tl
       }
    }
```

```
    \__datatool_map_row_seq:
  }
}
```

Load DTLTEX or DBTEX file:

```
\cs_new:Nn \__datatool_load_tex:
{
  \tl_clear:N \dtllastloadeddb
  \file_input:n { \l__datatool_name_tl }
  \tl_if_empty:NT \l__datatool_io_name_tl
  {
    \tl_set_eq:NN \l__datatool_io_name_tl \dtllastloadeddb
  }
}
\ExplSyntaxOff
```

\DTLloaddbtex

\DTLloaddbtex{⟨*cs*⟩}{⟨*file*⟩}

Version 3.0: This now just uses \DTLread and assigns the control sequence.

```
\newcommand*{\DTLloaddbtex}[2]{%
  \DTLread[format=dbtex]{#2}%
  \let#1\dtllastloadeddb
}
```

\DTLloaddb

\DTLloaddb[⟨*options*⟩]{⟨*db name*⟩}{⟨*filename*⟩}

Version 3.0: rewritten to simply use \DTLread.

```
\NewDocumentCommand\DTLloaddb{O{}mm}{%
  \DTLread[name={#2},format=csv,csv-content=tex,#1]{#3}%
}
```

\DTLloadrawdb

\DTLloadrawdb[⟨*options*⟩]{⟨*db name*⟩}{⟨*filename*⟩}

```
\NewDocumentCommand\DTLloadrawdb{O{}mm}{%
  \DTLread[name={#2},format=csv,csv-content=literal,#1]{#3}%
}
```

\DTLrawmap

\DTLrawmap{⟨*string*⟩}{⟨*replacement*⟩}

Additional mappings to perform when reading a raw data file. Version 3.0: this is still supported for the literal CSV read.

```
\newcommand*{\DTLrawmap}[2]{%
  \expandafter\@dtl@toks\expandafter{\@dtl@rawmappings}%
```

```
    \ifdefempty{\@dtl@rawmappings}%
    {%
      \def\@dtl@rawmappings{{#1}{#2}}%
    }%
    {%
      \def\@dtl@tmp{{#1}{#2}}%
      \protected@edef\@dtl@rawmappings{\the\@dtl@toks,\@dtl@tmp}%
    }%
  }
```

`\@dtl@rawmappings` List of mappings.

```
\newcommand*{\@dtl@rawmappings}{}
```

`\dtl@domappings` `\dtl@domappings{⟨cmd⟩}`

Do all mappings in string given by ⟨*cmd*⟩.

```
\newcommand*{\dtl@domappings}[1]{%
  \@for\@dtl@map:=\@dtl@rawmappings\do{%
    \expandafter\DTLsubstituteall\expandafter#1\@dtl@map
  }%
}
```

12.13 Debugging commands

These commands are provided to assist debugging

`\DTLdbLog` `\DTLdbLog{⟨db name⟩}`

Writes detailed information in the log file about the database structure. NB this uses a regex for convenience but it means that it can only be used for small database as l3regex can't handle more than 32732 tokens. For large databases, try saving as dbtex instead. This is here to debug the code to check for incorrect markup.

```
\ExplSyntaxOn
\NewDocumentCommand{\DTLdbLog} { m }
{
  \group_begin:
  \tl_set:Nv \__datatool_tmpa_tl { dtldb@#1 }
  \regex_replace_case_all:nN
   {
     { \c{ db@row@elt@w } } { < ROW \ BLOCK \ START > ^^J }
     { \c{ db@row@elt@end@ } } { < ROW \ BLOCK \ END > ^^J }
     { \c{ db@row@id@w } } { \ < ROW \ ID \ DELIM \ START > }
     { \c{ db@row@id@end@ } } { < ROW \ ID \ DELIM \ END > ^^J }
     { \c{ db@col@id@w } } { \ \ < COL \ ID \ DELIM \ START > }
     { \c{ db@col@id@end@ } } { < COL \ ID \ DELIM \ END > ^^J }
```

```
      { \c{ db@col@elt@w } } { \ \ \ < ELEMENT \ START > }
      { \c{ db@col@elt@end@ } } { < ELEMENT \ END >  ^^J }
    }
    \__datatool_tmpa_tl
  \PackageInfo { datatool }
  {
    Details ~ of ~ database ~ `#1': ^^J
    Column ~ Count: ~ \DTLcolumncount{#1}. ~
    Row ~ Count: ~ \DTLrowcount{#1}. ^^J
    Correctly ~ formatted ~ database ~ content ~
    should ~ have ~ each ~ row ~ marked ~ up ~
    with: ^^J
    <ROW ~ BLOCK ~ START> ^^J
    <ROW ~ ID ~ DELIM ~ START> rowidx
    <ROW ~ ID ~ DELIM ~ END> ^^J
    column data ^^J
    <ROW ~ ID ~ DELIM ~ START> rowidx
    <ROW ~ ID ~ DELIM ~ END> ^^J
    <ROW ~ BLOCK ~ END>^^J
    Content ~ of ~ `#1':^^J
    \exp_not:V \__datatool_tmpa_tl
  }
  \group_end:
}
\ExplSyntaxOff
```

\dtlshowdb

```
\dtlshowdb{⟨db name⟩}
```

Shows the database.

```
\newcommand*{\dtlshowdb}[1]{%
  \ifcsundef{dtldb@#1}%
  {\PackageError{datatool}{Database `#1' not defined}{}}%
  {\expandafter\showthe\csname dtldb@#1\endcsname}%
}
```

\dtlshowdbkeys

```
\dtlshowdbkeys{⟨db name⟩}
```

Shows the key list for the named database.

```
\newcommand*{\dtlshowdbkeys}[1]{%
  \ifcsundef{dtlkeys@#1}%
  {\PackageError{datatool}{Database `#1' not defined}{}}%
  {\expandafter\showthe\csname dtlkeys@#1\endcsname}%
}
```

\dtlshowtype

```
\dtlshowtype{⟨db name⟩}{⟨key⟩}
```

Show the data type for given key in the named database. This should be an integer from 0 to 3.

```
\newcommand*{\dtlshowtype}[2]{%
  \ifcsundef{dtldb@#1}%
  {\PackageError{datatool}{Database `#1' not defined}{}}%
  {\DTLgetdatatype{\@dtl@type}{#1}{#2}\show\@dtl@type}%
}
```

12.14 Deprecated

These commands are marked for removal.

\@dtl@construct@lopoff

\@dtl@construct@lopoff⟨*separator char*⟩⟨*delimiter char*⟩

This defines

\@dtl@lopoff⟨*first element*⟩⟨*sep*⟩⟨*rest of list*⟩\to⟨*cmd1*⟩⟨*cmd2*⟩

for the current separator and delimiter.

```
\edef\@dtl@construct@lopoff#1#2{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@lopoff#1##1##2\noexpand\to##3##4{%
      \noexpand\ifx#2##1\noexpand\relax
        \noexpand\ifstrempty{##1}%
        {\noexpand\@dtl@qlopoff#1{}##2\noexpand\to##3##4\relax}%
        {%
          \noexpand\dtl@ifsingle{##1}%
          {\noexpand\@dtl@qlopoff#1##1##2\noexpand\to##3##4\relax}%
          {\noexpand\@dtl@qlopoff#1{##1}##2\noexpand\to##3##4\relax}%
        }%
      \noexpand\else
        \noexpand\ifstrempty{##1}%
        {\noexpand\@dtl@lop@ff#1{}##2\noexpand\to##3##4\relax}%
        {%
          \noexpand\dtl@ifsingle{##1}%
          {\noexpand\@dtl@lop@ff#1##1##2\noexpand\to##3##4\relax}%
          {\noexpand\@dtl@lop@ff#1{##1}##2\noexpand\to##3##4\relax}%
        }%
      \noexpand\fi
  }%
}
```

\@dtl@construct@qlopoff

\@dtl@construct@qlopoff⟨*separator char*⟩⟨*delimiter char*⟩

This constructs \@dtl@qlopoff to be used when the entry is surrounded by the current delimiter value.

```
\edef\@dtl@construct@qlopoff#1#2{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@qlopoff#1#2##1#2#1##2\noexpand\to##3##4{%
      \noexpand\def##4{##1}%
```

Replace any escaped delimiters

```
      \noexpand\DTLsubstituteall{##4}{#2#2}{#2}%
      \noexpand\edef\noexpand\@dtl@dosubs{%
        \noexpand\noexpand\noexpand\DTLsubstituteall{\noexpand\noexpand##4}%
        {\noexpand\expandafter\noexpand\noexpand\noexpand\csname#2\noexpand\endcsname#2}
        {\noexpand\expandafter\noexpand\noexpand\noexpand\csname#2\noexpand\endcsname}%
      }%
      \noexpand\@dtl@dosubs
      \noexpand\def##3{#1##2}%
  }%
}
```

\@dtl@construct@lop@ff

\@dtl@construct@lop@ff⟨*separator char*⟩

This constructs \@dtl@lop@ff to be used when the entry isn't surrouded by the delimiter.

```
\edef\@dtl@construct@lop@ff#1{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@lop@ff#1##1#1##2\noexpand\to##3##4{%
      \noexpand\def##4{##1}%
      \noexpand\def##3{#1##2}%
  }%
}
```

\@dtl@construct@lopoffs

\@dtl@construct@lopoffs

This constructs all the lopoff macros using the given separator and delimiter characters.

```
\newcommand{\@dtl@construct@lopoffs}{%
  \edef\@dtl@chars{{\@dtl@separator}{\@dtl@delimiter}}%
  \expandafter\@dtl@construct@lopoff\@dtl@chars
  \expandafter\@dtl@construct@qlopoff\@dtl@chars
  \expandafter\@dtl@construct@lop@ff\expandafter{\@dtl@separator}%
}
```

13 datagidx.sty

This package provides a means to produces indices and glossaries without the need for an external indexing application, such as `makeindex` or `xindy`. However, the code here has been developed to implement the word order style described by the Oxford Style Manual. If you are not writing in English, this may not be applicable to your needs. You may be able to define your own comparison handler to use with `\dtlsort`. If not, you'll need to use `xindy` with a package such as glossaries.

Declare package:

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{datagidx-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{datagidx}[2026/07/31 v3.4.4 (NLCT)]
\RequirePackage{etoolbox}
```

14 Default Settings

```
\ExplSyntaxOn
```

These commands need to be defined before the package options are used.

`\datagidx@columns` The number of columns to use for the index/glossary. Version 3.0: replaced `\datagidx@columns` with:

```
\int_new:N \l__datagidx_columns_int
\int_set:Nn \l__datagidx_columns_int { 2 }
```

Define commands used in package options.

`\DTLgidxSetColumns`

```
\NewDocumentCommand \DTLgidxSetColumns { m }
{
  \int_compare:nNnTF
    { #1 } < { \c_one_int }
  {
    \PackageError { datagidx }
    {
      \token_to_str:N \DTLgidxSetColumns \c_space_tl ~
      invalid ~ number ~ of ~ columns ~ #1
    }
    {
      The ~ number ~ of ~ columns ~ must ~ be ~
      greater ~ than ~ 0
    }
  }
  {
```

```
    \int_set:Nn \l__datagidx_columns_int { #1 }
  }
}
```

`\DTLgidxChildStyle` Should the child name be displayed? (Default: show name.) If the name shouldn't be displayed, replace with a number.

```
\newcommand*{\DTLgidxChildStyle}[1]{#1}
```

`\datagidx@setchildstyle` Version 3.0: removed.

```
\cs_new:Nn \__datagidx_set_child_style_named:
{
  \cs_set_eq:NN \DTLgidxChildStyle \use:n
}
\cs_new:Nn \__datagidx_set_child_style_noname:
{
  \renewcommand*{\DTLgidxChildStyle}[1]{
    \DTLgidxChildCountLabel
  }
}
```

`\DTLgidxPostName` What to put after the name. (Defaults to space.)

```
\newcommand*{\DTLgidxPostName}{ ~ }
```

`\DTLgidxPostChildName` What to put after the child name.

```
\newcommand*{\DTLgidxPostChildName}{\DTLgidxPostName}
```

`\DTLgidxNameCase` Should the name have a case change in the index/glossary? (Default: no change.)

```
\newcommand*{\DTLgidxNameCase}[1]{#1}
```

`\datagidx@setnamecase` Version 3.0: removed.

`\DTLgidxNameFont` The font to use for the name in the index/glossary. (Default: normal font.)

```
\newcommand*{\DTLgidxNameFont}[1]{\textnormal{#1}}
```

`\DTLgidxPostDescription` What to put after the description. (Defaults to nothing.)

```
\newcommand*{\DTLgidxPostDescription}{}
```

`\datagidx@setpostdesc` Version 3.0: removed.

`\DTLgidxPreLocation` What to put before the location list. (Defaults to en-space.)

```
\newcommand*{\DTLgidxPreLocation}{\enspace}
```

`\DTLgidxPostLocation` What to put after the location list. (Defaults to space.)

```
\newcommand*{\DTLgidxPostLocation}{ ~ }
```

`\datagidx@setprelocation` Version 3.0: removed.

`\DTLgidxLocation` How to display the location. (Defaults to show the location list.)

```
\newcommand*{\DTLgidxLocation}{\dtldolocationlist}
```

`\datagidx@setlocation` Version 3.0: removed.

`\DTLgidxSee` How to display the cross-reference list.

```
\newcommand*{\DTLgidxSee}{
  \datatool_if_null_or_empty:NF \See
  {
    \DTLgidxPreLocation
    \DTLgidxFormatSee { \seename } { \See }
  }
}
```

`\DTLgidxSeeAlso` How to display the “see also” list.

```
\newcommand*{\DTLgidxSeeAlso}{%
  \datatool_if_null_or_empty:NF \SeeAlso
  {
    \DTLgidxFormatSeeAlso
     { \seealsoname } { \SeeAlso }
  }
}
```

`\DTLgidxChildrenSeeAlso` Display the children and the see also attributes.

```
\newcommand*{\DTLgidxChildrenSeeAlso}{%
  \DTLgidxChildren
  \DTLgidxSeeAlso
}
```

`\datagidx@setsee` Version 3.0: removed.

`\DTLgidxSymDescSep` Separator character between symbol and description if both are present.

```
\newcommand*{\DTLgidxSymDescSep}{\space}
```

`\DTLgidxFormatDesc` How to format the description.

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

`DTLgidxSymbolDescription` How to format the symbol and description fields.

```
\newcommand*{\DTLgidxSymbolDescription}
{
  \DTLgidxSymbolDescLeft
  \DTLgidxSymbolDescRight
}
\newcommand*{\DTLgidxSymbolDescLeft}
{
  \tl_if_empty:NF \Symbol
    { ( \Symbol ) \DTLgidxSymDescSep }
}
\newcommand*{\DTLgidxSymbolDescRight}
{
  \tl_if_empty:NF \Description
  {
```

```
    \DTLgidxFormatDesc { \Description }
    \DTLgidxPostDescription
  }
}
```

`\if@datagidxsymbolleft` Identifies whether the symbol has been set to left or right. Version 3.0: replaced `\if@datagidxsymbolleft` with:

```
\bool_new:N \l__datagidx_symbol_left_bool
\bool_set_true:N \l__datagidx_symbol_left_bool
```

`\datagidx@formatsymdesc` Version 3.0: removed.

Only symbol

```
\cs_new:Nn \__datagix_set_symbol_only:
{
  \tl_set:Nn \DTLgidxSymbolDescLeft
   {
     \tl_if_empty:NF \Symbol { \Symbol }
   }
  \tl_clear:N \DTLgidxSymbolDescRight
  \bool_set_true:N \l__datagidx_symbol_left_bool
}
```

Only description

```
\cs_new:Nn \__datagix_set_desc_only:
{
  \tl_set:Nn \DTLgidxSymbolDescLeft
   {
     \tl_if_empty:NF \Description
      {
        \DTLgidxFormatDesc { \Description }
        \DTLgidxPostDescription
      }
   }
  \tl_clear:N \DTLgidxSymbolDescRight
  \bool_set_false:N \l__datagidx_symbol_left_bool
}
```

(symbol) description

```
\cs_new:Nn \__datagix_set_symbol_paren_desc:
{
  \tl_set:Nn \DTLgidxSymbolDescLeft
   {
     \tl_if_empty:NF \Symbol
       { ( \Symbol ) \DTLgidxSymDescSep }
   }
  \tl_set:Nn \DTLgidxSymbolDescRight
   {
     \tl_if_empty:NF \Description
       {
```

```
          \DTLgidxFormatDesc { \Description }
          \DTLgidxPostDescription
        }
    }
   \bool_set_true:N \l__datagidx_symbol_left_bool
 }
```

description (symbol)

```
 \cs_new:Nn \__datagix_set_desc_symbol_paren:
 {
   \tl_set:Nn \DTLgidxSymbolDescLeft
    {
      \tl_if_empty:NF \Description
       {
         \DTLgidxFormatDesc { \Description }
         \DTLgidxPostDescription
         \DTLgidxSymDescSep
       }
    }
   \tl_set:Nn \DTLgidxSymbolDescRight
    {
      \tl_if_empty:NF \Symbol
       { (\Symbol) }
    }
   \bool_set_false:N \l__datagidx_symbol_left_bool
 }
```

symbol description

```
 \cs_new:Nn \__datagix_set_symbol_desc:
 {
   \tl_set:Nn \DTLgidxSymbolDescLeft
    {
      \tl_if_empty:NF \Symbol
        { \Symbol \DTLgidxSymDescSep }
    }
   \tl_set:Nn \DTLgidxSymbolDescRight
    {
      \tl_if_empty:NF \Description
       {
         \DTLgidxFormatDesc { \Description }
         \DTLgidxPostDescription
       }
    }
   \bool_set_true:N \l__datagidx_symbol_left_bool
 }
```

description symbol

```
 \cs_new:Nn \__datagix_set_desc_symbol:
 {
   \tl_set:Nn \DTLgidxSymbolDescLeft
    {
```

```
    \tl_if_empty:NF \Description
     {
       \DTLgidxFormatDesc { \Description }
       \DTLgidxPostDescription
       \DTLgidxSymDescSep
     }
   }
  \tl_set:Nn \DTLgidxSymbolDescRight
   {
     \tl_if_empty:NF \Symbol { \Symbol }
   }
  \bool_set_false:N \l__datagidx_symbol_left_bool
}
```

`\datagidx@compositor` Version 3.0: replaced `\datagidx@compositor` with:

```
\tl_new:N \l__datagidx_compositor_tl
```

`\DTLgidxSetCompositor`

`\DTLgidxSetCompositor{`⟨*symbol*⟩`}`

Set the location compositor.

```
\NewDocumentCommand \DTLgidxSetCompositor { m }
{
  \tl_set:Nn \l__datagidx_compositor_tl { #1 }
}
```

Set the default compositor to `.` (full stop).

```
\DTLgidxSetCompositor{.}
```

`\DTLgidxCounter` The counter used for the location lists.

```
\newcommand*{\DTLgidxCounter}{page}

\cs_new:Nn \__datagidx_set_counter:n
{
  \tl_if_empty:nTF { #1 }
   {
     \PackageError { datagidx }
     { Missing ~ counter ~ name }
     { The ~ argument ~ of ~ `counter' ~ must ~ be ~ a ~ counter ~ name }
   }
   {
    \tl_if_exist:cTF { the#1 }
     {
       \tl_set:Nn \DTLgidxCounter { #1 }
     }
     {
       \PackageError { datagidx }
       { Invalid ~ counter ~ name ~ `#1' }
       {
```

```
            The ~ argument ~ of ~ `counter' ~ must ~ be ~ a ~
            counter ~ name ~ (or ~ \token_to_str:N \the#1 \c_space_tl
            must ~ be ~ defined)
          }
        }
      }
    }
```

Sorting can take a long time (especially with large databases) but two LaTeX runs are usually required to get the index or glossary up-to-date, so we usually don't need to worry about sorting on the first run (unless the order in some way affects the document, e.g. the group headings are to appear in the table of contents). It may also be that some modifications are done to the document that don't require a re-sort. The optimize setting tries to minimize the amount of sorting done to help speed up document compilation.

There are to optimization levels: low and high. The low level optimization just sorts every other LaTeX run. This is done by writing to the aux file to determine whether or not the sort should be done next run. This is a cheap and easy hack that won't work if sorting makes the document out-of-date (for example, if the sorted index or glossary affects the table of contents by, say, making the group headings a sectional unit).

The high level optimization is more complicated and involves writing the sorted database to an external file and reading it in on the next run. This requires checks to see if the location lists have changed, in which case a new sort may be required.

> The optimization function is only implemented when the sorting is specified via the sort key. Any explicit sorting done by the user via commands such as `\dltsort` are not effected by the optimization setting.

`\datagidx@do@sort` Indicate what to do when it's time to sort the index/glossary. This defaults to un-optimised setting to avoid confusing users who don't like to read the manual.

```
\newcommand*{\datagidx@do@sort}{ \l__datagidx_sort_tl }
```

First deal with the low-level optimization as it's easier to implement.

`\datagidx@optimize@sort` The code to perform when the low optimize setting is on. If the command `\datagidx@do@optimize@sort` has been defined, do the sort. If it hasn't been defined, don't sort. If a sort isn't performed, the command definition is written to the aux file. If a sort is performed, the command definition isn't written to the aux file. This will do the sort every other run.

```
\newcommand*{\datagidx@optimize@sort}{%
```

First, has `\datagidx@do@optimize@sort` been defined?

```
  \ifdef\datagidx@do@optimize@sort
  {
```

It has been defined so go ahead and do the sort.

```
    \l__datagidx_sort_tl
  }
  {
```

It hasn't been defined so don't sort. Write the command definition into the aux file for the next run.

```
      \protected@write\@auxout{}{%
        \string\gdef\string\datagidx@do@optimize@sort{}%
      }
```

Let the user know they need to recompile the document.

```
      \cs_gset_eq:NN
        \@datagidx@dorerun@warn@sort
        \@data@rerun@warn@sort
    }
  }
```

List of labels to check for re-run.

```
\seq_new:N \g__datagidx_refd_labels_seq
```

`\if@datagidx@warn` Provide a switch to allow warnings to be suppressed. Version 3.0: replaced `\if@datagidx@warn` with:

```
\bool_new:N \l__datagidx_warn_bool
\bool_set_true:N \l__datagidx_warn_bool
```

`\@datagidx@dorerun@warn`

```
\newcommand*\@datagidx@dorerun@warn{%
  \seq_map_inline:Nn \g__datagidx_refd_labels_seq
   {
     \iftermexists { ##1 }
     {
       \DTLaction
        [
          name = { \__datagidx_term_database:n { ##1 } },
          key = Label, value = { ##1 },
          return =
           {
```

Locations from previous run that may be a page out:

```
              \UnsafeLocation = UnsafeLocation,
```

Locations from this run that may be a page out:

```
              \CurrentLocation = CurrentLocation
```

If the document hasn't changed, they will both be out by the same amount so they can still be compared. However, protected expansion may cause extra spaces.

```
           }
        ]
        { select ~ row }
       \tl_set:Nx \UnsafeLocation
         { \text_purify:n { \UnsafeLocation } }
       \tl_set:Nx \CurrentLocation
         { \text_purify:n { \CurrentLocation } }
       \tl_remove_all:Nn  \UnsafeLocation { ~ }
```

```
        \tl_remove_all:Nn  \CurrentLocation { ~ }
        \tl_if_eq:NNF \UnsafeLocation \CurrentLocation
         {
           \@data@rerun@warn
           \seq_map_break:
         }
      }
      { }
   }
}
```

`tagidx@dorerun@warn@sort`

```
\newcommand*\@datagidx@dorerun@warn@sort{}

\AtEndDocument
 {
  \bool_if:NT \l__datagidx_warn_bool
   {
     \@datagidx@dorerun@warn
     \@datagidx@dorerun@warn@sort
   }
 }
```

`datagidx@rerun@warn@sort` Warning issued when a rerun is required to sort the index or glossary.

```
\newcommand*\@data@rerun@warn@sort{
  \PackageWarningNoLine {datagidx}
   {
     Rerun ~ required ~ to ~ sort ~ the ~
     index/glossary ~ databases
   }
}
```

`\@datagidx@rerun@warn` Warning issued when a rerun is required to update the location lists.

```
\newcommand*\@data@rerun@warn{
  \PackageWarningNoLine {datagidx}
   {
     Rerun ~ required ~ to ~ ensure ~ the ~
     index/glossary ~ location ~ lists ~ are ~ up-to-date
   }
}
```

The high optimize setting is more complicated. This involves writing each database to an external file (named `\jobname`-⟨*db label*⟩.gidx). The sort is only performed if new terms are added or used.

`gidx@do@highopt@optimize`

```
\newcommand*{\datagidx@do@highopt@optimize}{%
  \renewcommand*{\datagidx@do@sort}{%
```

Only sort if database has changed.

```
\ifcsdef{datagidx@do@highopt@sort@\DTLgidxCurrentdb}%
{%
  \csuse{datagidx@do@highopt@sort@\DTLgidxCurrentdb}%
}%
{%
```

Do nothing

```
 \dtl@message
  {
    Not ~ sorting ~ `\DTLgidxCurrentdb' : ~
    optimize=high ~ on ~ and ~ no ~ change ~
    detected
  }
}%
```

Save the database to file.

```
\group_begin:
```

Locally redefine `\__datagidx_used:n` to expand to 0. Since this is encapsulated with `\dtlspecialvalue`, it will expand regardless of the `expand` setting. Similarly the location needs to expand to nothing.

```
\cs_set:Nn \__datagidx_used:n { 0 }
\cs_set_eq:NN \__datagidx_location:n \use_none:n
\cs_set_eq:NN
 \__datagidx_letter_group:n
 \__datagidx_write_letter_group:n
```

Write in latest dbtex format.

```
\DTLwrite
 [
   overwrite=allow,
   format = dbtex, expand=none,
   name = \DTLgidxCurrentdb
 ]
 { \datagidxhighoptfilename \DTLgidxCurrentdb }
\group_end:
}%
```

Change the behaviour of `\newgidx`

```
\def\newgidx{\datagidx@highopt@newgidx}%
```

Change the behaviour of `\newterm`

```
\def\newterm{\datagidx@highopt@newterm}%
}
```

`\@datagidx@db@col@id@w` Version 3.0: removed.

With the ‘highopt optimize’ setting, whenever a location is written to the aux file, if no location has been defined the database needs sorting.

tagidx@do@highopt@update Default does nothing. (Argument is the entry's label.)

```
\newcommand*{\datagidx@do@highopt@update}[1]{}
```

\datagidxhighoptfilename Expands to the name of the filename associated with the database identified by the argument for the 'highopt' setting.

```
\newcommand*{\datagidxhighoptfilename}[1]{\jobname-#1.gidx}
```

15 Package Options

These options govern the general layout of the glossary/index and may be passed as package options.

```
\keys_define:nn { datatool }
{
  columns .code:n = { \DTLgidxSetColumns { #1 } },
  child .choice: ,
  child / named .code:n = { \__datagidx_set_child_style_named: } ,
  child / noname .code:n = { \__datagidx_set_child_style_noname: } ,
  namecase .choice: ,
  namecase / nochange .code:n =
    { \let \DTLgidxNameCase \use:n } ,
  namecase / uc .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \text_uppercase:n { ##1 } } } ,
  namecase / lc .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \text_lowercase:n { ##1 } } } ,
  namecase / firstuc .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \xmakefirstuc { ##1 } } } ,
  namecase / capitalise .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \xcapitalisewords { ##1 } } } ,
  namefont .code:n =
    { \renewcommand \DTLgidxNameFont [ 1 ] { { #1 { ##1 } } } } ,
  postname .code:n =
    { \renewcommand \DTLgidxPostName { #1 } } ,
  postdesc .choice: ,
  postdesc / none .code:n =
   { \tl_clear:N \DTLgidxPostDescription } ,
  postdesc / dot .code:n =
   { \tl_set:Nn \DTLgidxPostDescription { . } } ,
  prelocation .choice: ,
  prelocation / none .code:n =
   { \tl_clear:N \DTLgidxPreLocation } ,
  prelocation / enspace .code:n =
   { \tl_set:Nn \DTLgidxPreLocation { \enspace } } ,
  prelocation / space .code:n =
   { \tl_set:Nn \DTLgidxPreLocation { ~ } } ,
  prelocation / dotfill .code:n =
   { \tl_set:Nn \DTLgidxPreLocation { \dotfill } } ,
  prelocation / hfill .code:n =
   { \tl_set:Nn \DTLgidxPreLocation { \hfill } } ,
```

```
location .choice: ,
location / hide .code:n =
  { \tl_clear:N \DTLgidxLocation },
location / list .code:n =
  { \tl_set:Nn \DTLgidxLocation { \dtldolocationlist } },
location / first .code:n =
  { \tl_set:Nn \DTLgidxLocation { \dtldofirstlocation } },
see .choice: ,
see / comma .code:n =
  {
    \renewcommand*{\DTLgidxSee}
     {
       \datatool_if_null_or_empty:NF \See
        {
         , ~ \DTLgidxFormatSee { \seename } { \See }
        }
     }
  } ,
see / brackets .code:n =
  {
    \renewcommand*{\DTLgidxSee}
     {
       \datatool_if_null_or_empty:NF \See
        {
          \space ( \DTLgidxFormatSee { \seename } { \See } )
        }
     }
  } ,
see / dot .code:n =
  {
    \renewcommand*{\DTLgidxSee}
     {
       \datatool_if_null_or_empty:NF \See
        {
          .  ~
          \DTLgidxFormatSee
            { \xmakefirstuc { \seename } } { \See }
        }
     }
  } ,
see / space .code:n =
  {
    \renewcommand*{\DTLgidxSee}
     {
       \datatool_if_null_or_empty:NF \See
        {
          \space
          \DTLgidxFormatSee { \seename } { \See }
        }
     }
```

```
  } ,
see / nosep .code:n =
  {
    \renewcommand*{\DTLgidxSee}
     {
       \datatool_if_null_or_empty:NF \See
        {
          \DTLgidxFormatSee { \seename } { \See }
        }
     }
  } ,
see / semicolon .code:n =
  {
    \renewcommand*{\DTLgidxSee}
     {
       \datatool_if_null_or_empty:NF \See
        {
          ; ~ \DTLgidxFormatSee { \seename } { \See }
        }
     }
  } ,
see / location .code:n =
  {
    \renewcommand*{\DTLgidxSee}
     {
       \datatool_if_null_or_empty:NF \See
        {
          \DTLgidxPreLocation
          \DTLgidxFormatSee { \seename } { \See }
        }
     }
  } ,
symboldesc .choice: ,
symboldesc / symbol .code:n =
 {
   \__datagix_set_symbol_only:
 } ,
symboldesc / desc .code:n =
 {
   \__datagix_set_desc_only:
 } ,
symboldesc / (symbol) ~ desc .code:n =
 {
   \__datagix_set_symbol_paren_desc:
 } ,
symboldesc / desc ~ (symbol) .code:n =
 {
   \__datagix_set_desc_symbol_paren:
 } ,
symboldesc / symbol ~ desc .code:n =
```

```
    {
      \__datagix_set_symbol_desc:
    } ,
   symboldesc / desc ~ symbol .code:n =
    {
      \__datagix_set_desc_symbol:
    } ,
   counter .code:n =
    {
      \__datagidx_set_counter:n { #1 }
    },
   counter .value_required:n = true ,
   compositor .tl_set:N = \l__datagidx_compositor_tl ,
   compositor .value_required:n = true ,
   final .code:n =
     { \cs_set_eq:NN \datagidxshowifdraft \use_none:n } ,
   final .value_forbidden:n = true ,
   draft .code:n =
     { \cs_set_eq:NN \datagidxshowifdraft \use:n } ,
   draft .value_forbidden:n = true ,
```

`optimize` A boolean option indicating whether or not to optimize the sort. This is only available as a global option. If you want to optimize some glossaries but not others, switch on the optimize function and clear the sort key for the relevant glossaries and manually sort using `\dtlsort` before the glossary is displayed.

```
   optimize .choice: ,
   optimize / off .code:n =
    {
     \renewcommand*{\datagidx@do@sort}{ \l__datagidx_sort_tl }
    } ,
   optimize / low .code:n =
    {
     \renewcommand*{\datagidx@do@sort}{\datagidx@optimize@sort}
    } ,
   optimize / high .code:n =
    {
      \datagidx@do@highopt@optimize
    } ,
   optimize .default:n = { high } ,
   nowarn .bool_set_inverse:N = \l__datagidx_warn_bool ,
 }
```

Set final as default:

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

```
\ExplSyntaxOff
```

Process package options and load datatool if not already loaded. This allows datatool package options to be supplied with datagix.

```
\IfPackageLoadedTF{datatool}
```

```
{
  \ProcessKeyOptions[datatool]
}
{
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datatool}}
  \ProcessOptions
}
\RequirePackage{datatool}
```

Remove the package option keys so they can't be used with `\DTLsetup` directly.

```
\ExplSyntaxOn
\keys_define:nn { datatool }
{
  columns .undefine: ,
  child .undefine: ,
  namecase .undefine: ,
  postname .undefine: ,
  postdesc .undefine: ,
  prelocation .undefine: ,
  location .undefine: ,
  see .undefine: ,
  symboldesc .undefine: ,
  counter .undefine: ,
  compositor .undefine: ,
  final .undefine: ,
  draft .undefine: ,
  optimize .undefine: ,
  nowarn .undefine:
}
\ExplSyntaxOff
%
```

16 Initialisation

Required packages. Version 3.0: removed xkeyval

```
\RequirePackage{mfirstuc}[2022/10/14]
\RequirePackage{multicol}

\ExplSyntaxOn
```

Scratch variables:

`\datagidx@child` Version 3.0: replaced `\datagidx@child` with:

```
\tl_new:N \l__datagidx_child_label_tl
\clist_new:N \l__datagidx_child_clist
\seq_new:N \l__datagidx_child_seq
```

These labels are token list variables not strings as they may be set to another command that the user may want to redefine.

\datagidx@parentdatabase Version 3.0: replaced \datagidx@parentdatabase with:
\tl_new:N \l__datagidx_parent_database_tl

Temporary database, column and term labels.
\tl_new:N \l__datagidx_database_tl
\tl_new:N \l__datagidx_label_tl

\datagidx@id Version 3.0: replaced \datagidx@id with:
\tl_new:N \l__datagidx_id_tl

\@datagidx@target Version 3.0: replaced \@datagidx@target with:
\tl_new:N \l__datagidx_target_tl

\datagidx@parent Version 3.0: replaced \datagidx@parent with:
\tl_new:N \l__datagidx_parent_tl

\datagidx@list Version 3.0: replaced \datagidx@list with:
\clist_new:N \l__datagidx_database_clist

Temporary value:
\tl_new:N \l__datagidx_value_tl

\datagidx@title Database title. Replaced \datagidx@title with:
\tl_new:N \l__datagidx_title_tl

Location variables

\datagidx@loc Version 3.0: replaced \datagidx@loc with:
\tl_new:N \l__datagidx_location_tl
\clist_new:N \l__datagidx_location_clist

\datagidx@sep Version 3.0: replaced \datagidx@sep with:
\tl_new:N \l__datagidx_sep_tl

Database to keep track of all the defined terms.
\DTLnewdb{datagidx}

DTLgidxChildCount Child counter.
\newcounter{DTLgidxChildCount}

\theHDTLgidxChildCount Reduce duplicate identifier warnings if hyperref in use. \Label is a placeholder locally set while iterating.
\def\theHDTLgidxChildCount{\Label.\arabic{DTLgidxChildCount}}

\DTLgidxChildCountLabel Label for child counter.
\newcommand*{\DTLgidxChildCountLabel}{\theDTLgidxChildCount) ~ }

`\datagidx@foreachchild` Iterate through each child label in `\Children`, which should be a comma-separated list of labels. The current database should be in `\DTLgidxCurrentdb`.

```
\newcommand{\datagidx@foreachchild}[1]{%
 \bool_if:NTF \l__datagidx_childsort_bool
  {
    \seq_clear:N \l__datagidx_child_seq
    \clist_map_inline:Nn \Children
     {
```

Get the row index only

```
        \__datatool_get_row_index:Nnxn
         \l__datatool_row_idx_tl
         { \DTLgidxCurrentdb }
         { \dtlcolumnindex { \DTLgidxCurrentdb } { Label } }
         { ##1 }
        \datatool_if_null:NF \l__datatool_row_idx_tl
         {
           \seq_put_right:Nx \l__datagidx_child_seq
             { { \l__datatool_row_idx_tl } { ##1 } }
         }
     }
```

Sort by row index:

```
    \seq_sort:Nn \l__datagidx_child_seq
     {
       \int_compare:nNnTF
         { \use_i:nn ##1 } > { \use_i:nn ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
     }
    \seq_map_inline:Nn \l__datagidx_child_seq
     {
       \tl_set:Nx \Label { \use_ii:nn ##1 }
       #1
     }
  }
  {
    \clist_map_variable:NNn \Children \Label
     { #1 }
  }
}
```

`\datagidx@sortchildren` The list of child labels needs to be sorted so that the child list follows the same ordering as the database. Version 3.0: removed `\datagidx@sortchildren`.

`tagidx@sort@foreachchild` Sorted iteration through all the child labels. Version 3.0: deprecated. Hierarchical sorting now implemented with HierSort column. Version 3.0: removed `\datagidx@sort@foreachchild`.

`gidx@unsort@foreachchild` Unsorted iteration through all the child labels. Version 3.0: removed `\datagidx@unsort@foreachchild`.

\datagidx@setchildsort Version 3.0: removed \datagidx@setchildsort.

\datagidxsymbolwidth Space to allocate for the symbol. If zero or negative, symbol just occupies its natural space.

```
\newlength\datagidxsymbolwidth
```

\datagidxlocationwidth Space to allocate for the location list. If zero or negative, the list just occupies its natural space.

```
\newlength\datagidxlocationwidth
```

17 Glossary/Index Formatting

\seename

```
\providecommand*{\seename}{see}
```

\seealsoname

```
\tl_if_exist:NF \seealsoname
 {
  \tl_if_exist:NTF \alsoname
  {
    \tl_set:Nn \seealsoname { \alsoname }
  }
  {
    \tl_set:Nn \seealsoname { see ~ also }
  }
}
```

\DTLgidxSeeTagFont

```
\newcommand*{\DTLgidxSeeTagFont}[1]{\emph{#1}}
```

\DTLgidxFormatSee

```
\DTLgidxFormatSee{⟨tag⟩}{⟨label list⟩}
```

```
\newcommand*{\DTLgidxFormatSee}[2]{%
  \DTLgidxSeeTagFont{ #1 } ~ \DTLgidxSeeList{ #2 }
}
```

\DTLgidxFormatSeeAlso

```
\DTLgidxFormatSeeAlso{⟨tag⟩}{⟨label list⟩}
```

```
\NewDocumentCommand \DTLgidxFormatSeeAlso { m m }
{
  \datagidxdoseealso
  {
    \DTLgidxSeeTagFont{ #1 } ~ \DTLgidxSeeList { #2 }
  }
}
```

\datagidxdoseealso

```
\NewDocumentCommand \datagidxdoseealso { m }
{
  \datagidxseealsostart
   #1
  \datagidxseealsoend
}
```

\DTLgidxSeeList

\DTLgidxSeeList{⟨*label list*⟩}

```
\NewDocumentCommand \DTLgidxSeeList { m }
{
 \group_begin:
  \tl_clear:N \l__datagidx_sep_tl
  \exp_args:NNo \seq_set_from_clist:Nn
    \l__datatool_tmp_seq { #1 }
  \int_set:Nn \l__datatool_count_int
    { \seq_count:N \l__datatool_tmp_seq }
  \seq_map_indexed_inline:Nn \l__datatool_tmp_seq
  {
    \tl_set:Nn \l_datatool_label_tl { ##2 }
    \int_compare:nNnTF
       { ##1 } = { \l__datatool_count_int }
     {
```

Last iteration.

```
       \tl_if_empty:NF \l__datagidx_sep_tl
        {
```

Not the only element in the list so separator needed.

```
          \DTLidxSeeLastSep
        }
      }
      {
```

Not last iteration

```
        \l__datagidx_sep_tl
        \tl_set_eq:NN \l__datagidx_sep_tl \DTLidxSeeSep
      }
    \DTLidxFormatSeeItem { \l_datatool_label_tl }
  }
 \group_end:
}
```

\DTLidxFormatSeeItem

\DTLidxFormatSeeItem{⟨*label*⟩}

```
\NewDocumentCommand \DTLidxFormatSeeItem { m }
{
 \group_begin:
   \DTLgidxFetchEntry
     \l__datagidx_value_tl { #1 } { Name }
   \datatool_if_null:NTF \l__datagidx_value_tl
   {
     \bool_if:NT \l__datagidx_warn_bool
      {
        \PackageWarning { datagidx }
         { Can't ~ find ~ cross-reference ~ `#1' }
      }
   }
   {
     \datagidxlink { #1 }
      { \l__datagidx_value_tl }
   }
 \group_end:
}
```

`\DTLidxSeeSep` Separator in cross-reference list.

```
\newcommand*{\DTLidxSeeSep}{ , ~ }
```

`\DTLidxSeeLastSep` Final separator in cross-reference list.

```
\newcommand*{\DTLidxSeeLastSep}{ ~ \& ~ }
```

`\DTLgidxDoSeeOrLocation` You shouldn't have both a "see" list and a location list. This checks if `\See` is null. If it isn't null, it does the "see" part, otherwise it deals with the location list.

```
\newrobustcmd* \DTLgidxDoSeeOrLocation
{
  \datatool_if_null_or_empty:NTF \See
  {
```

`\See` is null. Do we have a location?

```
    \datatool_if_null_or_empty:NF \Location
     {
       \tl_if_empty:NF \DTLgidxLocation
        {
          \DTLgidxPreLocation
          \DTLgidxLocation
          \DTLgidxPostLocation
        }
     }
  }
  {
```

`\See` is not null, so do the cross-reference.

```
    \DTLgidxSee
  }
}
```

`dtlgidxchildlist` (*env.*) Provided to allow for any font changes etc that might be required.

```
\newenvironment{dtlgidxchildlist}{}{}
```

`\DTLgidxChildren` How to display the children

```
\newcommand*{\DTLgidxChildren}{
  \begin { dtlgidxchildlist }
    \datatool_if_null_or_empty:NF \Children
    {
      \int_incr:N \datagidx@level
      \datagidxchildstart
      \let\Parent\Label
      \datagidx@foreachchild
      {
        \DTLaction
        [
          name = \DTLgidxCurrentdb ,
          key = Label,
          expand-value = \Label ,
          return =
           {
             \Location = Location ,
             \See = See ,
             \SeeAlso = SeeAlso
           }
        ]
        { select ~ row }
        \bool_lazy_all:nF
            {
             { \datatool_if_null_or_empty_p:N \Location }
             { \datatool_if_null_or_empty_p:N \See }
             { \datatool_if_null_or_empty_p:N \SeeAlso }
            }
          {
            \datagidx@displaychild
          }
      }
      \datagidxchildend
    }
  \end { dtlgidxchildlist }
}
```

`\datagidxgetchildfields` Get the child fields from the current row.

```
\newcommand*{\datagidxgetchildfields}{%
  \dtlgetentryfromcurrentrow
    {\Name}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Name}}%
  \dtlgetentryfromcurrentrow
    {\Description}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Description}}%
```

```
    \dtlgetentryfromcurrentrow
      {\Symbol}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Symbol}}%
    \dtlgetentryfromcurrentrow
      {\Long}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Long}}%
    \dtlgetentryfromcurrentrow
      {\Short}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Short}}%
    \dtlgetentryfromcurrentrow
      {\Text}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Text}}%
    \dtlgetentryfromcurrentrow
      {\Plural}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Plural}}%
    \dtlgetentryfromcurrentrow
      {\Used}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Used}}%
    \dtlgetentryfromcurrentrow
      {\Children}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Child}}%
  }
```

`\datagidx@displaychild`

```
  \newcommand*{\datagidx@displaychild}{%
    \datagidxgetchildfields
    \datagidxchilditem
  }
```

`\datagidx@heading` Indicates how to format the heading in the glossary/index. Version 3.0: replaced `\datagidx@heading` with:

```
\tl_new:N \l__datagidx_heading_tl
\cs_if_exist:NTF \chapter
{
  \tl_set:Nn \l__datagidx_heading_tl { \chapter* }
}
{
  \tl_set:Nn \l__datagidx_heading_tl { \section* }
}
```

`\DTLgidxNoHeading` Allow user to suppress the heading. (So to suppress the heading do `heading=\DTLgidxNoHeading`).

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

`\datagidx@postheading` Indicates what to do immediately after the heading. Version 3.0: replaced `\datagidx@postheading` with:

```
\tl_new:N \l__datagidx_post_heading_tl
```

`\datagidx@multicols` Should we use multicols or multicols*? Version 3.0: replaced `\datagidx@multicols` with:

```
\tl_new:N \l__datagidx_multicols_tl
\tl_set:Nn \l__datagidx_multicols_tl { multicols }
```

`\datagidx@sort` Indicates how to sort the glossary/index. Defaults to word order. The expansion of this token list variable is added to the `datagidx` database (in the Sort column), so it needs to be expanded (and is therefore a token list variable not a function). Version 3.0: replaced `\dtlsort` with `\DTLsortdata` replaced `\datagidx@sort` with:

```
\tl_new:N \l__datagidx_sort_tl
\tl_set:Nn \l__datagidx_sort_tl
 {
    \DTLsortdata
     [ save-group-key = LetterGroup ]
     { \DTLgidxCurrentdb }
     {HierSort={replacements=Sort},FirstId}
 }
```

`\@idxitem` Some classes, such as beamer, don't define `\@idxitem` so if it's not already defined, define it here.

```
\providecommand{\@idxitem}{\par\hangindent 40\p@}
```

`\datagidxstart` Indicates what to do at the start of the glossary/index.

```
\newcommand* \datagidxstart
{
  \group_begin:
  \dim_zero:N \parindent
  \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
  \let\item\@idxitem
}
```

`\datagidxend` Indicates what to do at the end of the glossary/index.

```
\newcommand*{\datagidxend} { \datagidx_end: }
\cs_new:Nn \datagidx_end: { \expandafter \group_end: \if@endpe\@doendpe\fi }
```

`\datagidxtarget` Provide a means to add a hypertarget if `\hypertarget` has been defined.

```
\newcommand*{\datagidxtarget}{ \__datagidx_target:nn }
```

Version 3.0: replaced `\@datagidxtarget` with:

```
\cs_new:Nn \__datagidx_target:nn
{
  \cs_if_exist:NT \hypertarget
  {
    \group_begin:
      \datatool_measure_height:Nn
        \l__datatool_tmpa_dim { #2 }
      \raisebox
        { \l__datatool_tmpa_dim }
        { \hypertarget { #1 } { } }
    \group_end:
  }
```

```
  #2
}
```

`\datagidxlink` Provide a means to add a link if `\hyperlink` has been defined.

```
\newcommand*{\datagidxlink}{ \__datagidx_link:nn }
```

Version 3.0: replaced `\@datagidxlink` with:

```
\cs_new:Nn \__datagidx_link:nn
{
  \cs_if_exist:NTF \hyperlink
   { \hyperlink { #1 } { #2 } }
   { #2 }
}
```

`\DTLgidxEnableHyper` Enable hyperlinks (if they are defined).

```
\NewDocumentCommand \DTLgidxEnableHyper { }
{
  \cs_set_eq:NN \datagidxtarget \__datagidx_target:nn
  \cs_set_eq:NN \datagidxlink \__datagidx_link:nn
}
```

`\DTLgidxDisableHyper` Disable hyperlinks (if they are defined).

```
\NewDocumentCommand \DTLgidxDisableHyper { }
{
  \cs_set_eq:NN \datagidxtarget \use_ii:nn
  \cs_set_eq:NN \datagidxlink \use_ii:nn
}
```

`\datagidxgroupsep` Indicates what to do between groups (after the previous group and before the header of the next group).

```
\newcommand*{\datagidxgroupsep}{}
```

`\datagidxgroupheader` Indicates what to do at the start of a group. (The current group label can be accessed via `\datagidxcurrentgroup` and the previous group label can be accessed via `\datagidxprevgroup`.)

```
\newcommand*{\datagidxgroupheader}{}
```

`\datagidxitem` Indicates what to do at the start of each item of the glossary/index.

```
\newcommand*{\datagidxitem}{}%
```

`\datagidxchildstart` Indicates what to do at the start of the child glossary/index.

```
\newcommand*{\datagidxchildstart}{}
```

`\datagidxchildend` Indicates what to do at the end of the child glossary/index.

```
\newcommand*{\datagidxchildend}{}
```

`\datagidxchilditem` Indicates what to do at the start of each item of the child glossary/index.

```
\newcommand*{\datagidxchilditem}{}
```

\datagidxseealsostart Indicates what to do at the start of the “see also” list.

```
\newcommand*{\datagidxseealsostart}{}
```

\datagidxseealsoend Indicates what to do at the end of the “see also” list.

```
\newcommand*{\datagidxseealsoend}{}
```

\DTLgidxEndItem

```
\newcommand{\DTLgidxEndItem}{\par\smallskip}
```

\datagidx@doifsymlocwidth

```
\datagidx@doifsymlocwidth{⟨indent⟩}{⟨Name code⟩}
{⟨Location code⟩}
```

What to do if both the symbol width and the location width have been set. Version 3.0: replaced \datagidx@doifsymlocwidth with:

```
\cs_new:Nn \__datagidx_do_ifsymlocwidth:nnn
{
```

Calculate remaining space left for the description. If L is the linewidth, i is the ⟨*indent*⟩, w_N is the width of ⟨*Name code*⟩, w_S is the symbol width (\datagidxsymbolwidth), w_L is the location width (\datagidxlocationwidth), w_P is the width of \DTLgidxPreLocation, and w_D is the width of \DTLgidxSymDescSep then the available width is $L - i - w_N - w_S - w_L - w_P$.

```
  \datatool_measure_width:Nn
    \l__datatool_tmpa_dim
     { #2 \DTLgidxPreLocation \DTLgidxSymDescSep }
  \dim_set:Nn \l__datatool_tmpa_dim
    {
      \linewidth - #1
      - \l__datatool_tmpa_dim
      - \datagidxsymbolwidth
      - \datagidxlocationwidth
    }
  \bool_if:NTF \l__datagidx_symbol_left_bool
  {
    \begin{minipage} [t]
       { \datagidxsymbolwidth }
      \datagidxsymalign
      \tl_clear:N \DTLgidxSymDescSep
      \DTLgidxSymbolDescLeft
    \end{minipage}
    \DTLgidxSymDescSep
    \begin{minipage} [t]
       { \l__datatool_tmpa_dim }
      \tl_clear:N \DTLgidxSymDescSep
      \DTLgidxSymbolDescRight
    \end{minipage}
  }
```

```
  {
    \begin{minipage} [t]
       { \l__datatool_tmpa_dim }
      \tl_clear:N \DTLgidxSymDescSep
      \DTLgidxSymbolDescRight
    \end{minipage}
    \DTLgidxSymDescSep
    \begin{minipage} [t]
       { \datagidxsymbolwidth }
      \datagidxsymalign
      \tl_clear:N \DTLgidxSymDescSep
      \DTLgidxSymbolDescLeft
    \end{minipage}
  }
  \DTLgidxPreLocation
  \begin{minipage} [t]
     { \datagidxlocationwidth }
    \datagidxlocalign
    \tl_clear:N \DTLgidxPreLocation
    #3
  \end{minipage}
}
```

\datagidx@doiflocwidth

```
\datagidx@doiflocwidth{⟨indent⟩}{⟨Name code⟩}
{⟨Location code⟩}
```

What to do if only the location width has been set. Version 3.0: replaced `\datagidx@doiflocwidth` with:

```
\cs_new:Nn \__datagidx_do_iflocwidth:nnn
{
```

Calculate remaining space left for the symbol and description.

```
  \datatool_measure_width:Nn
    \l__datatool_tmpa_dim
     { #2 \DTLgidxPreLocation }
  \dim_set:Nn \l__datatool_tmpa_dim
    {
      \linewidth - #1
      - \l__datatool_tmpa_dim
      - \datagidxlocationwidth
    }
  \begin{minipage} [t]
      { \l__datatool_tmpa_dim }
    \DTLgidxSymbolDescription
  \end{minipage}
  \DTLgidxPreLocation
  \begin{minipage} [t]
      { \datagidxlocationwidth }
```

```
    \datagidxlocalign
    \tl_clear:N \DTLgidxPreLocation
    #3
  \end{minipage}
}
```

`\datagidx@doifsymwidth` TODO: what uses this?

`\datagidx@doifsymwidth{`⟨*indent*⟩`}{`⟨*Name code*⟩`}{`⟨*Location code*⟩`}`

What to do if only the location width has been set. Version 3.0: replaced `\datagidx@doifsymwidth` with:

```
\cs_new:Nn \__datagidx_do_ifsymwidth:nnn
{
```

Calculate remaining space left for the description and location.

```
  \datatool_measure_width:Nn
    \l__datatool_tmpa_dim
     { #2 \DTLgidxSymDescSep }
  \dim_set:Nn \l__datatool_tmpa_dim
    {
      \linewidth -#1
      - \l__datatool_tmpa_dim
      - \datagidxsymbolwidth
    }
  \bool_if:NTF \l__datagidx_symbol_left_bool
  {
    \begin{minipage} [t]
       { \datagidxsymbolwidth }
      \datagidxsymalign
      \tl_clear:N \DTLgidxSymDescSep
      \DTLgidxSymbolDescLeft
    \end{minipage}
    \DTLgidxSymDescSep
    \begin{minipage} [t]
     { \l__datatool_tmpa_dim }
      \tl_clear:N \DTLgidxSymDescSep
      \DTLgidxSymbolDescRight
      #3
    \end{minipage}
  }
  {
    \begin{minipage} [t]
       { \l__datatool_tmpa_dim }
      \tl_clear:N \DTLgidxSymDescSep
      \DTLgidxSymbolDescRight
    \end{minipage}
    \DTLgidxSymDescSep
```

```
      \begin{minipage} [t]
          { \datagidxsymbolwidth }
        \datagidxsymalign
        \tl_clear:N \DTLgidxSymDescSep
        \DTLgidxSymbolDescLeft
```

This arrangement may look a bit weird.

```
        #3
      \end{minipage}
    }
  }
```

`\datagidxlocalign` Alignment of the location when the location width has been set.

```
\newcommand*{\datagidxlocalign}{\raggedleft}
```

`\datagidxsymalign` Alignment of the symbol when the symbol width has been set.

```
\newcommand*{\datagidxsymalign}{\centering}
```

17.1 Predefined styles

`\datagidxsetstyle` Sets the current index/glossary style

```
\NewDocumentCommand \datagidxsetstyle { m }
{
  \cs_if_exist_use:cF {datagidx@style@#1}
  {
    \PackageError {datagidx} {Unknown ~ style ~ `#1'} {}
  }
}
```

`\datagidxnewstyle` Defines a new index/glossary style

```
\NewDocumentCommand \datagidxnewstyle { m +m }
{
  \cs_if_exist:cTF { datagidx@style@#1 }
   {
     \PackageError { datagidx }
     {
       Style ~ `#1 ' ~ already ~ exists
     }
     { }
   }
   {
     \cs_set:cpn {  datagidx@style@#1 } { #2 }
   }
}
```

17.1.1 index

`\datagidx@style@index` Basic index style.

```
\datagidxnewstyle{index}
```

```
{
  \tl_set:Nn \datagidxstart
  {
    \group_begin:
    \dim_zero:N \parindent
    \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
```

Index columns are usually too narrow for fully justified text.

```
    \raggedright
    \cs_set_eq:NN \item \@idxitem
```

Have the symbol or location widths been set?

```
    \dim_compare:nNnTF
      { \datagidxsymbolwidth } > { \c_zero_dim }
     {
```

Symbol width has been set Has the location width been set?

```
       \dim_compare:nNnTF
          { \datagidxlocationwidth } > { \c_zero_dim }
        {
```

Both have been set.

```
         \tl_set:Nn \datagidx@item@body
          {
            \__datagidx_do_ifsymlocwidth:nnn
             { \c_zero_dim }
             {
               \DTLgidxNameFont { \DTLgidxNameCase { \Name } }
             }
             {
               \DTLgidxDoSeeOrLocation
             }
          }
        }
        {
```

Location width hasn’t been set.

```
         \tl_set:Nn \datagidx@item@body
          {
            \__datagidx_do_iflocwidth:nnn
             { \c_zero_dim }
             {
               \DTLgidxNameFont
                 { \DTLgidxNameCase { \Name } }
             }
             {
               \DTLgidxDoSeeOrLocation
             }
          }
        }
    }
    {
```

Symbol width hasn't been set Has the location width been set?

```
        \dim_compare:nNnTF
         { \datagidxlocationwidth } > { \c_zero_dim }
        {
```

Location width has been set.

```
          \tl_set:Nn \datagidx@item@body
           {
             \__datagidx_do_iflocwidth:nnn
              { \c_zero_dim }
              {
                \DTLgidxNameFont
                 { \DTLgidxNameCase { \Name } }
              }
              {
                \DTLgidxDoSeeOrLocation
              }
           }
        }
        {
```

Neither have been set.

```
          \tl_set:Nn \datagidx@item@body
           {
            \DTLgidxSymbolDescription
            \DTLgidxDoSeeOrLocation
           }
        }
     }
   }
   \tl_set:Nn \datagidxend { \datagidx_end: }
   \tl_set:Nn \datagidxgroupsep
    { \ifdatagidxshowgroups \indexspace \fi }
   \tl_set:Nn \datagidxgroupheader
    {
     \legacy_if:nT { datagidxshowgroups }
      {
       \item
        \makebox [ \linewidth ]
        {
          \textbf
           {
             \DTLgidxGroupHeaderTitle { \datagidxcurrentgroup }
           }
        }
        \DTLpar\nobreak\@afterheading
      }
    }
   \tl_set:Nn \datagidxitem
    {
```

Is this the start of a new group?

```
      \tl_if_empty:NTF \datagidxprevgroup
      {
```

First item of the list.

```
        \datagidxgroupheader
      }
      {
```

Not the first item of the list. Is this item's group the same as the last item's group? Do nothing if the same.

```
        \tl_if_eq:NNF \datagidxcurrentgroup \datagidxprevgroup
        {
```

Different, so do the separator and the header.

```
          \datagidxgroupsep
          \datagidxgroupheader
        }
      }
```

Now get on with this item.

```
      \item
      \datagidxtarget { \Label }
      {
        \DTLgidxNameFont { \DTLgidxNameCase { \Name } }
      }
      \DTLgidxPostName
      \datagidx@item@body
      \DTLgidxChildrenSeeAlso
      \DTLgidxEndItem
    }
  \tl_set:Nn \datagidxchildstart
   {
     \group_begin:
     \dim_zero:N \parindent
     \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
     \cs_set_eq:NN \item \@idxitem
   }
  \tl_set:Nn \datagidxchildend { \group_end: }
  \tl_set:Nn \datagidxchilditem
   {
    \dim_set:Nn \l__datatool_tmpa_dim
      {
         \datagidx@level \datagidxindent
      }
    \@idxitem
    \hspace* { \l__datatool_tmpa_dim }
    \refstepcounter {DTLgidxChildCount}
    \datagidxtarget { \Label }
    {
      \DTLgidxChildStyle
```

```
        {
          \DTLgidxNameFont { \DTLgidxNameCase { \Name } }
          \DTLgidxPostChildName
        }
      }
      \DTLgidxSymbolDescription
      \DTLgidxDoSeeOrLocation
      \DTLgidxChildrenSeeAlso
    }
    \tl_set:Nn \datagidxseealsostart
     {
      \group_begin:
        \dim_zero:N \parindent
        \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
        \int_incr:N \datagidx@level
        \dim_set:Nn \l__datatool_tmpa_dim
         {
           \datagidx@level \datagidxindent
         }
        \@idxitem
        \hspace* { \l__datatool_tmpa_dim }
     }
    \tl_set:Nn \datagidxseealsoend { \group_end: }
  }
```

Make this the default style:

```
  \datagidxsetstyle { index }
  \dim_new:N \l__datagidx_childindent_dim
```

`\datagidxmapdata`

```
  \NewDocumentCommand \datagidxmapdata { m }
  {
   \DTLmapdata [ name = \DTLgidxCurrentdb , read-only ]
    {
      \exp_args:NV \__datatool_map_get_values_noerr:n
         \DTLgidxAssignList
      \__datagidx_filter:T
       {
         \__datagidx_unwrap_location:N \Location
         #1
       }
    }
  }
```

17.1.2 indexalign

Similar to index style but aligns the descriptions.

`atagidx@style@indexalign`

```
\datagidxnewstyle { indexalign }
{
  \tl_set:Nn \datagidxstart
   {
    \group_begin:
    \dim_zero:N \parindent
    \dim_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
    \dim_zero:N \datagidxnamewidth
    \datagidxmapdata
     {
       \datatool_if_null:NT \Parent
       {
         \datagidx@doifdisplayed
         {
           \datatool_measure_width:Nn
             \l__datatool_tmpa_dim
            {
              \DTLgidxNameFont { \DTLgidxNameCase { \Name } }
            }
           \dim_compare:nNnT
             { \l__datatool_tmpa_dim }
              >
             { \datagidxnamewidth }
            {
              \dim_set_eq:NN
                \datagidxnamewidth
                \l__datatool_tmpa_dim
            }
         }
       }
     }
    \datatool_measure_width:Nn
      \l__datatool_tmpa_dim { \DTLgidxPostName }
    \dim_add:Nn \datagidxnamewidth
      {
        \l__datatool_tmpa_dim
      }
    \dim_set:Nn \datagidxdescwidth
      {
        \linewidth - \datagidxnamewidth
      }
    \dim_compare:nNnT
       { \datagidxsymbolwidth } > { \c_zero_dim }
     {
       \datatool_measure_width:Nn
         \l__datatool_tmpa_dim { \DTLgidxSymDescSep }
       \dim_sub:Nn \datagidxdescwidth
        {
           \datagidxsymbolwidth
         + \l__datatool_tmpa_dim
```

```
    }
  }
 \dim_compare:nNnT
    { \datagidxlocationwidth } > { \c_zero_dim }
  {
    \datatool_measure_width:Nn
      \l__datatool_tmpa_dim { \DTLgidxPreLocation }
    \dim_sub:Nn \datagidxdescwidth
     {
       \datagidxlocationwidth
       + \l__datatool_tmpa_dim
     }
  }
```

Has the symbol width been set?

```
 \dim_compare:nNnTF
   { \datagidxsymbolwidth } > { \c_zero_dim }
  {
```

Yes, symbol width has been set. Has the location width been set?

```
  \dim_compare:nNnTF
    { \datagidxlocationwidth } > { \c_zero_dim }
   {
```

Both symbol and location widths have been set.

```
   \bool_if:NTF \l__datagidx_symbol_left_bool
    {
```

Symbol is on the left.

```
     \tl_set:Nn \datagidx@item@body
      {
       \begin{minipage} [t]
          { \datagidxsymbolwidth }
         \datagidxsymalign
         \tl_clear:N \DTLgidxSymDescSep
         \DTLgidxSymbolDescLeft
       \end{minipage}
       \DTLgidxSymDescSep
       \begin{minipage} [t]
          { \datagidxdescwidth }
         \tl_clear:N \DTLgidxSymDescSep
         \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
         \DTLgidxSymbolDescRight
       \end{minipage}
       \DTLgidxPreLocation
       \begin{minipage} [t]
          { \datagidxlocationwidth }
         \datagidxlocalign
         \tl_clear:N \DTLgidxPreLocation
         \DTLgidxDoSeeOrLocation
       \end{minipage}
```

```
    }
  }
  {
```

Symbol is on the right.

```
   \tl_set:Nn \datagidx@item@body
    {
     \begin{minipage} [t]
        { \datagidxdescwidth }
       \tl_clear:N \DTLgidxSymDescSep
       \DTLgidxSymbolDescLeft
     \end{minipage}
     \DTLgidxSymDescSep
     \begin{minipage} [t]
        { \datagidxsymbolwidth }
       \datagidxsymalign
       \tl_clear:N \DTLgidxSymDescSep
       \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt}
       \DTLgidxSymbolDescRight
     \end{minipage}
     \DTLgidxPreLocation
     \begin{minipage} [t]
        { \datagidxlocationwidth }
       \datagidxlocalign
       \tl_clear:N \DTLgidxPreLocation
       \DTLgidxDoSeeOrLocation
     \end{minipage}
    }
  }
}
{
```

Location width hasn't been set. (Only symbol width has been set.)

```
 \bool_if:NTF \l__datagidx_symbol_left_bool
  {
   \tl_set:Nn \datagidx@item@body
    {
     \begin{minipage} [t]
        { \datagidxsymbolwidth }
       \datagidxsymalign
       \tl_clear:N \DTLgidxSymDescSep
       \DTLgidxSymbolDescLeft
     \end{minipage}
     \DTLgidxSymDescSep
     \begin{minipage} [t]
        { \datagidxdescwidth }
       \tl_clear:N \DTLgidxSymDescSep
       \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
       \DTLgidxSymbolDescRight
       \DTLgidxDoSeeOrLocation
     \end{minipage}
```

```
          }
        }
        {
```

Symbol is on the right. This combination may look weird.

```
         \tl_set:Nn \datagidx@item@body
          {
           \begin{minipage} [t]
              { \datagidxdescwidth }
             \tl_clear:N \DTLgidxSymDescSep
             \DTLgidxSymbolDescLeft
           \end{minipage}
           \DTLgidxSymDescSep
           \begin{minipage} [t]
              { \datagidxsymbolwidth }
             \datagidxsymalign
             \tl_clear:N \DTLgidxSymDescSep
             \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
             \DTLgidxSymbolDescRight
             \DTLgidxDoSeeOrLocation
           \end{minipage}
          }
        }
      }
      {
```

Symbol width hasn't been set. Has the location width been set?

```
     \dim_compare:nNnTF
       { \datagidxlocationwidth } > { \c_zero_dim }
      {
```

Only location width has been set.

```
       \tl_set:Nn \datagidx@item@body
        {
         \begin{minipage} [t]
            { \datagidxdescwidth }
          \skip_set:Nn \parskip {0pt ~ plus ~ 0.3pt }
          \DTLgidxSymbolDescription
         \end{minipage}
         \DTLgidxPreLocation
         \begin{minipage} [t]
            { \datagidxlocationwidth }
          \datagidxlocalign
          \tl_clear:N \DTLgidxPreLocation
          \DTLgidxDoSeeOrLocation
        }
      }
      {
```

Neither location nor symbol widths have been set.

```
        \tl_set:Nn \datagidx@item@body
         {
          \begin{minipage} [t]
             { \datagidxdescwidth }
           \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
           \DTLgidxSymbolDescription
           \DTLgidxDoSeeOrLocation
          \end{minipage}
         }
       }
     }
   }
  \tl_set:Nn \datagidxend { \datagidx_end: }
  \tl_clear:N \datagidxgroupsep
  \tl_clear:N \datagidxgroupheader
  \tl_set:Nn \datagidxitem
   {
```

Is this the start of a new group?

```
    \tl_if_empty:NTF \datagidxprevgroup
     {
```

First item of the list.

```
      \datagidxgroupheader
     }
     {
```

Not the first item of the list. Is this item's group the same as the last item's group? If the same, do nothing

```
      \tl_if_eq:NNF \datagidxcurrentgroup \datagidxprevgroup
       {
```

Different, so do the separator and the header.

```
        \datagidxgroupsep
        \datagidxgroupheader
       }
     }
```

Get on with this item

```
    \hangindent0pt\relax
    \parindent0pt\relax
    \makebox [ \datagidxnamewidth ] [l]
     {
      \datagidxtarget { \Label }
       {
        \DTLgidxNameFont
          { \DTLgidxNameCase { \Name } }
        \DTLgidxPostName
       }
     }
    \datagidx@item@body
    \par
```

```
  \DTLgidxChildrenSeeAlso
  \DTLgidxEndItem
}
\tl_set:Nn \datagidxchildstart
 {
  \group_begin:
  \dim_set:Nn \l__datatool_tmpa_dim
    { \datagidx@level \datagidxindent }
  \dim_set:Nn \l__datagidx_childindent_dim
    {
      \linewidth -  \l__datatool_tmpa_dim
    }
  \dim_zero:N \parindent
  \dim_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
  \tl_set:Nx \item
    {
      \exp_not:N \parshape = \c_one_int
      \dim_use:N \l__datatool_tmpa_dim \c_space_tl ~
      \dim_use:N \l__datagidx_childindent_dim
    }
  \dim_zero:N \datagidxnamewidth
  \datagidxmapdata
   {
     \datatool_if_null:NT \Parent
      {
        \datagidx@doifdisplayed
         {
           \datatool_measure_width:Nn
             \l__datatool_tmpa_dim
            {
              \DTLgidxChildStyle
               {
                 \DTLgidxNameFont
                  { \DTLgidxNameCase { \Name } }
               }
            }
           \dim_compare:nNnT
             { \l__datatool_tmpa_dim }
               >
             { \datagidxnamewidth }
            {
              \dim_set_eq:NN
                \datagidxnamewidth
                \l__datatool_tmpa_dim
            }
         }
      }
  }
 \datatool_measure_width:Nn
   \l__datatool_tmpa_dim
```

```
      { \DTLgidxChildStyle \DTLgidxPostChildName }
    \dim_add:Nn \datagidxnamewidth
     {
       \l__datatool_tmpa_dim
     }
    \dim_set:Nn \datagidxdescwidth
     {
       \l__datagidx_childindent_dim
       - \datagidxnamewidth
     }
    }
   \tl_set:Nn \datagidxchildend { \group_end: }
   \tl_set:Nn \datagidxchilditem
    {
      \item
      \refstepcounter {DTLgidxChildCount}
      \makebox [ \datagidxnamewidth ] [l]
       {
         \datagidxtarget { \Label }
          {
            \DTLgidxChildStyle
             {
               \DTLgidxNameFont
                { \DTLgidxNameCase {\Name} }
               \DTLgidxPostChildName
             }
          }
       }
      \begin{minipage} [t]
        { \datagidxdescwidth }
       \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
       \DTLgidxSymbolDescription
       \DTLgidxDoSeeOrLocation
       \DTLgidxChildrenSeeAlso
      \end{minipage}
      \par
    }
}
```

`\datagidxindent` Indent used by `index` and `indexalign` styles.

```
\newlength\datagidxindent
\dim_set:Nn \datagidxindent { 10pt }
```

17.1.3 align

`\datagidxnamewidth` Length used by `align` and `indexalign` style name.

```
\newlength\datagidxnamewidth
```

`\datagidxdescwidth` Length used by `align` and `indexalign` style description.

```
\newlength\datagidxdescwidth
```

```
\datagidx@style@align
                        \datagidxnewstyle { align }
                        {
                          \tl_set:Nn \datagidxstart
                           {
                            \group_begin:
                            \dim_zero:N \parindent
                            \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
                            \dim_zero:N \datagidxnamewidth
                            \datagidxmapdata
                             {
                               \datatool_if_null:NT \Parent
                                {
                                 \datagidx@doifdisplayed
                                  {
                                    \datatool_measure_width:Nn
                                     \l__datatool_tmpa_dim
                                      {
                                        \DTLgidxNameFont
                                          { \DTLgidxNameCase { \Name } }
                                      }
                                   \dim_compare:nNnT
                                     { \l__datatool_tmpa_dim }
                                      >
                                     { \datagidxnamewidth }
                                    {
                                      \dim_set_eq:NN
                                        \datagidxnamewidth
                                        \l__datatool_tmpa_dim
                                    }
                                  }
                                }
                             }
                            \datatool_measure_width:Nn
                               \l__datatool_tmpa_dim
                               { \DTLgidxPostName }
                            \dim_add:Nn \datagidxnamewidth
                             {
                               \l__datatool_tmpa_dim
                             }
                            \dim_set:Nn \datagidxdescwidth
                             {
                               \linewidth - \datagidxnamewidth
                             }
                            \dim_compare:nNnT
                               { \datagidxsymbolwidth } > { \c_zero_dim }
                             {
                               \datatool_measure_width:Nn
                                 \l__datatool_tmpa_dim
```

```
      { \DTLgidxSymDescSep }
    \dim_sub:Nn \datagidxdescwidth
      {
        \datagidxsymbolwidth
        + \l__datatool_tmpa_dim
      }
  }
 \dim_compare:nNnT
   { \datagidxlocationwidth } > { \c_zero_dim }
  {
    \datatool_measure_width:Nn
      \l__datatool_tmpa_dim
      { \DTLgidxPreLocation }
    \dim_sub:Nn \datagidxdescwidth
     {
       \datagidxlocationwidth
       + \l__datatool_tmpa_dim
     }
  }
```

Has the symbol width been set?

```
 \dim_compare:nNnTF
   { \datagidxsymbolwidth } > { \c_zero_dim }
  {
```

Yes, symbol width has been set. Has the location width been set?

```
    \dim_compare:nNnTF
      { \datagidxlocationwidth } > { \c_zero_dim }
     {
```

Both symbol and location widths have been set.

```
       \bool_if:NTF \l__datagidx_symbol_left_bool
        {
```

Symbol is on the left.

```
          \tl_set:Nn \datagidx@item@body
           {
             \begin{minipage} [t]
               { \datagidxsymbolwidth }
             \datagidxsymalign
             \tl_clear:N \DTLgidxSymDescSep
             \DTLgidxSymbolDescLeft
             \end{minipage}
             \DTLgidxSymDescSep
             \begin{minipage} [t]
              { \datagidxdescwidth }
             \tl_clear:N \DTLgidxSymDescSep
             \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
             \DTLgidxSymbolDescRight
             \end{minipage}
             \DTLgidxPreLocation
```

```
            \begin{minipage} [t]
               { \datagidxlocationwidth }
              \datagidxlocalign
              \tl_clear:N \DTLgidxPreLocation
              \DTLgidxDoSeeOrLocation
              \DTLgidxChildrenSeeAlso
            \end{minipage}
          }
        }
      {
```

Symbol is on the right.

```
       \tl_set:Nn \datagidx@item@body
        {
         \begin{minipage} [t]
            { \datagidxdescwidth }
           \tl_clear:N \DTLgidxSymDescSep
           \DTLgidxSymbolDescLeft
         \end{minipage}
         \DTLgidxSymDescSep
         \begin{minipage} [t]
            { \datagidxsymbolwidth }
           \datagidxsymalign
           \tl_clear:N \DTLgidxSymDescSep
           \skip_set:N \parskip { 0pt ~ plus ~ 0.3pt }
           \DTLgidxSymbolDescRight
         \end{minipage}
         \DTLgidxPreLocation
         \begin{minipage} [t]
            { \datagidxlocationwidth }
           \datagidxlocalign
           \tl_clear:N \DTLgidxPreLocation
           \DTLgidxDoSeeOrLocation
           \DTLgidxChildrenSeeAlso
         \end{minipage}
        }
      }
    }
    {
```

Location width hasn't been set. (Only symbol width has been set.)

```
    \bool_if:NTF \l__datagidx_symbol_left_bool
     {
      \tl_set:Nn \datagidx@item@body
       {
        \begin{minipage} [t]
           { \datagidxsymbolwidth }
          \datagidxsymalign
          \tl_clear:N \DTLgidxSymDescSep
          \DTLgidxSymbolDescLeft
        \end{minipage}
```

```
            \DTLgidxSymDescSep
            \begin{minipage} [t]
               { \datagidxdescwidth }
              \tl_clear:N \DTLgidxSymDescSep
              \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
              \DTLgidxSymbolDescRight
              \DTLgidxDoSeeOrLocation
              \DTLgidxChildrenSeeAlso
            \end{minipage}
          }
        }
        {
```

Symbol is on the right. This combination may look weird.

```
          \tl_set:Nn \datagidx@item@body
           {
            \begin{minipage} [t]
               { \datagidxdescwidth }
              \tl_clear:N \DTLgidxSymDescSep
              \DTLgidxSymbolDescLeft
            \end{minipage}
            \DTLgidxSymDescSep
            \begin{minipage} [t]
               { \datagidxsymbolwidth }
              \datagidxsymalign
              \tl_clear:N \DTLgidxSymDescSep
              \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
              \DTLgidxSymbolDescRight
              \DTLgidxDoSeeOrLocation
              \DTLgidxChildrenSeeAlso
            \end{minipage}
          }
        }
      }
    }
    {
```

Symbol width hasn't been set. Has the location width been set?

```
      \dim_compare:nNnTF
        { \datagidxlocationwidth } > { \c_zero_dim }
       {
```

Only location width has been set.

```
        \tl_set:Nn \datagidx@item@body
         {
          \begin{minipage} [t]
            { \datagidxdescwidth }
           \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
           \DTLgidxSymbolDescription
          \end{minipage}
          \DTLgidxPreLocation
```

```
          \begin{minipage} [t]
            { \datagidxlocationwidth }
           \datagidxlocalign
           \tl_clear:N \DTLgidxPreLocation
           \DTLgidxDoSeeOrLocation
           \DTLgidxChildrenSeeAlso
          \end{minipage}
         }
        }
        {
```

Neither location nor symbol widths have been set.

```
         \tl_set:Nn \datagidx@item@body
         {
          \begin{minipage} [t]
             { \datagidxdescwidth }
           \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
           \DTLgidxSymbolDescription
           \DTLgidxDoSeeOrLocation
           \DTLgidxChildrenSeeAlso
          \end{minipage}
         }
        }
      }
    }
   \tl_set:Nn \datagidxend { \datagidx_end: }
   \tl_set:Nn \datagidxgroupsep
    { \ifdatagidxshowgroups \indexspace \fi }
   \tl_set:Nn \datagidxgroupheader
    {
     \ifdatagidxshowgroups
       \item
        \makebox [ \linewidth ]
        {
          \textbf
           {
             \DTLgidxGroupHeaderTitle
               { \datagidxcurrentgroup }
           }
        }
        \DTLpar\nobreak\@afterheading
     \fi
    }
   \tl_set:Nn \datagidxitem
    {
```

Is this the start of a new group?

```
     \tl_if_empty:NTF \datagidxprevgroup
      {
```

First item of the list.

```
        \datagidxgroupheader
      }
      {
```
Not the first item of the list. Is this item's group the same as the last item's group? If the same, do nothing.
```
        \tl_if_eq:NNF \datagidxcurrentgroup \datagidxprevgroup
        {
```
Different, so do the separator and the header.
```
          \datagidxgroupsep
          \datagidxgroupheader
        }
      }
     \hangindent \c_zero_dim
     \parindent \c_zero_dim
     \makebox [ \datagidxnamewidth ] [l]
      {
        \datagidxtarget { \Label }
         {
           \DTLgidxNameFont
             { \DTLgidxNameCase {\Name} }
           \DTLgidxPostName
         }
      }
      \datagidx@item@body
      \DTLgidxEndItem
   }
  \tl_set:Nn \datagidxchildstart
   {
    \group_begin:
    \dim_zero:N \parindent
    \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
    \dim_zero:N \datagidxnamewidth
    \datagidxmapdata
     {
       \datatool_if_null:NT \Parent
        {
          \datagidx@doifdisplayed
           {
             \datatool_measure_width:Nn
               \l__datatool_tmpa_dim
              {
                \DTLgidxChildStyle
                 {
                   \DTLgidxNameFont
                    { \DTLgidxNameCase {\Name} }
                 }
              }
             \dim_compare:nNnT
               { \l__datatool_tmpa_dim }
```

```
                >
               { \datagidxnamewidth }
              {
                \dim_set_eq:NN
                  \datagidxnamewidth
                  \l__datatool_tmpa_dim
              }
            }
          }
        }
       \datatool_measure_width:Nn
         \l__datatool_tmpa_dim
        {
          \DTLgidxChildStyle \DTLgidxPostChildName
        }
       \dim_add:Nn \datagidxnamewidth
         { \l__datatool_tmpa_dim }
       \dim_set:Nn \datagidxdescwidth
        {
          \linewidth - \datagidxnamewidth
        }
     }
   \tl_set:Nn \datagidxchildend { \group_end: }
   \tl_set:Nn \datagidxchilditem
    {
     \hangindent \c_zero_dim
     \parindent \c_zero_dim
     \refstepcounter {DTLgidxChildCount}
     \makebox [ \datagidxnamewidth ] [l]
      {
        \datagidxtarget {\Label}
         {
           \DTLgidxChildStyle
            {
              \DTLgidxNameFont
                { \DTLgidxNameCase {\Name} }
              \DTLgidxPostChildName
            }
         }
      }
     \begin{minipage} [t]
        { \datagidxdescwidth }
      \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
      \DTLgidxSymbolDescription
      \DTLgidxDoSeeOrLocation
      \DTLgidxChildrenSeeAlso
     \end{minipage}
     \par
    }
}
```

17.1.4 gloss

```
\datagidx@style@gloss
                        \datagidxnewstyle { gloss }
                        {
                          \tl_set:Nn \datagidxstart
                           {
                            \group_begin:
                            \dim_zero:N \parindent
                            \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
                            \dim_zero:N \datagidxnamewidth
                            \datagidxmapdata
                             {
                               \datatool_if_null:NT \Parent
                                {
                                  \datagidx@doifdisplayed
                                   {
                                     \datatool_measure_width:Nn
                                      \l__datatool_tmpa_dim
                                      {
                                        \DTLgidxNameFont
                                         { \DTLgidxNameCase {\Name} }
                                      }
                                     \dim_compare:nNnT
                                       { \l__datatool_tmpa_dim }
                                         >
                                       { \datagidxnamewidth }
                                      {
                                        \dim_set_eq:NN
                                          \datagidxnamewidth
                                          \l__datatool_tmpa_dim
                                      }
                                   }
                                }
                             }
                            \datatool_measure_width:Nn
                               \l__datatool_tmpa_dim
                              { \DTLgidxPostName }
                            \dim_add:Nn \datagidxnamewidth
                               { \l__datatool_tmpa_dim }
                            \dim_set:Nn \datagidxdescwidth
                             {
                              \linewidth - \datagidxnamewidth
                             }
                           }
                          \tl_set:Nn \datagidxend { \datagidx_end: }
                          \tl_set:Nn \datagidxgroupsep
                           {
                             \ifdatagidxshowgroups \indexspace \fi
                           }
```

```
\tl_set:Nn \datagidxgroupheader
 {
   \ifdatagidxshowgroups
     \item
     \makebox [ \linewidth ]
      {
       \textbf
        {
          \DTLgidxGroupHeaderTitle
            { \datagidxcurrentgroup }
        }
      }
     \DTLpar\nobreak\@afterheading
   \fi
 }
\tl_set:Nn \datagidxitem
 {
```

Is this the start of a new group?

```
   \tl_if_empty:NTF \datagidxprevgroup
    {
```

First item of the list.

```
      \datagidxgroupheader
    }
    {
```

Not the first item of the list. Is this item's group the same as the last item's group? If the same, do nothing.

```
      \tl_if_eq:NNF \datagidxcurrentgroup\datagidxprevgroup
       {
```

Different, so do the separator and the header.

```
         \datagidxgroupsep
         \datagidxgroupheader
       }
    }
   \dim_zero:N \hangindent
   \dim_zero:N \parindent
   \makebox [ \datagidxnamewidth ] [l]
    {
      \datagidxtarget {\Label}
       {
         \DTLgidxNameFont
           { \DTLgidxNameCase {\Name} }
         \DTLgidxPostName
       }
    }
   \begin{minipage} [t]
     { \datagidxdescwidth }
    \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
```

```
      \bool_lazy_all:nTF
         {
          { \tl_if_empty_p:N \Description }
          { \tl_if_empty_p:N \Symbol }
          { \tl_if_empty_p:N \Location }
         }
        {
          \mbox { }
        }
        {
          \DTLgidxSymbolDescription
          \DTLgidxDoSeeOrLocation

        }
       \DTLgidxChildrenSeeAlso
      \end{minipage}
      \DTLgidxEndItem
     }
    \tl_set:Nn \datagidxchildstart
     {
       \group_begin:
       \tl_clear:N \datagidx@childsep
       \setcounter {DTLgidxChildCount} { 0 }
     }
    \tl_set:Nn \datagidxchildend
     { \DTLgidxPostChild \group_end: }
    \tl_set:Nn \datagidxchilditem
     {
       \datagidx@childsep
       \refstepcounter {DTLgidxChildCount}
       \datagidxtarget {\Label}
        {
          \DTLgidxChildStyle
           {
             \DTLgidxNameFont
               { \DTLgidxNameCase {\Name} }
             \DTLgidxPostChildName
           }
        }
       \DTLgidxSymbolDescription
       \DTLgidxDoSeeOrLocation
       \DTLgidxChildrenSeeAlso
       \tl_set_eq:NN \datagidx@childsep \DTLgidxChildSep
     }
  }
```

`\DTLgidxChildSep` Separator between child entries for gloss style.

```
\newcommand*{\DTLgidxChildSep}{ ~ }
```

`\DTLgidxPostChild` What to put at the end of child entries for gloss style.

```
\newcommand*{\DTLgidxPostChild}{}
```

`\DTLgidxDictHead` Group header for dict style.

```
\cs_if_exist:NTF \chapter
{
  \newcommand \DTLgidxDictHead
   {
     \chapter
      {
        \DTLgidxGroupHeaderTitle { \datagidxcurrentgroup }
      }
   }
}
{
  \newcommand \DTLgidxDictHead
   {
     \section
      {
        \DTLgidxGroupHeaderTitle { \datagidxcurrentgroup }
      }
   }
}
```

`\DTLgidxCategoryNameFont` Font used for ‘category’ entries with ‘dict’ style.

```
\newcommand*{\DTLgidxCategoryNameFont}[1]{#1}
```

`\DTLgidxCategorySep` Separator used with ‘dict’ style.

```
\newcommand*{\DTLgidxCategorySep}{\space}
```

`\DTLgidxSubCategorySep` Separator used with ‘dict’ style.

```
\newcommand*{\DTLgidxSubCategorySep}{\space}
```

`\datagidxdictindent` Indent used by ‘dict’ style.

```
\newcommand*{\datagidxdictindent}{1em}
```

`\DTLgidxDictPostItem` What to do at the end of each item in the ‘dict’ style.

```
\newcommand{\DTLgidxDictPostItem}{\DTLgidxEndItem}
```

`\datagidx@style@dict` Dictionary style. This assumes a hierarchical structure where the top level entries have a name. The next level is used to indicate a category, such as “adjective” or “noun”. If there is only one meaning this level also has a description. If there is more than one meaning, each meaning should be a child of the category entry. Only third level entries are numbered. The `child` key is ignored in this style. The symbol is ignored. The location and symbols widths are also ignored.

```
\datagidxnewstyle { dict }
{
  \tl_set:Nn \datagidxstart
   {
```

```
      \group_begin:
      \dim_zero:N \parindent
      \dim_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
      \dim_set:Nn \l__datatool_tmpa_dim
       {
         \linewidth - \datagidxdictindent
       }
      \dim_set:Nn \l__datagidx_childindent_dim
       { \datagidxdictindent }
      \tl_set:Nx \datagidxdictparshape
       {
         \exp_not:N \parshape=2 ~ 0pt ~
          \dim_use:N \linewidth \c_space_tl ~
          \dim_use:N \l__datagidx_childindent_dim
          \c_space_tl ~
          \dim_use:N \l__datatool_tmpa_dim
       }
     \int_set_eq:NN \datagidx@level \c_one_int
```

Index columns are usually too narrow for fully justified text.

```
     \raggedright
    }
   \tl_set:Nn \datagidxend { \datagidx_end: }
   \tl_clear:N \datagidxgroupsep
   \tl_set:Nn \datagidxgroupheader
    {
      \ifdatagidxshowgroups
        \datagidxend
        \datagidx@postend
        \DTLgidxDictHead
        \l__datagidx_prestart_tl
        \datagidxstart
      \fi
    }
   \tl_set:Nn \datagidxitem
    {
```

Is this the start of a new group?

```
     \tl_if_empty:NTF \datagidxprevgroup
      {
```

First item of the list.

```
        \datagidxgroupheader
      }
      {
```

Not the first item of the list. Is this item's group the same as the last item's group? If the same, do nothing.

```
       \tl_if_eq:NNF \datagidxcurrentgroup \datagidxprevgroup
        {
```

Different, so do the separator and the header.

```
          \datagidxgroupsep
          \datagidxgroupheader
        }
      }
```

Now get on with this item.

```
     \datagidxdictparshape
     \datagidxtarget {\Label}
     {
       \DTLgidxNameFont
         { \DTLgidxNameCase {\Name} }
     }
     \DTLgidxPostName
```

Initialise category separator to do nothing.

```
     \tl_clear:N \datagidx@catsep
     \tl_clear:N \datagidx@subcatsep
     \DTLgidxSymbolDescription
```

No location list.

```
     \DTLgidxChildrenSeeAlso
     \DTLgidxDictPostItem
    }
   \tl_set:Nn \datagidxchildstart
    { \group_begin: }
   \tl_set:Nn \datagidxchildend { \group_end: }
   \tl_set:Nn \datagidxchilditem
    {
```

Which level are we on?

```
      \int_compare:nNnTF
        { \datagidx@level } = { 2 }
       {
```

Category entry

```
         \datagidx@catsep
         \tl_set_eq:NN
           \datagidx@catsep
           \DTLgidxCategorySep
         \tl_clear:N \datagidx@subcapsep
         \datagidxtarget { \Label }
          {
            \DTLgidxChildStyle
             {
               \DTLgidxCategoryNameFont
                { \DTLgidxNameCase{\Name} }
               \DTLgidxPostChildName
             }
          }
         \setcounter {DTLgidxChildCount} {0}
       }
       {
```

Sub Category entry

```
          \datagidx@subcatsep
          \tl_set_eq:NN
            \datagidx@subcatsep
            \DTLgidxSubCategorySep
          \refstepcounter {DTLgidxChildCount}
          \DTLgidxChildCountLabel
          \DTLgidxPostChildName
        }
       \DTLgidxSymbolDescription
       \DTLgidxDoSeeOrLocation
       \DTLgidxChildrenSeeAlso
     }
    \tl_set:Nn \datagidxseealsostart
     {
      \group_begin:
        \dim_zero:N \parindent
        \skip_set:Nn \parskip { 0pt ~ plus ~ 0.3pt }
        \int_incr:N \datagidx@level
        \dim_set:Nn \l__datatool_tmpa_dim
         {
           \datagidx@level \datagidxindent
         }
        \@idxitem
        \hspace* { \l__datatool_tmpa_dim }
     }
    \tl_set:Nn \datagidxseealsoend { \group_end: }
  }
```

17.2 Location Lists

`\dtldofirstlocation` Only display the first location in the list.

```
\newcommand*{\dtldofirstlocation}{%
 \group_begin:
 \__datagidx_unwrap_location:N \Location
 \clist_map_inline:Nn \Location
 {
    \tl_if_empty:nF { ##1 }
    {
       \tl_if_head_eq_meaning:nNTF { ##1 } [
        {
          \__datagidx_getlocation:wnn ##1
        }
        {
          \__datagidx_getlocation:wnn [ ] { } { }
        }
       \datagidxlink
        { \datagidx@current@target }
        {
```

```
            \__datagidx_formatlocation:VV
             \datagidx@current@format
             \datagidx@current@locationstring
          }
```

Only interested in the first item, so break out of loop.

```
        \clist_map_break:
      }
    }
   \group_end:
  }
```

`\datagidx@formatlocation` Version 3.0: replaced `\datagidx@formatlocation` with:

```
\cs_new:Nn \__datagidx_formatlocation:nn
{
 \tl_if_empty:nTF { #1 }
  { #2 }
  {
    \cs_if_exist:cTF { #1 }
    {
      \use:c { #1 } { #2 }
    }
    {
      \PackageWarning {datagidx}
       { Unknown ~ format ~ `#1' }
      #2
    }
  }
}
\cs_generate_variant:Nn
 \__datagidx_formatlocation:nn
 { VV }
```

`\dtldolocationlist` Display the location list.

```
\NewDocumentCommand \dtldolocationlist { }
{
 \datatool_if_null_or_empty:NF \Location
  {
    \group_begin:
    \int_set:Nn  \l__datagidx_prevloc_int {-1}
    \tl_clear:N \datagidx@prev@locationstring
    \tl_clear:N \datagidx@prev@format
    \tl_clear:N \datagidx@prev@locationformat
    \tl_clear:N \datagidx@prev@prefix
    \tl_clear:N \datagidx@prev@target
    \tl_clear:N \datagidx@location@sep
    \int_set:Nn \l__datagidx_startloc_int {-1}
    \__datagidx_unwrap_location:N \Location
    \clist_map_function:NN
      \Location \datagidx@parse@location
```

```
    \__datagidx_do_prev_location: % tidy up loose ends
    \group_end:
  }
}
```

`\if@dtl@sequential` Conditional to keep track of sequences. Version 3.0: replaced `\if@dtl@sequential` with:

```
\bool_new:N \l__datagidx_sequential_bool
```

`\datagidx@getlocdo` Handler for `\datagidx@docomplist` Version 3.0: removed

Assign a numeric value to a location:

```
\regex_const:Nn \c__datagidx_location_int_regex
  { ( \d+ ) \Z }
\regex_const:Nn \c__datagidx_location_roman_regex
  { ( [ I V X L C D M ]+ | [ i v x l c d m ]+ ) \Z }
\regex_const:Nn \c__datagidx_location_alph_regex
  { ( [ a-z A-Z ] ) \Z }
\cs_new:Nn \__datagidx_get_location_num:Nn
{
  \DTLifint { #2 }
   {
     \int_set:Nn #1 { \l__datatool_datum_value_tl }
   }
   {
     \regex_extract_once:NnNTF
       \c__datagidx_location_int_regex { #2 }
        \l__datatool_tmpb_seq
     {
       \exp_args:NNx \int_set:Nn #1
         { \seq_item:Nn \l__datatool_tmpb_seq { 2 } }
     }
     {
        \regex_extract_once:NnNTF
          \c__datagidx_location_roman_regex { #2 }
           \l__datatool_tmpb_seq
        {
          \exp_args:NNx \tl_set:Nn
            \l__datatool_tmpb_tl
             {
               \exp_not:N \int_from_roman:n
                {
                  \seq_item:Nn \l__datatool_tmpb_seq { 2 }
                }
             }
          \int_set:Nn #1 { \l__datatool_tmpb_tl }
        }
        {
          \regex_extract_once:NnNTF
            \c__datagidx_location_alph_regex { #2 }
```

```
                \l__datatool_tmpb_seq
              {
                \exp_args:NNx \tl_set:Nn
                  \l__datatool_tmpb_tl
                  {
                    \exp_not:N \int_from_alph:n
                     {
                       \seq_item:Nn \l__datatool_tmpb_seq { 2 }
                     }
                  }
                \int_set:Nn #1 { \l__datatool_tmpb_tl }
              }
              {
                \int_set:Nn #1 { -1 }
              }
            }
          }
        }
      }
    \cs_generate_variant:Nn  \__datagidx_get_location_num:Nn
     { NV }
    \int_new:N \l__datagidx_prevloc_int
    \int_new:N \l__datagidx_currentloc_int
```

\datagidx@location@start Version 3.0: replaced `\datagidx@location@start` with:

```
\int_new:N \l__datagidx_startloc_int
```

\datagidx@getlocation Syntax: `[`⟨*format*⟩`]{`⟨*location*⟩`}{`⟨*anchor*⟩`}` Get the location and store in `\current@location`:

```
\cs_new:Npn \__datagidx_getlocation:wnn
  [ #1 ] #2 #3
{
```

Store the original value.

```
  \tl_set:Nn \datagidx@current@locationstring { #2 }
```

Store the format:

```
  \tl_set:Nn \datagidx@current@format { #1 }
```

Store the target:

```
  \tl_set:Nn \datagidx@current@target { #3 }
  \tl_if_empty:nTF { #2 }
  {
    \tl_clear:N \datagidx@current@prefix
    \tl_clear:N \datagidx@current@location
  }
  {
```

If the location contains a compositor, we need to get the final element and store the rest as a prefix:

```
    \seq_set_split:NVn \l__datatool_tmpa_seq
      \l__datagidx_compositor_tl
```

```
      { #2 }
    \seq_pop_right:NN \l__datatool_tmpa_seq
       \datagidx@current@location
    \tl_set:Nx \datagidx@current@prefix
      {
        \seq_use:Nn \l__datatool_tmpa_seq
          \l__datagidx_compositor_tl
      }
  }
}
```

`\datagidx@parse@location` Parses the location list (given in the argument).

```
\newcommand*{\datagidx@parse@location}[1]{
  \tl_set:Nn \l__datagidx_location_tl { #1 }
  \__datagidx_unwrap_location:N
    \l__datagidx_location_tl
```

Parse location format.

```
  \exp_args:NV \tl_if_head_eq_meaning:nNTF
   \l__datagidx_location_tl [
  {
    \exp_after:wN \__datagidx_getlocation:wnn
      \l__datagidx_location_tl
  }
  {
    \__datagidx_getlocation:wnn [ ] { } { }
  }
```

If this is the same as the previous location, do nothing.

```
  \tl_if_eq:NNTF
    \datagidx@prev@locationstring
    \datagidx@current@locationstring
  {
```

If the format is different, let the non-empty format over-ride the empty format.

```
    \tl_if_eq:NNF
      \datagidx@prev@format
      \datagidx@current@format
    {
      \tl_if_empty:NF \datagidx@current@format
      {
        \tl_if_empty:NTF \datagidx@prev@format
        {
```

Previous format is empty, so update.

```
          \tl_set_eq:NN
            \datagidx@prev@format
            \datagidx@current@format
        }
        {
           \PackageWarning {datagidx}
```

```
                {
                   Conflicting ~ location ~ formats ~
                   `\datagidx@prev@format' ~ and ~
                   `\datagidx@current@format' ~ for ~
                   location ~ `\datagidx@current@location'
                }
            }
          }
        }
      }
      {
        \@datagidx@parse@location
      }
    }
@datagidx@parse@location
     \newcommand*{\@datagidx@parse@location}{
```

Get a numeric value for this location.

```
   \__datagidx_get_location_num:NV
     \l__datagidx_currentloc_int
     \datagidx@current@location
```

Check if we have a sequence.

```
   \bool_set_true:N \l__datagidx_sequential_bool
```

A change in font format breaks the sequence.

```
   \tl_if_eq:NNTF
      \datagidx@prev@format \datagidx@current@format
    {
```

A change in location format breaks the sequence.

```
     \tl_if_eq:NNTF
       \datagidx@prev@locationformat
       \datagidx@current@locationformat
      {
```

A change in prefix breaks the sequence.

```
        \tl_if_eq:NNF
           \datagidx@prev@prefix \datagidx@current@prefix
         {
```

Prefixes are different, so not a sequence.

```
           \bool_set_false:N \l__datagidx_sequential_bool
         }
      }
      {
```

Formats are different, so not a sequence.

```
         \bool_set_false:N \l__datagidx_sequential_bool
      }
    }
    {
```

Formats are different, so not a sequence.

```
      \bool_set_false:N \l__datagidx_sequential_bool
    }
  \bool_if:NT \l__datagidx_sequential_bool
    {
```

Is this location one more than the previous location?

```
      \int_compare:nNnTF
        { \l__datagidx_prevloc_int + 1 }
          =
        { \l__datagidx_currentloc_int }
      {
```

It is one more than previous value. Is this location the same type as the previous location?

```
          \tl_if_eq:NNTF
            \datagidx@current@locationformat
            \datagidx@prev@locationformat
        {
```

They are the same, so we have a sequence.

```
          \bool_set_true:N \l__datagidx_sequential_bool
        }
        {
```

They aren't the same, so we don't have a sequence.

```
          \bool_set_false:N \l__datagidx_sequential_bool
        }
      }
      {
        \bool_set_false:N \l__datagidx_sequential_bool
      }
    }
```

Has the sequence flag been set?

```
  \bool_if:NTF \l__datagidx_sequential_bool
    {
```

Yes, we have a sequence. Has the start of the sequence been set?

```
       \int_compare:nNnT
        { \l__datagidx_startloc_int } = { -1 }
       {
```

No it hasn't, so set it

```
          \int_set_eq:NN
            \l__datagidx_startloc_int
            \l__datagidx_prevloc_int
          \let\datagidx@location@startval\datagidx@prev@locationstring
          \let\datagidx@location@format\datagidx@prev@format
          \let\datagidx@location@target\datagidx@prev@target
       }
      }
      {
```

We don't have a sequence, so do the previous location.

```
      \__datagidx_do_prev_location:
    }
```

Update previous location macros to this location.

```
  \let\datagidx@prev@location\datagidx@current@location
  \let\datagidx@prev@format\datagidx@current@format
  \let\datagidx@prev@prefix\datagidx@current@prefix
  \let\datagidx@prev@locationformat\datagidx@current@locationformat
  \let\datagidx@prev@locationstring\datagidx@current@locationstring
  \let\datagidx@prev@target\datagidx@current@target
  \int_set_eq:NN
    \l__datagidx_prevloc_int
    \l__datagidx_currentloc_int
}
```

`\DTLgidxLocationSep` Separator between locations.

```
\newcommand*{\DTLgidxLocationSep}{, ~ }
```

`\DTLgidxLocationF` How to format a location list consisting of only two locations.

```
\newcommand*{\DTLgidxLocationF}[2]{%
  #1\DTLgidxLocationSep#2%
}
```

`\DTLgidxLocationFF` How to format a location list consisting of three or more locations.

```
\newcommand*{\DTLgidxLocationFF}[2]{%
  #1--#2%
}
```

`\do@prevlocation` Do the previous location in the current list. Version 3.0: replaced `\do@prevlocation` with:

```
\cs_new:Nn \__datagidx_do_prev_location:
{
```

Have we come to the end of a sequence?

```
  \int_compare:nNnTF
    { \l__datagidx_startloc_int} = { -1 }
  {
```

Not the end of a sequence.

```
    \tl_if_empty:NF \datagidx@prev@locationstring
    {
      \datagidx@location@sep
      \datagidxlink{\datagidx@prev@target}%
      {
        \__datagidx_formatlocation:VV
          \datagidx@prev@format
          \datagidx@prev@locationstring
      }
      \def\datagidx@location@sep{\DTLgidxLocationSep}%
```

```
      }
    }
    {
```

At the end of a sequence.

```
      \datagidx@location@sep
      \do@locrange
      \def\datagidx@location@sep{\DTLgidxLocationSep}%
      \int_set:Nn \l__datagidx_startloc_int { -1 }
    }%
  }
```

`\do@locrange` Format the location range.

```
\newcommand*{\do@locrange}{%
```

Are the start and end locations 2 or more apart?

```
  \int_compare:nNnTF
   {\l__datagidx_prevloc_int}
     >
   { \l__datagidx_startloc_int + 1}%
  {%
```

Yes, they are, so form a range:

```
      \DTLgidxLocationFF
        {%
          \datagidxlink{\datagidx@location@target}%
          {%
            \__datagidx_formatlocation:VV
              \datagidx@location@format
              \datagidx@location@startval
          }%
        }%
        {%
          \datagidxlink{\datagidx@prev@target}%
          {%
            \__datagidx_formatlocation:VV
              \datagidx@prev@format
              \datagidx@prev@locationstring
          }%
        }%
  }%
  {%
```

No, they aren’t so don’t form a range:

```
      \DTLgidxLocationF
        {%
          \datagidxlink{\datagidx@location@target}%
          {%
            \__datagidx_formatlocation:VV
              \datagidx@location@format
              \datagidx@location@startval
```

```
      }%
    }%
    {%
      \datagidxlink{\datagidx@prev@target}%
      {%
        \__datagidx_formatlocation:VV
          \datagidx@prev@format
          \datagidx@prev@locationstring
      }%
    }%
  }%
}
```

18 Defining New Glossary/Index Databases

`datagidx@defaultdatabase` The default database to which terms should be added. Version 3.0: replaced `\datagidx@defaultdatabase` with:

```
\tl_new:N \l__datagidx_default_database_tl
```

`\DTLgidxSetDefaultDB` Allow user to set the default database. Note that the name isn't expanded yet to allow for a placeholder command.

```
\newcommand*{\DTLgidxSetDefaultDB}[1]{%
  \tl_set:Nn \l__datagidx_default_database_tl { #1 }
}
```

`\ifdatagidxbalance`

```
\newif\ifdatagidxbalance
\datagidxbalancetrue
```

Default style is 'index':

```
\newcommand*{\datagidx@style}{index}
```

Define conditional to determine whether or not to show group headers and do sep. (Default is false.)

`\ifdatagidxshowgroups`

```
\newif\ifdatagidxshowgroups
\newcommand*{\datagidx@showgroups}{false}
```

`\newgidx`

```
\newgloss[⟨options⟩]{⟨database name⟩}{⟨title⟩}
```

Define `\newgidx` if it hasn't already been defined by the 'highopt' optimize setting.

```
\ifundef\newgidx
{%
  \newcommand*{\newgidx}{\datagidx@newgidx}
}%
{}
```

May only be used in the preamble (otherwise the entries will be undefined when their locations are read from the aux file).

```
\@onlypreamble\newgidx
```

`datagidx@highopt@newgidx` The behaviour of `\newgidx` when the 'highopt' optimize option has been set.

```
\NewDocumentCommand \datagidx@highopt@newgidx { o m m }
{
```

Get the file name:

```
  \tl_set:Nx \datagidx@indexfilename
    { \text_purify:n { \datagidxhighoptfilename{ #2 } } }
```

Has the file been created?

```
  \IfFileExists{\datagidx@indexfilename}%
  {%
```

File does exists. Load it.

```
    \DTLread[format=dbtex] { \datagidx@indexfilename }
```

If the file was created pre v3.0, the new columns will be missing.

```
    \__datagidx_update_loaded_data:n { #2 }
```

Update the 'datagidx' database.

```
    \group_begin:
      \IfValueT { #1 }
       {
         \keys_set_groups:nnn
          { datatool / index } { newgloss } { #1 }
       }
      \datagidx@newgidx@update {#2} {#3}
    \group_end:
  }
  {
```

File doesn't exist. Behave as normal.

```
    \datagidx@newgidx[#1]{#2}{#3}
  }
}
```

`\loadgidx`

```
\loadgidx[⟨options⟩]{⟨filename⟩}{⟨title⟩}
```

Loads a datagidx database.

```
\NewDocumentCommand \loadgidx { O{} m m }
{%
```

Load database:

```
  \DTLread [ format = dbtex ] { #2 }
```

Update the ‘datagidx’ database. (Assume database is already sorted.)

```
   \__datagidx_update_loaded_data:n { \dtllastloadeddb }
   \bgroup
     \keys_set_groups:nnn
      { datatool / index } { newgloss }
       { sort = {} , #1 }
     \expandafter\datagidx@newgidx@update\expandafter
       {\dtllastloadeddb}{#3}%
   \egroup
```

Set this as the default database:

```
   \tl_set_eq:NN
     \l__datagidx_default_database_tl
     \dtllastloadeddb
```

Assign labels to this database.

```
   \dtlforcolumn{\Label}{\dtllastloadeddb}{Label}%
   {%
      \__datagidx_assign_label_db:nn
        { \Label } { \dtllastloadeddb }
   }%
 }
```

May only be used in the preamble (otherwise the entries will be undefined when their locations are read from the aux file).

```
 \@onlypreamble\loadgidx
```

Marker for Used field:

```
 \cs_new:Nn \__datagidx_used:n { #1 }
```

Marker for LetterGroup field:

```
 \cs_new:Nn \__datagidx_letter_group:n { #1 }
```

Make sure the letter group doesn’t expand.

```
 \cs_new:Nn \__datagidx_write_letter_group:n
 {
   \tl_to_str:n { #1 }
 }
```

Marker for Location field:

```
 \cs_new:Nn \__datagidx_location:n { \exp_not:n { #1 } }
```

Extract the location from the special markers.

```
 \cs_new:Nn \__datagidx_unwrap_location:N
 {
   \exp_args:NV
      \tl_if_head_eq_meaning:nNT #1
      { \dtlspecialvalue }
    {
      \group_begin:
       \cs_set_eq:NN \__datagidx_location:n \exp_not:n
       \cs_set_eq:NN \dtlspecialvalue \use:n
```

```
      \exp_args:NNNx
     \group_end:
       \tl_set:Nn #1 { #1 }
  }
}
```

`\datagidx@newgidx` The normal behaviour of `\newgidx`

```
\NewDocumentCommand \datagidx@newgidx { o m m }
{
\bgroup
  \IfValueT { #1 }
   {
```

NB `\keys_set_filter:nnnN` is being deprecated in favour of `\keys_set_exclude_groups:nnn` (l3kernel 2024-01-22).

```
    \keys_set_filter:nnnN
      { datatool / index } { general , print-only } { #1 }
      \l__datagidx_remainder_tl
    \tl_if_empty:NF \l__datagidx_remainder_tl
     {
       \PackageError { datagidx }
        {
          Invalid ~ \token_to_str:N \newgidx \c_space_tl option(s): ~
           \tl_to_str:N \l__datagidx_remainder_tl
        }
        {
          The ~ listed ~ option ~ or ~ options ~ can't ~ be ~ passed ~
          to ~ \token_to_str:N \newgidx . \MessageBreak
          Try ~ \token_to_str:N \DTLsetup
           { index = { \tl_to_str:N \l__datagidx_remainder_tl } } ~
          or ~ pass ~ the ~ option(s) ~ to ~
          \token_to_str:N \printterms \c_space_tl instead
        }
     }
  }
```

If no default database has been identified, set the default to this database.

```
  \tl_if_empty:NT \l__datagidx_default_database_tl
   {
     \tl_gset:Nx \l__datagidx_default_database_tl { #2 }
   }
  \DTLgnewdb{#2}%
  \__datagidx_add_column:nn {#2} {Label}
  \__datagidx_add_column:nnn
    {#2} {Used} { \c_datatool_integer_int }
  \__datagidx_add_column:nn {#2} {Location}
  \__datagidx_add_column:nn {#2} {UnsafeLocation}
  \__datagidx_add_column:nn {#2} {CurrentLocation}
  \__datagidx_add_column:nnn {#2} {FirstId} { \c_datatool_integer_int }
  \__datagidx_add_column:nn {#2} {Name}
```

```
  \__datagidx_add_column:nn {#2} {Text}
  \__datagidx_add_column:nn {#2} {Plural}
  \__datagidx_add_column:nn {#2} {Parent}
  \__datagidx_add_column:nn {#2} {Child}
  \__datagidx_add_column:nn {#2} {Description}
  \__datagidx_add_column:nn {#2} {HierSort}
  \__datagidx_add_column:nn {#2} {Sort}
  \__datagidx_add_column:nn {#2} {LetterGroup}
  \__datagidx_add_column:nn {#2} {Symbol}
  \__datagidx_add_column:nn {#2} {Long}
  \__datagidx_add_column:nn {#2} {LongPlural}
  \__datagidx_add_column:nn {#2} {Short}
  \__datagidx_add_column:nn {#2} {ShortPlural}
  \__datagidx_add_column:nn {#2} {See}
  \__datagidx_add_column:nn {#2} {SeeAlso}
  \datagidx@newgidx@update{#2}{#3}%
\egroup
}
```

Assume all columns contain strings.

```
\cs_new:Nn \__datagidx_add_column:nn
{
  \__datagidx_add_column:nnn
   { #1 } { #2 } { \c_datatool_string_int }
}
\cs_new:Nn \__datagidx_add_column:nnn
{
  \__datatool_add_column_with_header:nxxx
    { #1 } { #2 } { \int_eval:n { #3 } } { #2 }
}
```

Add new fields in the event a pre v3.0 database has been loaded:

```
\cs_new:Nn \__datagidx_update_loaded_data:n
{
  \datatool_if_has_key:nnF { #1 } { HierSort }
  {
    \__datagidx_add_column:nn { #1 } { HierSort }
  }
  \datatool_if_has_key:nnF { #1 } { LetterGroup }
  {
    \__datagidx_add_column:nn { #1 } { LetterGroup }
  }
  \datatool_if_has_key:nnF { #1 } { UnsafeLocation }
  {
    \__datagidx_add_column:nn {#1 } { UnsafeLocation }
  }
}
```

`\datagidx@newgidx@update` Update the ‘datagidx’ database.

```
\newcommand*{\datagidx@newgidx@update}[2]{%
```

```
    \group_begin:
      \bool_set_true:N \l__datatool_db_global_bool
      \dtlexpandnewvalue
      \DTLnewrow {datagidx}
      \DTLnewdbentry {datagidx} {Glossary}
       { \exp_not:n { #1 } }
      \DTLnewdbentry {datagidx} {Title}
       { \exp_not:n { #2 } }
      \DTLnewdbentry {datagidx} {Heading}
       { \exp_not:V \l__datagidx_heading_tl }
      \DTLnewdbentry{datagidx}{PostHeading}
        {\exp_not:V \l__datagidx_post_heading_tl }
      \DTLnewdbentry {datagidx} {MultiCols}
        {\exp_not:V \l__datagidx_multicols_tl }
      \DTLnewdbentry{datagidx}{Sort}
       { \exp_not:V \l__datagidx_sort_tl }
      \DTLnewdbentry{datagidx}{Style}{\expandonce\datagidx@style}%
      \DTLnewdbentry{datagidx}{ShowGroups}{\expandonce\datagidx@showgroups}%
    \group_end:
  }
```

`\printterms@condition` Version 3.0: replaced `\printterms@condition` with a function instead.

```
\cs_new:Npn \__datagidx_filter:T #1 { #1 }

\bool_new:N \l__datagidx_childsort_bool
\bool_set_true:N \l__datagidx_childsort_bool
```

19 General Options

```
\clist_new:N \l__datagidx_styles_clist
```

Define options for use in the `index` setting in `\DTLsetup`:

```
\keys_define:nn { datatool / index }
{
```

The default index database name:

```
  database .tl_set:N =
    \l__datagidx_default_database_tl ,
  database .groups:n = { print-only },
```

Number of columns in the index:

```
  columns .code:n =
    { \DTLgidxSetColumns { #1 } },
  columns .groups:n = { print-only },
```

Symbol width:

```
  symbolwidth .dim_set:N =
    \datagidxsymbolwidth ,
  symbolwidth .groups:n = { print-only },
```

Synonym:

```
  symbol-width .dim_set:N =
```

```
    \datagidxsymbolwidth ,
  symbol-width .groups:n = { print-only },
```

Location width:

```
  locationwidth .dim_set:N =
    \datagidxlocationwidth ,
  locationwidth .groups:n = { print-only },
```

Synonym:

```
  location-width .dim_set:N =
    \datagidxlocationwidth ,
  location-width .groups:n = { print-only },
```

Child style (should the child's name be shown):

```
  child .choice: ,
  child .groups:n = { print-only },
  child / named .code:n =
   { \__datagidx_set_child_style_named: } ,
  child / noname .code:n =
   { \__datagidx_set_child_style_noname: } ,
```

Child sort. NB the child entries will be sorted by HierSort by default, along with all the parent entries. This option Governs whether or not `\datagidx@foreachchild` follows the ordering from the database (which will in the sort order, if the database has been sorted.)

```
  childsort .bool_set:N = \l__datagidx_childsort_bool ,
  childsort .groups:n = { print-only } ,
```

Synonym.

```
  child-sort .bool_set:N = \l__datagidx_childsort_bool ,
  child-sort .groups:n = { print-only } ,
```

Should a case change be applied to the name?

```
  namecase .choice: ,
  namecase .groups:n = { print-only },
  namecase / nochange .code:n =
    { \let \DTLgidxNameCase \use:n } ,
  namecase / uc .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \text_uppercase:n { ##1 } } } ,
  namecase / lc .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \text_lowercase:n { ##1 } } } ,
  namecase / firstuc .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \xmakefirstuc { ##1 } } } ,
  namecase / capitalise .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \xcapitalisewords { ##1 } } } ,
```

Synonym:

```
  name-case .choice: ,
  name-case .groups:n = { print-only },
  name-case / nochange .code:n =
    { \let \DTLgidxNameCase \use:n } ,
  name-case / uc .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \text_uppercase:n { ##1 } } } ,
  name-case / lc .code:n =
```

```
    { \renewcommand \DTLgidxNameCase [ 1 ] { \text_lowercase:n { ##1 } } } ,
  name-case / firstuc .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \xmakefirstuc { ##1 } } } ,
  name-case / capitalise .code:n =
    { \renewcommand \DTLgidxNameCase [ 1 ] { \xcapitalisewords { ##1 } } } ,
```

Font to apply to the name:

```
  namefont .code:n =
    { \renewcommand*\DTLgidxNameFont [ 1 ] { { #1 { ##1 } } } } ,
  namefont .groups:n = { print-only },
```

Synonym:

```
  name-font .code:n =
    { \renewcommand*\DTLgidxNameFont [ 1 ] { { #1 { ##1 } } } } ,
  name-font .groups:n = { print-only },
```

Content to be placed after the name:

```
  postname .code:n =
    { \renewcommand \DTLgidxPostName { #1 } } ,
  postname .groups:n = { print-only },
```

Synonym:

```
  post-name .code:n =
    { \renewcommand \DTLgidxPostName { #1 } } ,
  post-name .groups:n = { print-only },
```

Content to be placed after the description:

```
  postdesc .choice: ,
  postdesc / none .code:n =
   { \tl_clear:N \DTLgidxPostDescription } ,
  postdesc / dot .code:n =
   { \tl_set:Nn \DTLgidxPostDescription { . } } ,
  postdesc .groups:n = { print-only },
```

Synonym:

```
  post-desc .choice: ,
  post-desc / none .code:n =
   { \tl_clear:N \DTLgidxPostDescription } ,
  post-desc / dot .code:n =
   { \tl_set:Nn \DTLgidxPostDescription { . } } ,
  post-desc .groups:n = { print-only },
```

Content to be placed before the location:

```
  prelocation .choice: ,
  prelocation .groups:n = { print-only },
  prelocation / none .code:n =
   { \tl_clear:N \DTLgidxPreLocation } ,
  prelocation / enspace .code:n =
   { \tl_set:Nn \DTLgidxPreLocation { \enspace } } ,
  prelocation / space .code:n =
   { \tl_set:Nn \DTLgidxPreLocation { ~ } } ,
  prelocation / dotfill .code:n =
   { \tl_set:Nn \DTLgidxPreLocation { \dotfill } } ,
  prelocation / hfill .code:n =
```

```
    { \tl_set:Nn \DTLgidxPreLocation { \hfill } } ,
```

Synonym:

```
   pre-location .choice: ,
   pre-location .groups:n = { print-only },
   pre-location / none .code:n =
    { \tl_clear:N \DTLgidxPreLocation } ,
   pre-location / enspace .code:n =
    { \tl_set:Nn \DTLgidxPreLocation { \enspace } } ,
   pre-location / space .code:n =
    { \tl_set:Nn \DTLgidxPreLocation { ~ } } ,
   pre-location / dotfill .code:n =
    { \tl_set:Nn \DTLgidxPreLocation { \dotfill } } ,
   pre-location / hfill .code:n =
    { \tl_set:Nn \DTLgidxPreLocation { \hfill } } ,
```

Location style (don't show, only show first, or show list):

```
   location .choice: ,
   location .groups:n = { print-only },
   location / hide .code:n =
     { \tl_clear:N \DTLgidxLocation },
   location / list .code:n =
     { \tl_set:Nn \DTLgidxLocation { \dtldolocationlist } },
   location / first .code:n =
     { \tl_set:Nn \DTLgidxLocation { \dtldofirstlocation } },
```

Cross-reference (see) style:

```
   see .choice: ,
   see .groups:n = { print-only },
   see / comma .code:n =
     {
       \renewcommand \DTLgidxSee
        {
          \datatool_if_null_or_empty:NF \See
           {
            , ~ \DTLgidxFormatSee { \seename } { \See }
           }
        }
     } ,
   see / brackets .code:n =
     {
       \renewcommand \DTLgidxSee
        {
          \datatool_if_null_or_empty:NF \See
           {
             \space ( \DTLgidxFormatSee { \seename } { \See } )
           }
        }
     } ,
   see / dot .code:n =
     {
       \renewcommand \DTLgidxSee
```

```
    {
      \datatool_if_null_or_empty:NF \See
       {
         .  ~
         \DTLgidxFormatSee
           { \xmakefirstuc { \seename } } { \See }
       }
    }
  } ,
see / space .code:n =
  {
    \renewcommand \DTLgidxSee
     {
       \datatool_if_null_or_empty:NF \See
        {
          \space
          \DTLgidxFormatSee { \seename } { \See }
        }
     }
  } ,
see / nosep .code:n =
  {
    \renewcommand \DTLgidxSee
     {
       \datatool_if_null_or_empty:NF \See
        {
          \DTLgidxFormatSee { \seename } { \See }
        }
     }
  } ,
see / semicolon .code:n =
  {
    \renewcommand \DTLgidxSee
     {
       \datatool_if_null_or_empty:NF \See
        {
          ; ~ \DTLgidxFormatSee { \seename } { \See }
        }
     }
  } ,
see / location .code:n =
  {
    \renewcommand \DTLgidxSee
     {
       \datatool_if_null_or_empty:NF \See
        {
          \DTLgidxPreLocation
          \DTLgidxFormatSee { \seename } { \See }
        }
     }
```

```
     } ,
```
Style of the symbol and description:
```
   symboldesc .choice: ,
   symboldesc .groups:n = { print-only },
   symboldesc / symbol .code:n =
    {
      \__datagix_set_symbol_only:
    } ,
   symboldesc / desc .code:n =
    {
      \__datagix_set_desc_only:
    } ,
   symboldesc / (symbol) ~ desc .code:n =
    {
      \__datagix_set_symbol_paren_desc:
    } ,
   symboldesc / desc ~ (symbol) .code:n =
    {
      \__datagix_set_desc_symbol_paren:
    } ,
   symboldesc / symbol ~ desc .code:n =
    {
      \__datagix_set_symbol_desc:
    } ,
   symboldesc / desc ~ symbol .code:n =
    {
      \__datagix_set_desc_symbol:
    } ,
```
Synonym:
```
   symbol-desc .choice: ,
   symbol-desc .groups:n = { print-only },
   symbol-desc / symbol .code:n =
    {
      \__datagix_set_symbol_only:
    } ,
   symbol-desc / desc .code:n =
    {
      \__datagix_set_desc_only:
    } ,
   symbol-desc / (symbol) ~ desc .code:n =
    {
      \__datagix_set_symbol_paren_desc:
    } ,
   symbol-desc / desc ~ (symbol) .code:n =
    {
      \__datagix_set_desc_symbol_paren:
    } ,
   symbol-desc / symbol ~ desc .code:n =
    {
```

```
      \__datagix_set_symbol_desc:
    } ,
  symbol-desc / desc ~ symbol .code:n =
    {
      \__datagix_set_desc_symbol:
    } ,
```

Compositor:

```
  compositor .tl_set:N = \l__datagidx_compositor_tl ,
  compositor .value_required:n = true ,
  compositor .groups:n = { general } ,
```

Counter:

```
  counter .code:n =
    {
      \__datagidx_set_counter:n { #1 }
    },
  counter .value_required:n = true ,
  counter .groups:n = { general } ,
```

Switch warnings on or off:

```
  warn .bool_set:N = \l__datagidx_warn_bool ,
  warn .groups:n = { general } ,
```

Index heading:

```
  heading .code:n =
    {
      \clist_put_right:Nn \l__datagidx_styles_clist
        { heading = { #1 } }
      \tl_set:Nn \l__datagidx_heading_tl { #1 }
    } ,
  heading .groups:n = { newgloss, print } ,
```

After index heading:

```
  postheading .code:n =
    {
      \clist_put_right:Nn \l__datagidx_styles_clist
        { postheading = { #1 } }
      \tl_set:Nn \l__datagidx_post_heading_tl { #1 }
    },
  postheading .groups:n = { newgloss, print } ,
```

Synonym:

```
  post-heading .code:n =
    {
      \clist_put_right:Nn \l__datagidx_styles_clist
        { postheading = { #1 } }
      \tl_set:Nn \l__datagidx_post_heading_tl { #1 }
    },
  post-heading .groups:n = { newgloss, print } ,
```

Balance columns:

```
  balance .choice: ,
  balance .groups:n = { newgloss, print } ,
```

```
balance / true .code:n =
 {
   \clist_put_right:Nn \l__datagidx_styles_clist
     { balance = true }
   \tl_set:Nn \l__datagidx_multicols_tl { multicols }
   \datagidxbalancetrue
 } ,
balance / false .code:n =
 {
   \clist_put_right:Nn \l__datagidx_styles_clist
     { balance = false }
   \tl_set:Nn \l__datagidx_multicols_tl { multicols* }
   \datagidxbalancefalse
 } ,
balance .default:n = { true } ,
```

Sort. The value should be the code to sort the glossary/index database. The database should be referenced with `\DTLgidxCurrentdb`:

```
sort .code:n =
 {
   \clist_put_right:Nn \l__datagidx_styles_clist
     { sort = { #1 } }
   \tl_set:Nn \l__datagidx_sort_tl { #1 }
 } ,
sort .groups:n = { newgloss, print } ,
```

Style:

```
style .code:n =
  {
    \clist_put_right:Nn \l__datagidx_styles_clist
     { style = { #1 } }
    \tl_set:Nn \datagidx@style { #1 }
  } ,
style .groups:n = { newgloss, print } ,
```

Show groups:

```
showgroups .choice: ,
showgroups .groups:n = { newgloss, print } ,
showgroups / true .code:n =
  {
    \clist_put_right:Nn \l__datagidx_styles_clist
      { showgroups = true }
    \tl_set:Nn \datagidx@showgroups { true }
  } ,
showgroups / false .code:n =
  {
    \clist_put_right:Nn \l__datagidx_styles_clist
      { showgroups = false }
    \tl_set:Nn \datagidx@showgroups { false }
  } ,
showgroups .default:n = { true } ,
```

Synonym:
```
   show-groups .choice: ,
   show-groups .groups:n = { newgloss, print } ,
   show-groups / true .code:n =
     {
       \clist_put_right:Nn \l__datagidx_styles_clist
         { showgroups = true }
       \tl_set:Nn \datagidx@showgroups { true }
     } ,
   show-groups / false .code:n =
     {
       \clist_put_right:Nn \l__datagidx_styles_clist
         { showgroups = false }
       \tl_set:Nn \datagidx@showgroups { false }
     } ,
   show-groups .default:n = { true } ,
```
Condition:
```
   condition .code:n =
     {
       \cs_set:Npn \__datagidx_filter:T ##1
        {
          \ifthenelse { #1 } { ##1 } { }
        }
     } ,
   condition .groups:n = { print-only } ,
   include-if .cs_set:Np = \__datagidx_filter:T #1,
   include-if .groups:n = { print-only } ,
   include-if-fn .code:n =
    {
      \cs_set_eq:NN \__datagidx_filter:T #1
    },
   include-if-fn .groups:n = { print-only } ,
 }
```
NB package options draft, final and optimize not available.

Allow these keys to be set in `\DTLsetup{index={...}}`
```
 \keys_define:nn { datatool }
 {
   index .code:n = { \keys_set:nn { datatool / index } { #1 } }
 }
```
Used to store filtered settings:
```
 \tl_new:N \l__datagidx_remainder_tl
```
20 Defining New Terms

20.1 Options

Options for `\newterm`:

The internal commands all need to be in the form `\l__datagidx_term_`⟨*key*⟩`_tl` to work with `\newtermfield`

`\newterm@label` Version 3.0: replaced `\newterm@label` with:
`\tl_new:N \l__datagidx_term_label_tl`

`\newterm@parent` Version 3.0: replaced `\newterm@parent` with:
`\tl_new:N \l__datagidx_term_parent_tl`

`\newterm@name` Version 3.0: replaced `\newterm@name` with: This doesn't have a corresponding option since the name is the mandatory argument of `\newterm`:
`\tl_new:N \l__datagidx_term_name_tl`

`\newterm@text` Version 3.0: replaced `\newterm@text` with:
`\tl_new:N \l__datagidx_term_text_tl`

`\newterm@description` Version 3.0: replaced `\newterm@description` with:
`\tl_new:N \l__datagidx_term_description_tl`

`\newterm@plural` Version 3.0: replaced `\newterm@plural` with:
`\tl_new:N \l__datagidx_term_plural_tl`

`\newterm@sort` Version 3.0: replaced `\newterm@sort` with:
`\tl_new:N \l__datagidx_term_sort_tl`

`\newterm@symbol` Version 3.0: replaced `\newterm@symbol` with:
`\tl_new:N \l__datagidx_term_symbol_tl`

`\newterm@database` Version 3.0: replaced `\newterm@database` with:
`\tl_new:N \l__datagidx_term_database_tl`

`\newterm@long` Version 3.0: replaced `\newterm@long` with:
`\tl_new:N \l__datagidx_term_long_tl`

`\newterm@short` Version 3.0: replaced `\newterm@short` with:
`\tl_new:N \l__datagidx_term_short_tl`

`\newterm@longplural` Version 3.0: replaced `\newterm@longplural` with:
`\tl_new:N \l__datagidx_term_longplural_tl`

`\newterm@shortplural` Version 3.0: replaced `\newterm@shortplural` with:
`\tl_new:N \l__datagidx_term_shortplural_tl`

`\newterm@see` "see" should not be used with a location list. If you have a location list and want a cross-reference use "see also" instead. Version 3.0: replaced `\newterm@see` with:
`\tl_new:N \l__datagidx_term_see_tl`

`\newterm@seealso` "see also" should be used with a location list (or with child entries with location lists). If an entry has no location list and not child entries use "see" instead. Version 3.0: replaced `\newterm@seealso` with:
`\tl_new:N \l__datagidx_term_seealso_tl`

The following commands allow users to, say, specify the name as ⟨*name*⟩`\glsadd{`⟨*other label*⟩`}` without having to specify the label when defining a term. Assign the label token list variable:

```
\cs_new:Nn \__datagidx_create_label:n
{
 \group_begin:
   \cs_set_eq:NN \glsadd \DTLgidxGobble
```

Strip common formatting commands. NB v3.0: `\text_purify:n` should now deal with these.

```
    \cs_set_eq:NN \MakeUppercase \DTLgidxNoFormat
    \cs_set_eq:NN \MakeTextUppercase \DTLgidxNoFormat
    \cs_set_eq:NN \MakeLowercase \DTLgidxNoFormat
    \cs_set_eq:NN \MakeTextLowercase \DTLgidxNoFormat
    \cs_set_eq:NN \acronymfont \DTLgidxNoFormat
    \cs_set_eq:NN \textrm \DTLgidxNoFormat
    \cs_set_eq:NN \texttt \DTLgidxNoFormat
    \cs_set_eq:NN \textsf \DTLgidxNoFormat
    \cs_set_eq:NN \textsc \DTLgidxNoFormat
    \cs_set_eq:NN \textbf \DTLgidxNoFormat
    \cs_set_eq:NN \textmd \DTLgidxNoFormat
    \cs_set_eq:NN \textit \DTLgidxNoFormat
    \cs_set_eq:NN \textsl \DTLgidxNoFormat
    \cs_set_eq:NN \emph \DTLgidxNoFormat
    \cs_set_eq:NN \textsuperscript \DTLgidxNoFormat
```

Convert character commands like `\&`.

```
   \datagidxconvertchars
```

Strip `\ensuremath`.

```
    \cs_set_eq:NN \ensuremath \DTLgidxNoFormat
```

Ensure that inversions are dealt with for the label.

```
    \cs_set_eq:NN \DTLgidxParen \use_none:n
    \cs_set_eq:NN \DTLgidxName \use_ii:nn
    \cs_set_eq:NN \DTLgidxPlace \datagidx@invert
    \cs_set_eq:NN \DTLgidxSubject \datagidx@invert
    \cs_set_eq:NN \DTLgidxOffice \use_ii:nn
    \cs_set_eq:NN \DTLgidxParticle \datagidx@bothoftwo
    \cs_set_eq:NN \__datagidx_punc:n \use_none:n
```

Convert Greek maths (such as `\alpha`) to text.

```
    \datagidxwordifygreek
```

Allow user to hook into this.

```
    \newtermlabelhook
```

Using protected expansion first allows for the localised change to `\ensuremath`:

```
     \protected@edef \l__datagidx_term_label_tl { #1 }
```

Strip punctuation characters that are likely to cause a problem for syntax parsing.

```
     \regex_replace_all:nnN
```

```
        { [,=] } { } \l__datagidx_term_label_tl
    \tl_set:Nx \l__datagidx_term_label_tl
     {
       \text_purify:n { \l__datagidx_term_label_tl }
     }
   \exp_args:NNNV
  \group_end:
     \tl_set:Nn
       \l__datagidx_term_label_tl
       \l__datagidx_term_label_tl
}
```

Assign the sort token list variable:

```
\cs_new:Nn \__datagidx_create_sort:n
{
 \group_begin:
   \cs_set_eq:NN \glsadd \use_none:n
```

Strip common formatting commands. NB v3.0: `\text_purify:n` should now deal with these.

```
    \cs_set_eq:NN \MakeUppercase \DTLgidxNoFormat
    \cs_set_eq:NN \MakeTextUppercase \DTLgidxNoFormat
    \cs_set_eq:NN \MakeLowercase \DTLgidxNoFormat
    \cs_set_eq:NN \MakeTextLowercase \DTLgidxNoFormat
    \cs_set_eq:NN \acronymfont \DTLgidxNoFormat
    \cs_set_eq:NN \textrm \DTLgidxNoFormat
    \cs_set_eq:NN \texttt \DTLgidxNoFormat
    \cs_set_eq:NN \textsf \DTLgidxNoFormat
    \cs_set_eq:NN \textsc \DTLgidxNoFormat
    \cs_set_eq:NN \textbf \DTLgidxNoFormat
    \cs_set_eq:NN \textmd \DTLgidxNoFormat
    \cs_set_eq:NN \textit \DTLgidxNoFormat
    \cs_set_eq:NN \textsl \DTLgidxNoFormat
    \cs_set_eq:NN \emph \DTLgidxNoFormat
    \cs_set_eq:NN \textsuperscript \DTLgidxNoFormat
```

Convert character commands like `\&`.

```
   \datagidxconvertchars
```

Strip `\ensuremath`.

```
    \cs_set_eq:NN \ensuremath \DTLgidxNoFormat
```

These commands behave differently for the sort key:

```
    \cs_set_eq:NN \__datagidx_punc:n \exp_not:n
    \cs_set_eq:NN \DTLgidxName \datagidx@person
    \cs_set_eq:NN \DTLgidxPlace \datagidx@place
    \cs_set_eq:NN \DTLgidxSubject \datagidx@subject
    \cs_set_eq:NN \DTLgidxOffice \datagidx@person
    \cs_set_eq:NN \DTLgidxParen \datagidx@paren
    \cs_set_eq:NN \DTLgidxMac \datagidx@mac
    \cs_set_eq:NN \DTLgidxSaint \datagidx@saint
    \cs_set_eq:NN \DTLgidxIgnore \@gobble
```

```
    \cs_set_eq:NN \DTLgidxRank \datagidx@rank
    \cs_set_eq:NN \DTLgidxParticle \datagidx@particle
    \cs_set_eq:NN \DTLgidxNameNum \datagidx@namenum
```

Convert Greek maths (such as `\alpha`) to text.

```
    \datagidxwordifygreek
```

Allow user to hook into this.

```
    \newtermsorthook
```

Using protected expansion first allows for the localised change to `\ensuremath`:

```
    \protected@edef \l__datagidx_term_sort_tl { #1 }
  \exp_args:NNNV
 \group_end:
    \tl_set:Nn
      \l__datagidx_term_sort_tl
      \l__datagidx_term_sort_tl
}
```

Variable to store hierarchical sort value.

```
\tl_new:N \l__datagidx_term_hiersort_tl
```

```
\keys_define:nn { datatool / index / newterm }
{
```

Labels:

```
  database .code:n =
   {
     \tl_set:Nx \l__datagidx_term_database_tl
       { #1 }
   },
  label .code:n =
   {
     \__datagidx_create_label:n { #1 }
   } ,
  parent .code:n =
   {
     \tl_set:Nx \l__datagidx_term_parent_tl
       { #1 }
   } ,
```

Token lists:

```
  text .code:n =
    {
      \tl_set:Nn \l__datagidx_term_text_tl { #1 }
      \tl_if_eq:NnT \l__datagidx_term_plural_tl { \c_novalue_tl }
       {
         \tl_set:Nn \l__datagidx_term_plural_tl { #1 s }
       }
    } ,
  description .tl_set:N = \l__datagidx_term_description_tl ,
  plural .tl_set:N = \l__datagidx_term_plural_tl ,
```

```
  sort .code:n =
   {
     \__datagidx_create_sort:n { #1 }
   } ,
  symbol .tl_set:N = \l__datagidx_term_symbol_tl ,
  long .code:n =
   {
     \tl_set:Nn \l__datagidx_term_long_tl { #1 }
     \tl_if_eq:NnT \l__datagidx_term_longplural_tl { \c_novalue_tl }
      {
        \tl_set:Nn \l__datagidx_term_longplural_tl { #1 s }
      }
   } ,
  short .code:n =
   {
     \tl_set:Nn \l__datagidx_term_short_tl { #1 }
     \tl_if_eq:NnT \l__datagidx_term_shortplural_tl { \c_novalue_tl }
      {
        \tl_set:Nn \l__datagidx_term_shortplural_tl { #1 s }
      }
   } ,
  longplural .tl_set:N = \l__datagidx_term_longplural_tl ,
  shortplural .tl_set:N = \l__datagidx_term_shortplural_tl ,
  see .tl_set:N = \l__datagidx_term_see_tl ,
  seealso .tl_set:N = \l__datagidx_term_seealso_tl ,
}
```

20.2 New Terms

`\newterm@defaultshook` Version 3.0: replaced `\newterm@defaultshook` with:

```
\tl_new:N \g__datagidx_newterm_defaults_tl
```

`\newterm@extrafields` Version 3.0: replaced `\newterm@extrafields` with

```
\tl_new:N \g__datagidx_extra_fields_tl
```

`\DTLgidxAssignList` Assignment list used by `\printterms`

```
\newcommand*{\DTLgidxAssignList}{
  \Name=Name,
  \Description=Description,
  \Used=Used,
  \Symbol=Symbol,
  \Long=Long,
  \Short=Short,
  \LongPlural=LongPlural,
  \ShortPlural=ShortPlural,
  \Location=Location,
  \See=See,
  \SeeAlso=SeeAlso,
  \Text=Text,
```

```
  \Plural=Plural,
  \CurrentLocation=CurrentLocation,
  \Label=Label,
  \Parent=Parent,
  \Children=Child,
  \FirstId=FirstId,
  \HierSort=HierSort,
  \Sort=Sort,
  \datagidxcurrentgroup=LetterGroup
}
```

`\datagidxtermkeys` Keys defined for `\newterm` corresponding to fields.

```
\newcommand*{\datagidxtermkeys}{}
```

Access keys corresponding to given fields.
Syntax: `{`⟨*column key*⟩`}{`⟨*term key*⟩`}`
Add term key to list and define mapping:

```
\cs_new:Nn \__datagidx_add_field:nn
{
  \clist_gput_right:Nn \datagidxtermkeys { #2 }
  \tl_gset:cn { g__datagidx_fieldkey_ #1 } { #2 }
  \tl_clear_new:c { l__datagidx_term_ #2 _tl }
}
```

Syntax: `{`⟨*column key*⟩`}{`⟨*term key*⟩`}{`⟨*default val*⟩`}` Add term key to list, define mapping and provide the key as an option:

```
\cs_new:Nn \__datagidx_add_field_option:nnnN
{
  \__datagidx_add_field:nn { #1 } { #2 }
  \keys_define:nn { datatool / index / newterm }
  {
    #2 .tl_set:N = #4
  }
  \tl_put_right:Nn \g__datagidx_newterm_defaults_tl
   {
     \tl_if_eq:NnT #4 { \c_novalue_tl }
      {
        \tl_set:Nx #4 { #3 }
      }
   }
}
\cs_generate_variant:Nn
 \__datagidx_add_field_option:nnnN
 { xxnc }
```

Predefined fields:

```
\__datagidx_add_field:nn { Name } { name }
\__datagidx_add_field:nn { Description} { description }
\__datagidx_add_field:nn { Symbol } { symbol }
\__datagidx_add_field:nn { Long } { long }
```

```
\__datagidx_add_field:nn { Short } { short }
\__datagidx_add_field:nn { See } { see }
\__datagidx_add_field:nn { SeeAlso } { seealso }
\__datagidx_add_field:nn { Text } { text }
\__datagidx_add_field:nn { Plural } { plural }
\__datagidx_add_field:nn { Label } { label }
\__datagidx_add_field:nn { Parent } { parent }
\__datagidx_add_field:nn { Sort } { sort }
```

\newtermaddfield

\newtermaddfield[⟨*db list*⟩]{⟨*column key*⟩}
[⟨*placeholder cs*⟩]{⟨*new term key*⟩}[⟨*data type*⟩]{⟨*default value*⟩}

The default value may contain \field{⟨*key*⟩} to get the value of another field.

```
\NewDocumentCommand \newtermaddfield
  { O{} m o m O { \c_datatool_string_int } m }
{
  \__datagidx_set_label:Nn \l__datagidx_label_tl { #2 }
```

If optional argument not specified, iterate over all defined glossaries/indices.

```
  \tl_if_empty:nTF { #1 }
  {
    \dtlforcolumn
      \l__datagidx_database_tl { datagidx } { Glossary }
    {
      \datatool_if_has_key:nnF
       { \l__datagidx_database_tl } { \l__datagidx_label_tl }
       {
         \__datagidx_add_column:nnn
           { \l__datagidx_database_tl }
           { \l__datagidx_label_tl }
           { #5 }
       }
    }
  }
  {
    \clist_map_inline:nn { #1 }
    {
      \datatool_if_has_key:nnF
       { ##1 } { \l__datagidx_label_tl }
       {
         \__datagidx_add_column:nnn
           { ##1 }
           { \l__datagidx_label_tl }
           { #5 }
       }
    }
  }
```

```
  \__datagidx_add_field_option:xxnc
    { \l__datagidx_label_tl } { #4 } { #6 }
    { l__datagidx_term_ #4 _tl }
  \__datagidx_append_extra_field:cV
    { l__datagidx_term_ #4 _tl }
    \l__datagidx_label_tl
  \IfValueTF { #3 }
  {
    \tl_if_blank:n { #3 }
    {
      \__datagidx_add_placeholder:cV
        { \l__datagidx_label_tl }
        \l__datagidx_label_tl
    }
    {
      \__datagidx_add_placeholder:NV
        #3
        \l__datagidx_label_tl
    }
  }
  {
    \__datagidx_add_placeholder:cV
      { \l__datagidx_label_tl }
      \l__datagidx_label_tl
  }
}
```

Append to extra fields hook.

```
\cs_new:Nn \__datagidx_append_extra_field:Nn
{
  \tl_gput_right:Nn \g__datagidx_extra_fields_tl
   {
     \__datagidx_append_element_with_check:Nn
       #1 { #2 }
   }
}
\cs_generate_variant:Nn
  \__datagidx_append_extra_field:Nn
  { cV }
\cs_new:Nn \__datagidx_add_placeholder:Nn
{
 \tl_if_exist:NTF #1
  {
    \PackageError { datagidx }
    {
      \token_to_str: \newtermaddfield : ~ placeholder ~
      \token_to_str #1 \c_space_tl ~ already ~ exists
    }
    {
```

```
      Choose ~ a ~ different ~ command ~ name ~ in
      \token_to_str: \newtermaddfield [ ... ]
      { #2 } [ \token_to_str #1 ] { ... }
    }
  }
  {
    \clist_gput_right:Nn \DTLgidxAssignList
    {
      #1 = #2
    }
    \tl_gput_right:Nn \datagidxgetchildfields
    {
      \datatool_if_has_key:nnTF
       { \DTLgidxCurrentdb } { #2 }
       {
         \dtlgetentryfromcurrentrow { #1 }
         { \dtlcolumnindex { \DTLgidxCurrentdb } { #2 } }
       }
       {
         \tl_set_eq:NN #1 \dtlnovalue
       }
    }
  }
}
\cs_generate_variant:Nn \__datagidx_add_placeholder:Nn
 { cn, cV, NV }
```

`\newtermlabelhook` Hook used to provide local redefinitions to adjust the label.

```
\newcommand*{\newtermlabelhook}{}
```

`\newtermsorthook` Hook used to provide local redefinitions to adjust the sort.

```
\newcommand*{\newtermsorthook}{}
```

`\DTLgidxNoFormat`

```
\newcommand*{\DTLgidxNoFormat}[1]{#1}
```

`\DTLgidxGobble`

```
\newcommand*{\DTLgidxGobble}[1]{}
```

`\DTLgidxStripBackslash` Argument must be a control sequence. This is stringified and the backslash is removed).

```
\newcommand*{\DTLgidxStripBackslash}[1]{ \cs_to_str:N #1 }
```

`\DTLgidxName`

```
\DTLgidxName{⟨forenames⟩}{⟨surname⟩}
```

How to format a person's name in the text.

```
\newcommand*{\DTLgidxName}[2]{ #1 \c_space_tl #2 }
```

\DTLgidxNameNum

```
\DTLgidxNameNum{⟨n⟩}
```

The argument ⟨*n*⟩ should be a number applied to a name (e.g. `James␣\DTLgidxNameNum1`). This is converted to a two-digit number for sorting but a Roman numeral for the label and in the text.

```
\newcommand*{\DTLgidxNameNum}[1]{\@Roman{#1}}
```

\datagidx@namenum Conversion for sort key.

```
\newcommand*{\datagidx@namenum}[1]{\two@digits{#1}}
```

\DTLgidxPlace

```
\DTLgidxPlace{⟨country⟩}{⟨town/city⟩}
```

How to format a place in the text.

```
\newcommand*{\DTLgidxPlace}[2]{ #2 }
```

\DTLgidxSubject

```
\DTLgidxSubject{⟨main⟩}{⟨category⟩}
```

How to format a subject in the text. Ignore the main part in the text.

```
\newcommand*{\DTLgidxSubject}[2]{ #2 }
```

\DTLgidxOffice

```
\DTLgidxOffice{⟨office⟩}{⟨name⟩}
```

Put the office in parentheses in the document text.

```
\newcommand*{\DTLgidxOffice}[2]{ #2 ~ ( #1 ) }
```

\DTLgidxIgnore Show argument in document text, but disregard in the sort and label.

```
\newcommand*{\DTLgidxIgnore}[1]{#1}
```

\DTLgidxMac

```
\DTLgidxMac{⟨text⟩}
```

In the document, just does ⟨*text*⟩, but gets converted to “Mac” in the sort key. (Unless overridden by the user.)

```
\newcommand*{\DTLgidxMac}[1]{#1}
```

\datagidx@mac `\DTLgidxMac` gets temporarily redefined to `\datagidx@mac` when construction the sort key.

```
\newcommand*{\datagidx@mac}[1]{Mac}
```

\DTLgidxSaint

\DTLgidxSaint{⟨*text*⟩}

In the document, just does ⟨*text*⟩, but gets converted to “Saint” in the sort key. (Unless overridden by the user.)

```
\newcommand*{\DTLgidxSaint}[1]{#1}
```

\datagidx@saint

\DTLgidxMac gets temporarily redefined to \datagidx@saint when construction the sort key.

```
\newcommand*{\datagidx@saint}[1]{Saint}

\ExplSyntaxOff
```

\DTLgidxRank

\DTLgidxRank{⟨*rank*⟩}{⟨*forenames*⟩}

A person’s title, rank or sanctity should be ignored when sorting. This has a non-breakable space separator.

```
\newcommand*{\DTLgidxRank}[2]{#1~#2}
```

\datagidx@rank

\DTLidxRank gets temporarily redefined to \datagidx@rank when constructing the sort key. An extra dot is added to the end to ensure names without a rank are sorted before identical names with a rank.

```
\newcommand*{\datagidx@rank}[2]{#2.}
```

\DTLgidxParticle

\DTLgidxParticle{⟨*particle*⟩}{⟨*surname*⟩}

A particle such as “of”, “de” or “von” should be ignored when sorting. This has a non-breakable space separator.

```
\newcommand*{\DTLgidxParticle}[2]{#1~#2}

\ExplSyntaxOn
```

\datagidx@particle

\DTLidxParticle gets temporarily redefined to \datagidx@particle when constructing the sort key. An extra dot is added to the end to ensure names without a particle are sorted before identical names with a particle.

```
\newcommand*{\datagidx@particle}[2]{#2.}
```

\datagidx@bothoftwo

```
\newcommand*{\datagidx@bothoftwo}[2]{#1#2}

\cs_new:Nn \__datagidx_punc:n { #1 }
```

\datagidx@person

Used when constructing the sort key for a name.

```
\newcommand*{\datagidx@person}[2]{
 #2 \__datagidx_punc:n { \datatoolpersoncomma } #1
}
```

`\datagidx@place` Used when constructing the sort key for a place.

```
\newcommand*{\datagidx@place}[2]{
 #2 \__datagidx_punc:n { \datatoolplacecomma } #1
}
```

`\datagidx@subject` Used when constructing the sort key for a place.

```
\newcommand*{\datagidx@subject}[2]{
 #2 \__datagidx_punc:n { \datatoolsubjectcomma } #1
}
```

`\datagidx@paren` Used when constructing the sort key for a parenthesis.

```
\newcommand*{\datagidx@paren}[1]{
 \__datagidx_punc:n { \datatoolparenstart } #1
}
```

`\datagidx@invert`

```
\newcommand*{\datagidx@invert}[2]{
 #2 \__datagidx_punc:n { , } ~ #1
}
```

`\DTLgidxParen` Parenthetical material.

```
\newcommand*{\DTLgidxParen}[1]{ \c_space_tl ( #1 ) }
```

`\datagidxwordifygreek` Convert commands like `\alpha` into words for indexing and labelling purposes.

```
\newcommand*{\datagidxwordifygreek}{%
  \def\alpha{alpha}%
  \def\beta{beta}%
  \def\gamma{gamma}%
  \def\delta{delta}%
  \def\epsilon{epsilon}%
  \def\varepsilon{epsilon}%
  \def\zeta{zeta}%
  \def\eta{eta}%
  \def\theta{theta}%
  \def\vartheta{theta}%
  \def\iota{iota}%
  \def\kappa{kappa}%
  \def\lambda{lambda}%
  \def\mu{mu}%
  \def\nu{nu}%
  \def\xi{xi}%
  \def\pi{pi}%
  \def\varpi{pi}%
  \def\rho{rho}%
  \def\varrho{rho}%
  \def\sigma{sigma}%
  \def\varsigma{sigma}%
  \def\tau{tau}%
  \def\upsilon{upsilon}%
```

```
  \def\phi{phi}%
  \def\varphi{phi}%
  \def\chi{chi}%
  \def\psi{psi}%
  \def\omega{omega}%
  \def\Gamma{Gamma}%
  \def\Delta{Delta}%
  \def\Theta{Theta}%
  \def\Lambda{Lambda}%
  \def\Xi{Xi}%
  \def\Pi{Pi}%
  \def\Sigma{Sigma}%
  \def\Upsilon{Upsilon}%
  \def\Phi{Phi}%
  \def\Psi{Psi}%
  \def\Omega{Omega}%
}
```

`\datagidxextendedtoascii` Convert commands like `\aa` into the closest ASCII equivalent. Version 3.0: Deprecated (use localisation)

```
\newcommand{\datagidxextendedtoascii}{%
  \def\AE{AE}%
  \def\ae{ae}%
  \def\OE{OE}%
  \def\oe{oe}%
  \def\AA{AA}%
  \def\aa{aa}%
  \def\L{L}%
  \def\l{l}%
  \def\O{O}%
  \def\o{o}%
  \def\SS{SS}%
  \def\ss{ss}%
  \def\th{th}%
  \def\TH{TH}%
  \def\dh{dh}%
  \def\DH{DH}%
}

\ExplSyntaxOff
```

`\datagidxconvertchars` Convert commands like `\&`. This can mostly be done with the new kernel commands and v3.0 localisation support but maintained for backward compatibility.

```
\newcommand*{\datagidxconvertchars}{%
  \let~\space
  \ifdef\andname
  {%
    \let\&\andname
  }%
  {%
```

```
    \def\&{\expandafter\@gobble\string\&}%
  }%
  \def\_{\string_}%
  \def\${\string$}%
  \def\#{\expandafter\@gobble\string\#}%
  \def\%{\expandafter\@gobble\string\%}%
  \def\{{\expandafter\@gobble\string\{}%
  \def\}{\expandafter\@gobble\string\}}%
}

\ExplSyntaxOn
```

`\datagidxstripaccents` Strip accents so they don't interfere with the label and sort. If you want to write your own comparison handler macro, you'll need to redefine this if you want accented letters to be sorted differently from the unaccented version. This command should only be used within a scope. Need to test kernel version. The new 2019/10/01 version has a different definition of `\UTFviii@two@octets` which won't expand UTF-8 characters to `\IeC` in the required context. (This is really useful if you want extended characters in labels but not if they need to be stripped.) If you don't want accents stripped then redefine `\datagidxstripaccents` to do nothing.

Version 3.0: deprecated. This may not work with modern LaTeX kernels.

```
\newcommand*{\datagidxstripaccents}{%
  \PackageWarning { datagidx }
   {
     \token_to_str:N \datagidxstripaccents \c_space_tl ~
     is ~ deprecated
   }
```

These redefinitions will only work with `\edef` or `\xdef`:

```
    \let\add@accent@\@secondoftwo
    \let\@text@composite@x\@secondoftwo
    \let\@tabacckludge\@secondoftwo
```

The following are need with `\protected@edef` or `\protected@xdef`:

```
    \expandafter\def\csname \encodingdefault-cmd\endcsname##1##2##3{##3}%
    \expandafter\def\csname OT1-cmd\endcsname##1##2##3{##3}%
    \expandafter\def\csname T1-cmd\endcsname##1##2##3{##3}%
    \expandafter\def\csname PD1-cmd\endcsname##1##2##3{##3}%
    \def\IeC##1{\@gobbletwo##1}%
    \let\UTFviii@two@octets\UTFviii@two@octets@combine
}%
```

`\newterm`

```
\newterm[⟨options⟩]name
```

Defaults to normal behaviour.

```
\providecommand{\newterm}{\datagidx@newterm}
```

May only be used in the preamble. (Terms must be defined before the aux file is read.)

```
\@onlypreamble\newterm
```

Set `\l__datagidx_term_label_tl` to the given label:

```
\cs_new:Nn \__datagidx_set_label:n
{
  \__datagidx_set_label:Nn
    \l__datagidx_term_label_tl { #1 }
}
\cs_new:Nn \__datagidx_set_label:Nn
{
  \tl_set:Nx #1 { \text_purify:n { #2 } }
}
```

`\datagidx@setfieldvalues` Syntax: `{`⟨*options*⟩`}{`⟨*name*⟩`}`

Sets the values for all the field.

Version 3.0: replaced `\datagidx@setfieldvalues` with:

```
\cs_new:Nn \__datagidx_set_field_values:nn
{
```

Initialise all token list variables to null to begin with:

```
  \clist_map_inline:Nn \datagidxtermkeys
   {
     \tl_set:cn { l__datagidx_term_ ##1 _ tl} { \c_novalue_tl }
   }
```

Now set non-null defaults.

```
  \tl_set:Nx \l__datagidx_term_database_tl
    { \l__datagidx_default_database_tl }
  \tl_clear:N \l__datagidx_term_parent_tl
```

Predefined field values:

```
  \tl_set:Nn \l__datagidx_term_name_tl { #2 }
  \tl_set:Nn \l__datagidx_term_text_tl { #2 }
  \tl_clear:N \l__datagidx_term_description_tl
  \tl_clear:N \l__datagidx_term_symbol_tl
  \tl_set:Nn \l__datagidx_term_short_tl { #2 }
  \tl_set:Nn \l__datagidx_term_long_tl { #2 }
  \tl_clear:N \l__datagidx_term_see_tl
  \tl_clear:N \l__datagidx_term_seealso_tl
```

Assign values given in optional argument.

```
  \keys_set:nn { datatool / index / newterm } { #1 }
```

Allow hook to access other fields.

```
  \cs_set_eq:NN \__datagidx_org_field:n \field
  \def \field ##1
    { \exp_not:v { l__datagidx_term_ ##1 _tl } }
```

Hook to make it easier to add extra fields.

```
\g__datagidx_newterm_defaults_tl
\let \field \__datagidx_org_field:n
```

Set label if not provided in the options:

```
\tl_if_eq:NnT \l__datagidx_term_label_tl { \c_novalue_tl }
 {
   \__datagidx_create_label:n { #2 }
 }
\exp_args:NV \tl_if_blank:nT \l__datagidx_term_label_tl
 {
    \PackageError { datagidx }
    {
      Blank ~ label ~ not ~ permitted ~
      ( for ~ term ~ ` \tl_to_str:n { #2 } ')
    }
    {
      Use ~ label={...} ~ in ~ options ~
      to ~ set ~ the ~ label ~ to ~ a ~ non-blank ~ value. ~
      Bear ~ in ~ mind ~ that ~ commas ~ (,) ~ and ~
      equals ~ (=) ~ and ~ other ~ problematic ~
      content ~ will ~ be ~ stripped
    }
 }
\tl_if_eq:NnT \l__datagidx_term_sort_tl { \c_novalue_tl }
 {
   \__datagidx_create_sort:n { #2 }
 }
}
```

`\datagidx@add@term` The argument is the term's name. Add term (once all fields have been set. Argument is the name field. The column label should be in `\l__datagidx_term_label_tl` and the database label should be in `\l__datagidx_term_database_tl` Version 3.0: replaced `\datagidx@add@term` with:

```
\cs_new:Nn \__datagidx_add_term:n
{
  \__datagidx_assign_label_db:nn
    { \l__datagidx_term_label_tl }
    { \l__datagidx_term_database_tl }
```

Build the row. Don't use datum for labels, clists.

```
\tl_clear:N \l__datatool_row_tl
\tl_put_right:Nx \l__datatool_row_tl
 {
   \__datatool_row_element_markup:vV
    { dtl@ci@ \l__datagidx_term_database_tl @ Label }
     \l__datagidx_term_label_tl
 }
```

Markup `Used` with `\dtlspecialvalue`

```
\tl_put_right:Nx \l__datatool_row_tl
 {
   \__datatool_row_element_markup:vn
    { dtl@ci@ \l__datagidx_term_database_tl @ Used }
    {
      \dtlspecialvalue { \__datagidx_used:n { 0 } }
    }
 }
\__datagidx_append_element:nn { Name } { #1 }
\__datagidx_append_element:nV
  { Text } \l__datagidx_term_text_tl
\__datagidx_append_element:nV
  { Description } \l__datagidx_term_description_tl
\__datagidx_append_element:nV
  { Sort } \l__datagidx_term_sort_tl
\__datagidx_append_element:nV
  { Symbol } \l__datagidx_term_symbol_tl
\__datagidx_append_element:nV
  { Short } \l__datagidx_term_short_tl
\__datagidx_append_element:nV
  { Long } \l__datagidx_term_long_tl
\tl_if_eq:NnTF \l__datagidx_term_plural_tl { \c_novalue_tl }
  {
    \__datagidx_append_element:no
      { Plural }
      { \l__datagidx_term_text_tl s }
  }
  {
    \__datagidx_append_element:nV
      { Plural } \l__datagidx_term_plural_tl
  }
\tl_if_eq:NnTF \l__datagidx_term_shortplural_tl { \c_novalue_tl }
  {
    \__datagidx_append_element:no
      { ShortPlural }
      { \l__datagidx_term_short_tl s }
  }
  {
    \__datagidx_append_element:nV
      { ShortPlural }
      \l__datagidx_term_shortplural_tl
  }
 \tl_if_eq:NnTF \l__datagidx_term_longplural_tl { \c_novalue_tl }
  {
    \__datagidx_append_element:no
      { LongPlural }
      { \l__datagidx_term_long_tl s }
  }
```

```
    {
      \__datagidx_append_element:nV
        { LongPlural }
        \l__datagidx_term_longplural_tl
    }
  \tl_if_empty:NF \l__datagidx_term_see_tl
    {
      \tl_put_right:Nx \l__datatool_row_tl
        {
          \__datatool_row_element_markup:vV
            { dtl@ci@ \l__datagidx_term_database_tl @ See }
            \l__datagidx_term_see_tl
        }
    }
  \tl_if_empty:NF \l__datagidx_term_seealso_tl
    {
      \tl_put_right:Nx \l__datatool_row_tl
        {
          \__datatool_row_element_markup:vV
            { dtl@ci@ \l__datagidx_term_database_tl @ SeeAlso }
            \l__datagidx_term_see_tl
        }
    }
```

Hook to make it easier to add extra fields.

```
  \g__datagidx_extra_fields_tl
```

Add parent, if supplied.

```
  \tl_if_empty:NF \l__datagidx_term_parent_tl
    {
      \iftermexists { \l__datagidx_term_parent_tl }
        {
          \tl_set:Nx \l__datagidx_parent_database_tl
            {
              \__datagidx_term_database:n
                { \l__datagidx_term_parent_tl }
            }
```

Parent entry is required to belong to same database as child entry.

```
          \tl_if_eq:NNTF
            \l__datagidx_parent_database_tl
            \l__datagidx_term_database_tl
          {
            \tl_put_right:Nx \l__datatool_row_tl
              {
                \__datatool_row_element_markup:vV
                  { dtl@ci@ \l__datagidx_term_database_tl @ Parent }
                  \l__datagidx_term_parent_tl
              }
            \__datagidx_add_child:xxx
              { \l__datagidx_term_database_tl }
```

```
              { \l__datagidx_term_parent_tl }
              { \l__datagidx_term_label_tl }
```

Add the hierarchical sort.

```
            \tl_set:Nn \l__datagidx_term_hiersort_tl { \c_novalue_tl }
            \__datagidx_set_hiersort:
            \tl_if_eq:NnF
              \l__datagidx_term_hiersort_tl { \c_novalue_tl }
             {
               \__datagidx_append_element:nV
                 { HierSort } \l__datagidx_term_hiersort_tl
             }
          }
          {
            \PackageError{datagidx}
            {
              Parent ~ entry ~ ` \l__datagidx_term_parent_tl ' ~ must ~
              belong ~ to ~ the ~ same ~ database ~ as ~
              child ~ entry ~ `\l__datagidx_term_label_tl '
            }
            {
              Parent ~ entry ~ is ~ in ~ database ~
              ` \l__datagidx_parent_database_tl ' ~ and ~
              child ~ entry ~ is ~ in ~
              database ~ `\l__datagidx_term_database_tl'
            }
          }
        }
        {
          \PackageError {datagidx}
          {
             Can't ~ assign ~ parent ~ to ~
             `\l__datagidx_term_label_tl': ~
             ` \l__datagidx_term_parent_tl ' ~ doesn't ~ exist
          }
          {}
        }
     }
```

Add the row to the database. Increment total number of rows:

```
   \int_gincr:c { dtlrows@  \l__datagidx_term_database_tl  }
```

Append row markup to the database's internal token register.

```
   \__datatool_token_register_gput_right:cx
    { dtldb@ \l__datagidx_term_database_tl }
    {
      \__datatool_row_markup:vV
       { dtlrows@ \l__datagidx_term_database_tl }
       \l__datatool_row_tl
    }
```

Provide user with a means to access the label of the latest defined term:

```
   \tl_gset:Nx \datagidxlastlabel
    { \l__datagidx_term_label_tl }
```

Allow user to hook in here:

```
   \postnewtermhook
 }
 \cs_new:Nn \__datagidx_set_hiersort:
 {
   \group_begin:
     \DTLaction
      [
        key = Label,
        expand-value = \l__datagidx_term_parent_tl ,
        name =  \l__datagidx_term_database_tl ,
        return = { \Sort = Sort, \HierSort = HierSort}
      ]
      { select ~ row }
     \datatool_if_null_or_empty:NTF \HierSort
      {
        \datatool_if_null_or_empty:NF \Sort
         {
           \tl_set:Nx \l__datagidx_term_hiersort_tl
             {
                \exp_not:V \Sort
                \exp_not:N \datatoolctrlboundary
                \exp_not:N \datatoolasciistart
                \exp_not:V \l__datagidx_term_sort_tl
             }
         }
      }
      {
        \tl_set:Nx \l__datagidx_term_hiersort_tl
          {
             \exp_not:V \HierSort
             \exp_not:N \datatoolctrlboundary
             \exp_not:N \datatoolasciistart
             \exp_not:V \l__datagidx_term_sort_tl
          }
      }
   \exp_args:NNNV
   \group_end:
     \tl_set:Nn \l__datagidx_term_hiersort_tl
       \l__datagidx_term_hiersort_tl
 }
```

Append an element to the current row token list variable, taking the current expansion setting into account. Syntax: `{`⟨*column key*⟩`}{`⟨*value*⟩`}`

```
 \cs_new:Nn \__datagidx_append_element:nn
 {
   \tl_if_exist:cTF
```

```
      { dtl@ci@ \l__datagidx_term_database_tl @ #1 }
    {
      \__datatool_process_new_value:n { #2 }
      \tl_put_right:Nx \l__datatool_row_tl
       {
         \__datatool_row_element_markup:vV
           { dtl@ci@ \l__datagidx_term_database_tl @ #1 }
           \l__datatool_item_value_tl
       }
    }
    {
      \PackageError { datagidx }
      {
        Column ~ ` #1' ~ hasn't ~ been ~ defined ~ in ~
        database ~ ` \l__datagidx_term_database_tl '
      }
      {
        A ~ new ~ term ~ can't ~ be ~ added ~ to ~
        database ~ ` \l__datagidx_term_database_tl '. ~
        Has ~ the ~ database ~ been ~ defined ~ with ~
        \token_to_str:N \newgidx ? ~ If ~ this ~ is ~
        a ~ custom ~ field, ~ was ~
        \token_to_str:N  \newtermaddfield \c_space_tl ~
        used ~ after ~ ` \l__datagidx_term_database_tl ' ~
        was ~ defined?
      }
    }
}
\cs_generate_variant:Nn
  \__datagidx_append_element:nn
  { nV , no, nv }
```

Syntax: ⟨*value tl var*⟩{⟨*col-key*⟩}

```
\cs_new:Nn \__datagidx_append_element_with_check:Nn
{
  \datatool_if_null:NF #1
   {
     \datatool_if_has_key:nnT
       { \l__datagidx_term_database_tl } { #2 }
      {
        \__datagidx_append_element:nV
          { #2 } #1
      }
   }
}
```

`\postnewtermhook`

```
\newcommand*{\postnewtermhook}{}
```

`\newtermfield` Expandable access to field name. (No check for existence of the field. Expands to an empty string if the field is undefined.)

```
\newcommand*{\newtermfield}[1]{
 \use:c { l__datagidx_term_ #1 _ tl }
}
```

`\ifnewtermfield` If the named field (given in first argument) is empty or undefined do third argument, otherwise do second argument.

```
\newcommand{\ifnewtermfield}[3]{%
  \tl_if_exist:cTF { l__datagidx_term_ #1 _ tl }
   {
     \tl_if_empty:cTF { l__datagidx_term_ #1 _ tl }
       { #3 } { #2 }
   }
   { #3 }
}
```

`\datagidx@newterm` Normal behaviour for `\newterm`

```
\NewDocumentCommand \datagidx@newterm { O{} m }
{
```

Assign values to all the fields.

```
  \__datagidx_set_field_values:nn { #1 } { #2 }
```

Check if database exists.

```
  \DTLifdbexists { \l__datagidx_term_database_tl }
  {
```

Database exists. Check if term already exists.

```
    \iftermexists { \l__datagidx_term_label_tl }
    {
      \PackageError {datagidx}
      {
        Term ~ ` \l__datagidx_term_label_tl ' ~
        already ~ exists ~ in ~ database ~
         ` \l__datagidx_term_database_tl '
      }
      { }
    }
    {
```

Add this entry to the database.

```
      \__datagidx_add_term:n { #2 }
      \dtl@message
       {
         Added ~ term ~  ` \l__datagidx_term_label_tl ' ~
         to ~ database ~ ` \l__datagidx_term_database_tl ' ~
         (name: ~ ` \tl_to_str:N \l__datagidx_term_name_tl ' , ~
          sort: ~ ` \tl_to_str:N \l__datagidx_term_sort_tl ')
       }
    }
```

```
  }
  {
```

Database doesn't exist.

```
    \PackageError {datagidx}
    {
      Glossary/index ~ database ~ `\l__datagidx_term_database_tl' ~
      doesn't ~ exist
    }
    {
      You ~ must ~ define ~ the ~ glossary/index ~
      database ~ with \token_to_str:N \newgidx \c_space_tl ~
      before ~ you ~ can ~ add ~ any ~ terms ~ to ~ it.
    }
  }%
}
```

`datagidx@highopt@newterm` Used when high optimized setting enabled. This setting must be switched off if the user wants to modify the database.

```
\NewDocumentCommand \datagidx@highopt@newterm { O{} m }
{
```

Assign values to all the fields.

```
  \__datagidx_set_field_values:nn { #1 } { #2 }
```

Check if database exists.

```
  \DTLifdbexists { \l__datagidx_term_database_tl }
  {
```

Database exists. If this is the first run, we need to add the term as usual, otherwise we just need to define the mapping from the term label to the database label

```
    \__datatool_get_row_index:Nxxx
      \l__datatool_row_idx_tl
       { \l__datagidx_term_database_tl }
       {
         \dtlcolumnindex
           { \l__datagidx_term_database_tl } {Label}
       }
       { \l__datagidx_term_label_tl }
    \datatool_if_null:NTF \l__datatool_row_idx_tl
     {
```

Hasn't been defined so add.

```
      \__datagidx_add_term:n { #2 }
```

Database will need to be sorted.

```
      \tl_set:cx
        {
          datagidx@do@highopt@sort@
          \l__datagidx_term_database_tl
        }
        { \exp_not:V \l__datagidx_sort_tl }
```

```
      }
      {
```

Has been defined, so just define the mapping and `\datagidxlastlabel`

```
       \__datagidx_assign_label_db:nn
         \l__datagidx_term_label_tl
         \l__datagidx_term_database_tl
       \tl_set:Nx \datagidxlastlabel
         { \l__datagidx_term_label_tl }
      }
  }%
  {%
```

Database doesn't exist.

```
     \PackageError {datagidx}
     {
       Glossary/index ~ database ~
       `\l__datagidx_term_database_tl' ~ doesn't ~ exist
     }
     {
       You ~ must ~ define ~ the ~ glossary/index ~
       database ~ before ~ you ~ can ~
       add ~ any ~ terms ~ to ~ it.
     }
  }
}
```

`\datagidx@addchild` Version 3.0: replaced `\datagidx@addchild` with:

```
\cs_new:Nn \__datagidx_add_child:nnn
{
  \__datatool_get_row_for_value:xxn
   { #1 }
   { \dtlcolumnindex { #1 } { Label } }
   { #2 }
  \dtlgetentryfromcurrentrow
    \l__datagidx_child_clist
    { \dtlcolumnindex { #1 } { Child } }
  \datatool_if_null:NT \l__datagidx_child_clist
   {
     \clist_clear:N \l__datagidx_child_clist
   }
  \clist_put_right:Nx \l__datagidx_child_clist
    { #3 }
  \exp_args:NnV
    \dtlupdateentryincurrentrow
    {Child} \l__datagidx_child_clist
  \dtlrecombine
}
\cs_generate_variant:Nn
  \__datagidx_add_child:nnn
  { xxx }
```

20.3 Defining Acronyms

\newacro

\newacro[⟨*options*⟩]{⟨*short*⟩}{⟨*long*⟩}

Shortcut command for acronyms.

```
\NewDocumentCommand \newacro { O{} m m }
{
  \exp_args:Nno \datagidx_new_acro:nnnn { #1 }
   { \text_uppercase:n { #2 } } { #2 } { #3 }
}

\cs_new:Nn \datagidx_new_acro:nnnn
{
  \bool_if:NTF \l__datatool_new_element_expand_bool
   {
     \newterm
       [
         description = { \exp_not:N \capitalisewords { #4 } },
         short = { \exp_not:N \acronymfont { #3 } },
         long = { #4 },
         text =
          {
            \exp_not:N \DTLgidxAcrStyle
              { #4 } { \exp_not:N \acronymfont { #3 } }
          },
         plural =
          {
             \exp_not:N \DTLgidxAcrStyle
               { #4 s } {\exp_not:N \acronymfont { #3 s } }
          },
         sort = { #3 }, label = { #3 },
         #1
       ]
       { #2 }
   }
   {
     \newterm
       [
         description = { \capitalisewords { #4 } },
         short = { \acronymfont { #3 } },
         long = { #4 },
         text = { \DTLgidxAcrStyle { #4 } { \acronymfont { #3 } } },
         plural = { \DTLgidxAcrStyle { #4 s } { \acronymfont { #3 s } } },
         sort = { #3 }, label = { #3 },
         #1
       ]
       { #2 }
   }
}
```

`\acronymfont` The font to use for the acronym.

```
\newcommand*{\acronymfont}[1]{#1}
```

`\DTLgidxAcrStyle`

```
\DTLgidxAcrStyle{⟨long⟩}{⟨short⟩}
```

```
\newcommand*{\DTLgidxAcrStyle}[2]{#1 ~ (#2)}
```

21 Conditionals

`\iftermexists`

```
\iftermexists{⟨label⟩}{⟨true part⟩}{⟨false part⟩}
```

Check if term with given label exists. This uses the term to database mapping to avoid a database lookup.

```
\newcommand{\iftermexists}[3]{%
  \tl_if_exist:cTF
     { g__datagidxentry_ #1 _ tl }
   { #2 } { #3 }
}
```

Term label to database assignment.

```
\cs_new:Nn \__datagidx_assign_label_db:nn
{
  \tl_if_exist:cTF
    {  g__datagidxentry_ #1 _ tl }
  {
     \exp_args:Nxx \tl_if_eq:nnF
       { #2 } { \tl_use:c { g__datagidxentry_ #1 _ tl } }
     {
       \PackageError { datagidx}
       {
          Can't ~ associated ~ label ~ ` #1 ' ~
          with ~ database ~ `#2'. ~ Label ~
          already ~ associated ~ with ~ database ~
          ` \tl_use:c { g__datagidxentry_ #1 _ tl } '
       }
       { }
     }
  }
  {
    \tl_gset:cx { g__datagidxentry_ #1 _ tl } { #2 }
  }
}
```

Expands to the database label the given term is in.

```
\cs_new:Nn \__datagidx_term_database:n
{
```

```
  \csuse { g__datagidxentry_ #1 _ tl }
}
```

`\datagidxdb` Gets the label of database containing the given entry. No check is made for the existence of the entry. Expands to empty if label is undefined.

```
\newcommand*{\datagidxdb}[1]{
  \__datagidx_term_database:n { #1 }
}
```

`\dtlunknowntag`

```
\newcommand{\dtlunknowntag}{??}
```

Fetch the row from the index/glossary database associated with the given term. The term's label should be in `\l__datagidx_term_label_tl`. This will set `\l__datagidx_term_database_tl` to the database label and `\dtlcurrentrow` if successful.

```
\cs_new:Nn \__datagidx_fetch_row_for_label:NT
{
```

Fetch the name of the database with which this entry is associated.

```
  \tl_set:Nx \l__datagidx_term_database_tl
   {
     \__datagidx_term_database:n
       { \l__datagidx_term_label_tl }
   }
  \tl_if_empty:NTF \l__datagidx_term_database_tl
   {
     \dtlunknowntag
      \PackageError {datagidx}
      {
       \token_to_str:N #1 : ~
        No ~ term ~ ` \l__datagidx_term_label_tl ' ~ defined
      }
      {
       Check ~ that ~ you ~ have ~ correctly ~ spelt ~ the ~
       label. ~ If ~ you ~ defined ~ a ~ term ~ containing ~
       commands ~ and ~ you ~ didn't ~ explicitly ~ set ~
       the ~ label, ~
       \legacy_if:nF { dtlverbose } { switch ~ on ~ verbose ~
       mode ~ and ~ } examine ~ the ~ transcript ~
       messages ~ (Added ~ term ~ ` [...] ' ~ to ~
       database ~ ` [...] ') ~ to ~ check ~ the ~ label
      }
   }
   {
```

Get the row associated with this label and make it the current row.

```
    \tl_set:Nx \l__datatool_item_col_tl
     {
       \dtlcolumnindex
```

```
        { \l__datagidx_term_database_tl } {Label}
    }
  \tl_if_empty:NTF \l__datatool_item_col_tl
    {
      \PackageError { datagidx }
      {
        \token_to_str:N #1 : ~
        No ~ column ~ `Label' ~ found ~ in ~ database ~
        ` \l__datagidx_term_database_tl '
      }
      {
        Something ~ has ~ gone ~ wrong. ~ The ~
        index/glossary ~ database
        ` \l__datagidx_term_database_tl ' ~
        is ~ associated ~ with ~ the ~ term ~
        label ~ `  \l__datagidx_term_label_tl ' ~
        but ~ it ~ doesn't ~ have ~ a ~ column ~ with ~
        the ~ key ~ `Label'
      }
    }
    {
     \exp_args:NNVV
     \__datatool_get_row_for_value:nnnTF
       \l__datagidx_term_database_tl
       \l__datatool_item_col_tl
       \l__datagidx_term_label_tl
     {
```

Found it.

```
       #2
     }
     {
```

Not found. Something has gone wrong.

```
       \PackageError { datagidx }
       {
         \token_to_str:N #1 : ~
         No ~ row ~ found ~ in ~ database ~
         ` \l__datagidx_term_database_tl ' ~ in ~ column ~
         ` Label ' ~ matching ~ ` \l__datagidx_term_label_tl '
       }
       {
         Something ~ has ~ gone ~ wrong. ~ Term ~ with ~
         label ~ ` \l__datagidx_term_label_tl ' ~ was ~
         assigned ~ to ~ database ~
         ` \l__datagidx_term_database_tl ' ~ but ~
         there ~ isn't ~ a ~ corresponding ~ row ~
         in ~ that ~ database ~ for ~ the ~ term
       }
     }
  }
```

```
  }
}
```

\ifentryused `\ifentryused{⟨label⟩}{⟨true part⟩}{⟨false part⟩}`

Check if entry with given label has been used.

```
\NewDocumentCommand \ifentryused { m m m }
{
  \__datagidx_set_label:n { #1 }
  \__datagidx_fetch_row_for_label:NT \ifentryused
   {
     \dtlgetentryfromcurrentrow
       \l__datagidx_value_tl
       { \dtlcolumnindex { \l__datagidx_term_database_tl } {Used} }
     \datatool_if_null_or_empty:NTF
        \l__datagidx_value_tl
      { #1 }
      {
        \int_compare:nNnTF
           { \l__datagidx_value_tl } = { \c_one_int }
          { #2 } { #3 }
      }
   }
}
```

22 Unsetting and Resetting

\glsreset `\glsunset{⟨label⟩}`

Mark as un-used.

```
\NewDocumentCommand \glsreset { m }
{
  \__datagidx_set_label:n { #1 }
  \__datagidx_fetch_row_for_label:NT \glsreset
   {
```

Update the `Used` field.

```
      \dtlreplaceentryincurrentrow
        {
          \dtlspecialvalue { \__datagidx_used:n { 0 } }
        }
        {
          \dtlcolumnindex { \l__datagidx_term_database_tl } {Used}
        }
```

Current row has been edited, so we need to merge the current row back into the database.

```
      \dtlrecombine
    }
}
\MFUexcl { \glsreset }
```

\glsunset

\glsunset{⟨*label*⟩}

Mark as used without affecting location.

```
\NewDocumentCommand \glsunset { m }
{
  \__datagidx_set_label:n { #1 }
  \__datagidx_fetch_row_for_label:NT \glsunset
   {
```

Update the `Used` field.

```
     \dtlreplaceentryincurrentrow
       {
         \dtlspecialvalue { \__datagidx_used:n { 1 } }
       }
       {
         \dtlcolumnindex
           { \l__datagidx_term_database_tl } {Used}
       }
```

Current row has been edited, so we need to merge the current row back into the database.

```
     \dtlrecombine
    }
}
\MFUexcl { \glsunset }
```

\glsresetall

\glsresetall{⟨*db*⟩}

Resets all entries in the given database.

```
\NewDocumentCommand \glsresetall { m }
{
  \dtlforcolumn \l__datagidx_label_tl { #1 } { Label }
  {
    \exp_args:NV \glsreset \l__datagidx_label_tl
  }
}
\MFUexcl { \glsresetall }
```

\glsunsetall

\glsunsetall{⟨*db*⟩}

Resets all entries in the given database.

```
\NewDocumentCommand \glsunsetall { m }
{
  \dtlforcolumn \l__datagidx_label_tl { #1 } { Label }
  {
    \exp_args:NV \glsunset \l__datagidx_label_tl
  }
}
\MFUexcl { \glsunsetall }
```

23 Accessing Entry Information

`\datagidx@anchorcount` Register to make unique anchors. Version 3.0: replaced `\datagidx@anchorcount` with:

```
\int_new:N \g__datagidx_anchor_count_int
```

`\datagidx@formatanchor` Format number using six digits. Version 3.0: replaced `\datagidx@formatanchor` with:

```
\cs_new:Nn \__datagidx_format_anchor:n
{
 \int_compare:nNnT { #1 } < { 10000 }
  {
   0
   \int_compare:nNnT { #1 } < { 1000 }
    {
      0
      \int_compare:nNnT { #1 } < { 100 }
       {
         0
         \int_compare:nNnT { #1 } < { 10 }
          { 0 }
       }
    }
  }
 \int_eval:n { #1 }
}
```

Append FirstId to current row.

```
\cs_new:Nn \__datagidx_append_firstid:n
{
  \exp_args:Nnx \dtlappendentrytocurrentrow
   {FirstId} { \__datagidx_format_anchor:n { #1 } }
}
```

Syntax: ⟨*tl var*⟩`{`⟨*db-name*⟩`}`

Gets the FirstId from the current row:

```
\cs_new:Nn \__datagidx_get_firstid:Nn
{
  \dtlgetentryfromcurrentrow #1
```

```
    {
      \dtlcolumnindex { #2 } {FirstId}
    }
}
```

\@datagidx@escloc

```
\newcommand*{\@datagidx@escloc}[2]
{
  \token_to_str:c { #1 } { \exp_not:N \int_eval:n { #2 } }
}
```

\datagidx@escapelocation Version 3.0: removed \datagidx@escapelocation

idx@escapelocationformat Version 3.0: removed \datagidx@escapelocationformat

gidx@clearlocationformat Version 3.0: removed \datagidx@clearlocationformat

\DTLgidxAddLocationType Allow user to add their own location type. Argument must be control sequence name without initial backslash. Version 3.0: deprecated

```
\newcommand*{\DTLgidxAddLocationType}[1]{%
 \PackageWarning { datagidx }
  {
    \token_to_str:N \DTLgidxAddLocationType \c_space_tl ~
    deprecated. ~ Ignoring
  }
}
```

```
\tl_put_right:Nn \l_datatool_measure_hook_tl
 {
   \cs_set_eq:NN \glsadd \use_none:n
 }
```

\datagidx@target

\datagidx@target{⟨*label*⟩}{⟨*format*⟩}{⟨*location*⟩}{⟨*text*⟩}

Make a target if \hypertarget has been defined, but only write content in the document environment. Ignore the location in the preamble (the aux file isn't open at this point, but increment the anchor count). Version 3.0: replaced \datagidx@target with:

```
\cs_new:Nn \__datagidx_target:nnnn
 {
  \__datagidx_target_create_anchor:nnn
    { #1 } { #2 } {  }
 }
\AtBeginDocument
  {
    \cs_set_eq:NN
      \__datagidx_target:nnnn
      \__datagidx_target_doc:nnnn
  }
```

Create just the anchor information.

```
\cs_new:Nn \__datagidx_target_create_anchor:nnn
 {
  \int_gincr:N \g__datagidx_anchor_count_int
  \tl_set:Nx \l__datagidx_target_tl
   {
     datagidx .
     \__datagidx_format_anchor:n { \g__datagidx_anchor_count_int }
   }
  \tl_if_empty:nTF { #3 }
  {
    \exp_args:Nx \datagidx@write@usedentry
     { #1 } { }
  }
  {
    \__datagidx_write_used_entry:xxnx
      { #1 } { #2 } { #3 } { \l__datagidx_target_tl }
  }
 }
```

Document environment version:

```
\cs_new:Nn \__datagidx_target_doc:nnnn
{
  \__datagidx_target_create_anchor:nnn
    { #1 } { #2 } { #3 }
  \cs_if_exist:NT \hypertarget
  {
```

Make sure the current line doesn't scroll off the top of the screen.

```
    \datagidxshowifdraft
    {
      [ \l__datagidx_target_tl ]
      \discretionary {} {} {}
    }
    \group_begin:
      \datatool_measure_height:Nn
       \l__datatool_tmpa_dim { #4 }
      \raisebox { \l__datatool_tmpa_dim }
      {
        \datagidxtarget { \l__datagidx_target_tl } { }
      }
    \group_end:
  }
  \datagidxshowifdraft
    { [ #1 ] \discretionary {} {} {} }
  #4
}
```

\glsdispentry

\glsdispentry{⟨*label*⟩}{⟨*field*⟩}

Short cut that fetches and displays a value.

```
\NewDocumentCommand \glsdispentry { m m }
{
  \DTLgidxFetchEntry \l__datagidx_value_tl { #1 } { #2 }
  \datatool_if_null:NF \l__datagidx_value_tl
   {
     \l__datagidx_value_tl
   }
}
```

\Glsdispentry

\Glsdispentry{⟨*label*⟩}{⟨*field*⟩}

As previous but makes the first letter upper case.

```
\NewDocumentCommand \Glsdispentry { m m }
{
  \DTLgidxFetchEntry \l__datagidx_value_tl { #1 } { #2 }
  \datatool_if_null:NF \l__datagidx_value_tl
   {
    \xmakefirstuc \l__datagidx_value_tl
   }
}
\MFUaddmap { \glsdispentry } { \Glsdispentry }
```

\DTLgidxFetchEntry

\DTLgidxFetchEntry⟨*tl var*⟩{⟨*label*⟩}{⟨*field*⟩}

Fetch value for the given field for the term identified by ⟨*label*⟩ and store the value in ⟨*cs*⟩ (a control sequence). Triggers an error and sets ⟨*tl var*⟩ to null if not found.

```
\NewDocumentCommand \DTLgidxFetchEntry { m m m }
{
  \__datagidx_set_label:n { #2 }
  \tl_set_eq:NN #1 \dtlnovalue
```

Does this entry exist?

```
  \iftermexists { \l__datagidx_term_label_tl }
  {
    \tl_set:Nx \l__datagidx_term_database_tl
     {
       \__datagidx_term_database:n
        { \l__datagidx_term_label_tl }
     }
    \datatool_if_has_key:nnTF
        { \l__datagidx_term_database_tl }
        { #3 }
```

```
      {
        \__datagidx_fetch_row_for_label:NT \DTLgidxFetchEntry
         {
```

Get the entry for the given field in the current row and store in ⟨*cs*⟩.

```
            \dtlgetentryfromcurrentrow
             #1
            {
              \dtlcolumnindex { \l__datagidx_term_database_tl } { #3 }
            }
         }
      }
      {
        \PackageError { datagidx }
        {
           Database ~ ` \l__datagidx_term_database_tl ' ~
           doesn't ~ have ~ a ~ column ~ labelled ~ `#3'
        }
        {
           Check ~ that ~ you ~ have ~ correctly ~ spelt ~
           the ~ column ~ label ~ (key)
        }
      }
    }
    {
      \PackageError { datagidx }
       {
         No ~ term ~ ` \l__datagidx_term_label_tl ' ~ defined
       }
       {
         Check ~ that ~ you ~ have ~ spelt ~ the ~ label ~
         correctly ~ and ~ that ~ the ~ term ~ was ~ added ~
         to ~ the ~ database ~ with ~ \token_to_str:N \newterm
         \c_space_tl ~ or \token_to_str:N \newacro
       }
    }
  }
```

`\datagidx@format` Version 3.0: replaced `\datagidx@format` with:

```
\tl_new:N \l__datagidx_format_tl
```

`tagidx@parse@formatlabel`

```
\parse@formatlabel{⟨[⟨format⟩]label⟩}
```

Separate format and label from argument.

```
\newcommand*{\datagidx@parse@formatlabel}[1]{%
  \datagidx@parse@format@label@#1\@endparse@formatlabel@
}
\newcommand*\datagidx@parse@format@label@{%
```

```
  \@ifnextchar[{\datagidx@parse@formatlabel@}{\datagidx@parse@formatlabel@[]}%
}
\def\datagidx@parse@formatlabel@[#1]#2\@endparse@formatlabel@{%
  \tl_set:Nn \l__datagidx_format_tl { #1 }
  \__datagidx_set_label:n { #2 }
}
```

\@datagidx@use@entry

\@datagidx@use@entry{⟨*link text*⟩}

\@datagidx@use@entry

The label and format should have been stored in \l__datagidx_term_label_tl and \l__datagidx_format_tl before calling this macro. Version 3.0: replaced \@datagidx@use@entry with:

Syntax: ⟨*calling-cs*⟩ {⟨*link text*⟩}

```
\cs_new:Nn \__datagidx_use_entry:Nn
{
  \__datagidx_fetch_row_for_label:NT #1
   {
```

Get the entry for the FirstId field and store in \l__datagidx_id_tl

```
      \__datagidx_get_firstid:Nn
        \l__datagidx_id_tl
        { \l__datagidx_term_database_tl }
```

If it hasn't been defined set it.

```
      \datatool_if_null:NT \l__datagidx_id_tl
       {
```

Count register hasn't been updated yet so add 1.

```
        \exp_args:Nx \__datagidx_append_firstid:n
         {
           \int_eval:n
             {  \g__datagidx_anchor_count_int + \c_one_int }
         }
       }
```

Update the Used field.

```
      \dtlreplaceentryincurrentrow
       {
         \dtlspecialvalue { \__datagidx_used:n { 1 } }
       }
       {
         \dtlcolumnindex
          { \l__datagidx_term_database_tl } {Used}
       }
```

Get the parent entry label (if one exists).

```
      \dtlgetentryfromcurrentrow
        \l__datagidx_parent_tl
        {
```

```
          \dtlcolumnindex
            { \l__datagidx_term_database_tl }{Parent}
        }
```

Current row has been edited, so we need to merge the current row back into the database.

```
      \dtlrecombine
```

If parent hasn't be used, give it an empty location.

```
      \__datagidx_mark_parent:NN
       \l__datagidx_term_database_tl \l__datagidx_parent_tl
```

Write the location to the auxiliary file and display value of field.

```
       \__datagidx_target:nnnn
        { \l__datagidx_term_label_tl }
        { \l__datagidx_format_tl }
        { \csuse { the\DTLgidxCounter } }
        { #2 }
    }
 }
```

`\datagidx@markparent` Syntax: ⟨*db-tl-var*⟩ ⟨*parent-tl-var*⟩ Assign empty location to parent, if location of that parent is null. (Recursive). Version 3.0: replaced `\datagidx@markparent` with:

```
 \cs_new:Nn \__datagidx_mark_parent:NN
 {
   \datatool_if_null:NF #2
    {
```

Write empty location to the auxiliary file.

```
     \__datagidx_target:nnnn { #2 } { } { } { }
```

Fetch this parent's parent entry. Get the row associated with this and make it the current row.

```
     \__datatool_get_row_for_value:xxx
       { #1 }
       { \dtlcolumnindex { #1 } { Label } }
       { #2 }
```

Get the entry for the `FirstId` field and store in `\l__datagidx_id_tl`

```
       \__datagidx_get_firstid:Nn
         \l__datagidx_id_tl
         { \l__datagidx_term_database_tl }
```

If it hasn't been defined set it.

```
     \datatool_if_null:NT \l__datagidx_id_tl
     {
       \__datagidx_append_firstid:n
        {
          \g__datagidx_anchor_count_int
        }
     }
```

Get the parent

```
      \dtlgetentryfromcurrentrow
        \l__datagidx_parent_tl
        {\dtlcolumnindex{#1}{Parent}}%
```

Current row has been edited, so we need to merge the current row back into the database.

```
    \dtlrecombine
```

Recurse

```
    \__datagidx_mark_parent:NN #1 \l__datagidx_parent_tl
  }
}
```

Syntax: `{`⟨*label*⟩`}{`⟨*format*⟩`}{`⟨*location*⟩`}{`⟨*target*⟩`}`

```
\cs_new:Nn \__datagidx_write_used_entry:nnnn
{
  \datagidx@write@usedentry
    { #1 }
    { [ #2 ] { #3 } { #4 } }
}
\cs_generate_variant:Nn
  \__datagidx_write_used_entry:nnnn
   { xxnx }
```

`datagidx@write@usedentry` Write out location to aux file and add location to the location list for the current run.

```
\newcommand*{\datagidx@write@usedentry}[2]{%
```

Do update if 'highopt optimize' setting is on.

```
  \datagidx@do@highopt@update{#1}%
```

```
  \protected@edef \l__datagidx_location_tl { #2 }
```

Write out location to aux file.

```
  \protected@write{\@auxout}{}%
    {%
      \token_to_str:N \datagidx@aux@usedentry
       { #1 } { #2 }
       { \tl_to_str:N \l__datagidx_location_tl }
    }%
```

This may be a page off if indexing occurs by a page break, so it should be checked with UnsafeLocation rather than Location.

```
    \__datagidx_used_entry:nxx
      { CurrentLocation } { #1 }
      { \tl_to_str:N \l__datagidx_location_tl }
}
```

`\datagidx@usedentry{`⟨*location tag*⟩`}{`⟨*label*⟩`}{`⟨*location*⟩`}`

`\datagidx@xusedentry`

Like `\datagidx@usedentry` but expands the location. Unlike `\datagidx@usedentry` the first argument isn't optional. Version 3.0: replaced `\datagidx@xusedentry` with:

```
\cs_new:Nn \__datagidx_used_entry:nnn
{
  \datagidx@usedentry [ #1 ] { #2 } { #3 }
}
\cs_generate_variant:Nn
 \__datagidx_used_entry:nnn
 { xxx, nxx }
```

`\datagidx@usedentry`

`\datagidx@usedentry[⟨location tag⟩]{⟨label⟩}{⟨location⟩}`

Add to the location list for the given entry.

```
\newcommand*{\datagidx@usedentry}[3][Location]{%
```

Check if label exists. (It may have been deleted or had a label change.)

```
  \iftermexists { #2 }
  {%
```

Fetch the name of the database with which this entry is associated.

```
    \tl_set:Nx \l__datagidx_term_database_tl
      {  \__datagidx_term_database:n { #2 } }
```

Get the row associated with this label and make it the current row.

```
    \__datatool_get_row_for_value:xxx
      { \l__datagidx_term_database_tl }
      { \dtlcolumnindex{\l__datagidx_term_database_tl}{Label} }
      { #2 }
```

Get the entry for the ⟨*location tag*⟩ field in the current row and store in `\l__datagidx_location_tl`.

```
    \dtlgetentryfromcurrentrow
      \l__datagidx_location_tl
      { \dtlcolumnindex { \l__datagidx_term_database_tl } { #1 } }
    \__datagidx_unwrap_location:N
      \l__datagidx_location_tl
```

Check the success of the previous command.

```
    \datatool_if_null:NTF \l__datagidx_location_tl
     {
```

There's no ⟨*location tag*⟩ field in the current row, so add one with the given location.

```
      \tl_set:Nn \l__datagidx_location_tl { #3 }
      \__datagidx_unwrap_location:N
        \l__datagidx_location_tl
      \exp_args:Nne \dtlappendentrytocurrentrow
        { #1 }
        {
          \exp_not:N \dtlspecialvalue
```

```
            {
              \exp_not:N \__datagidx_location:n
              {
                \exp_not:V \l__datagidx_location_tl
              }
            }
          }
        }
        {
```

There is a ⟨*location tag*⟩ field in the current row, so append the given location to the list, unless one or the other is empty.

```
          \clist_clear:N \l__datagidx_location_clist
          \tl_if_empty:NTF \l__datagidx_location_tl
          {
            \clist_put_right:Nn \l__datagidx_location_clist {#3}
          }
          {
            \clist_set:NV \l__datagidx_location_clist
              \l__datagidx_location_tl
            \tl_if_empty:nF { #3 }
            {
              \clist_put_right:Nn \l__datagidx_location_clist { #3 }
            }
          }
```

and update the entry in the current row.

```
          \exp_args:Nx \dtlreplaceentryincurrentrow
            {
              \exp_not:N \dtlspecialvalue
              {
                \exp_not:N \__datagidx_location:n
                {
                  \clist_use:Nn \l__datagidx_location_clist { , }
                }
              }
            }
            { \dtlcolumnindex { \l__datagidx_term_database_tl } {#1} }
        }
```

Current row has been edited, so we need to merge the current row back into the database.

```
        \dtlrecombine
    }
    {
      \PackageWarning{datagidx}{No ~ term ~ `#2' ~ defined. ~ Ignoring}
    }
  }
```

\datagidx@aux@usedentry

\datagidx@aux@usedentry{⟨*label*⟩}{⟨*location*⟩}{⟨*unsafe location*⟩}

This command is for the aux file. The unsafe location is a location that may be off due to the delayed shipout. The value is used to compare with the current location to determine if a rerun is required.

```
\newcommand*{\datagidx@aux@usedentry}[3]
{
```

Check if label exists. (It may have been deleted or had a label change.)

```
  \iftermexists { #1 }
  {
```

Fetch the name of the database with which this entry is associated.

```
    \tl_set:Nx \l__datagidx_term_database_tl
      {  \__datagidx_term_database:n { #1 } }
```

Get the row associated with this label and make it the current row.

```
    \__datatool_get_row_for_value:xxxT
      { \l__datagidx_term_database_tl }
      { \dtlcolumnindex{\l__datagidx_term_database_tl}{Label} }
      { #1 }
    {
```

Get the entry for the Location field in the current row and store in `\l__datagidx_location_tl`.

```
      \dtlgetentryfromcurrentrow
        \l__datagidx_location_tl
        { \dtlcolumnindex { \l__datagidx_term_database_tl } { Location } }
      \__datagidx_unwrap_location:N
        \l__datagidx_location_tl
```

Check the success of the previous command.

```
      \datatool_if_null:NTF \l__datagidx_location_tl
       {
```

There's no Location field in the current row, so add one with the given location.

```
        \tl_set:Nn \l__datagidx_location_tl { #2 }
        \__datagidx_unwrap_location:N
          \l__datagidx_location_tl
        \exp_args:Nne \dtlappendentrytocurrentrow
          { Location }
          {
            \exp_not:N \dtlspecialvalue
            {
              \exp_not:N \__datagidx_location:n
              {
                \exp_not:V \l__datagidx_location_tl
              }
            }
          }
      }
```

```
{
```
There is a Location field in the current row, so append the given location to the list, unless one or the other is empty.
```
\clist_clear:N \l__datagidx_location_clist
\tl_if_empty:NTF \l__datagidx_location_tl
{
  \clist_put_right:Nn \l__datagidx_location_clist
    { #2 }
}
{
  \clist_set:NV \l__datagidx_location_clist
    \l__datagidx_location_tl
  \tl_if_empty:nF { #2 }
  {
    \clist_put_right:Nn \l__datagidx_location_clist
      { #2 }
  }
}
```
and update the entry in the current row.
```
\exp_args:Nx \dtlreplaceentryincurrentrow
  {
    \exp_not:N \dtlspecialvalue
    {
      \exp_not:N \__datagidx_location:n
      {
        \clist_use:Nn \l__datagidx_location_clist { , }
      }
    }
  }
  {
    \dtlcolumnindex
      { \l__datagidx_term_database_tl }
      { Location }
  }
}
```
Get the entry for the UnsafeLocation field in the current row and store in `\l__datagidx_location_tl`.
```
\dtlgetentryfromcurrentrow
  \l__datagidx_location_tl
  {
    \dtlcolumnindex
    { \l__datagidx_term_database_tl }
    { UnsafeLocation }
  }
\__datagidx_unwrap_location:N
  \l__datagidx_location_tl
```
Check the success of the previous command.
```
\datatool_if_null:NTF \l__datagidx_location_tl
 {
```

There's no UnsafeLocation field in the current row, so add one with the given location.

```
\tl_set:Nn \l__datagidx_location_tl { #3 }
\__datagidx_unwrap_location:N
  \l__datagidx_location_tl
\exp_args:Nne \dtlappendentrytocurrentrow
  { UnsafeLocation }
  {
    \exp_not:N \dtlspecialvalue
    {
      \exp_not:N \__datagidx_location:n
      {
        \tl_to_str:N \l__datagidx_location_tl
      }
    }
  }
}
{
```

There is an UnsafeLocation field in the current row, so append the given location to the list, unless one or the other is empty.

```
\clist_clear:N \l__datagidx_location_clist
\tl_if_empty:NTF \l__datagidx_location_tl
{
  \tl_set:Nn \l__datagidx_location_tl { #3 }
  \__datagidx_unwrap_location:N
    \l__datagidx_location_tl
  \clist_put_right:Nx \l__datagidx_location_clist
    { \tl_to_str:N \l__datagidx_location_tl }
}
{
  \clist_set:NV \l__datagidx_location_clist
    \l__datagidx_location_tl
  \tl_if_empty:nF { #3 }
  {
    \tl_set:Nn \l__datagidx_location_tl { #3 }
    \__datagidx_unwrap_location:N
      \l__datagidx_location_tl
    \clist_put_right:Nx \l__datagidx_location_clist
      { \tl_to_str:N \l__datagidx_location_tl }
  }
}
```

and update the entry in the current row.

```
\exp_args:Nx \dtlreplaceentryincurrentrow
  {
    \exp_not:N \dtlspecialvalue
    {
      \exp_not:N \__datagidx_location:n
      {
        \clist_use:Nn \l__datagidx_location_clist { , }
```

```
          }
        }
      }
      {
        \dtlcolumnindex
          { \l__datagidx_term_database_tl }
          { UnsafeLocation }
      }
  }
```

Current row has been edited, so we need to merge the current row back into the database.

```
    \dtlrecombine
  }
}
{
  \PackageWarning{datagidx}{No ~ term ~ `#1' ~ defined. ~ Ignoring}
}
}
```

`\datagidx@save@loc` Store the current location from the previous run. Version 3.0: no longer used but still defined until next version as it may be in the aux file of existing documents. TODO: remove

```
\newcommand*{\datagidx@save@loc}[2]{%
}
```

`\glsadd`

```
\glsadd{⟨[⟨format⟩]label⟩}
```

```
\NewDocumentCommand \glsadd { m }
{
  \group_begin:
```

Check term has been defined.

```
  \iftermexists { \l__datagidx_term_label_tl }
  {
    \datagidx@parse@formatlabel{#1}%
```

Write the location to the auxiliary file.

```
    \__datagidx_target:nnnn
      { \l__datagidx_term_label_tl }
      { \l__datagidx_format_tl }
      { \csuse{the\DTLgidxCounter} }
      { }
```

Fetch the name of the database with which this entry is associated.

```
    \tl_set:Nx \l__datagidx_term_database_tl
     { \__datagidx_term_database:n { \l__datagidx_term_label_tl } }
```

Get the row associated with this label and make it the current row.

```
    \__datatool_get_row_for_value:xxxTF
```

```
       { \l__datagidx_term_database_tl }
       { \dtlcolumnindex { \l__datagidx_term_database_tl } {Label} }
       { \l__datagidx_term_label_tl }
      {
```

Update the `Used` field.

```
        \dtlreplaceentryincurrentrow
         {
           \dtlspecialvalue { \__datagidx_used:n { 1 } }
         }
         {
           \dtlcolumnindex { \l__datagidx_term_database_tl } {Used}
         }
```

Get the entry for the `FirstId` field and store in `\l__datagidx_id_tl`

```
        \__datagidx_get_firstid:Nn
          \l__datagidx_id_tl
          { \l__datagidx_term_database_tl }
```

If it hasn't been defined set it.

```
        \datatool_if_null:NT \l__datagidx_id_tl
         {
           \__datagidx_append_firstid:n
            { \g__datagidx_anchor_count_int }
         }
```

Current row has been edited, so we need to merge the current row back into the database.

```
        \dtlrecombine
      }
      {
```

The label to database mapping has been found but the label can't be found in the database.

```
        \PackageError { datagidx }
         {
           No ~ row ~ found ~ for ~ label ~
           ` \l__datagidx_term_label_tl ' ~ in ~
           database ~ ` \l__datagidx_term_database_tl '
         }
         {
           Something ~ has ~ gone ~ wrong ~ with ~ the ~ underlying ~
           structure. ~ Did ~ you ~ define ~ ` \l__datagidx_term_label_tl ' ~
           using ~ \token_to_str:N \newterm \c_space_tl or
           \token_to_str:N \newacro ? ~ Has ~ the ~ database ~
           ` \l__datagidx_term_database_tl ' ~ been ~ modified ~
           outside ~ of ~ datagidx.sty ~ commands?
         }
    }
  }
  {
    \PackageError {datagidx}
     { Term ~ `\l__datagidx_term_label_tl' ~ doesn't ~ exist }
```

```
      {
        Check ~ the ~ spelling ~ of ~ the ~ label. ~ Verbose ~ mode ~
        will ~ show ~ the ~ labels ~ in ~ the ~ transcript ~ for ~
        each ~ new ~ term
      }
    }
   \group_end:
  }
  \MFUexcl { \glsadd }
```

`\datagidx@count` Version 3.0: Removed `\datagidx@count`

`\glsaddall`

`\glsaddall{`⟨*db*⟩`}`

Adds all entries in the given database.

```
\NewDocumentCommand \glsaddall { m }
{
  \DTLifdbexists{#1}%
  {
    \int_step_inline:nn
       { \DTLrowcount { #1 } }
    {
      \datatool_get_row:nnT { #1 } { ##1 }
      {
```

Get the label for this row.

```
        \dtlgetentryfromcurrentrow
           \l__datagidx_term_label_tl
          { \dtlcolumnindex { #1 } {Label} }
```

Write blank location to the auxiliary file but temporarily change `\hypertarget` as it doesn't make sense to have a target here.

```
        \group_begin:
          \cs_set_eq:NN \hypertarget \use_ii:nn
          \__datagidx_target:nnnn
            { \l__datagidx_term_label_tl }{}{}{}
        \group_end:
```

Update the `Used` field.

```
        \dtlreplaceentryincurrentrow
         {
           \dtlspecialvalue { \__datagidx_used:n { 1 } }
         }
         { \dtlcolumnindex { #1 } {Used} }
```

Get the entry for the `FirstId` field and store in `\l__datagidx_id_tl`

```
        \__datagidx_get_firstid:Nn
          \l__datagidx_id_tl { #1 }
```

If it hasn't been defined set it.

```
          \datatool_if_null:NT \l__datagidx_id_tl
          {
            \__datagidx_append_firstid:n
             { \g__datagidx_anchor_count_int }
          }
```

Current row has been edited, so we need to merge the current row back into the database.

```
          \dtlrecombine
        }
      }
    }
    {
      \PackageError {datagidx}
      {
        \token_to_str:N \glsaddall : ~
         Database ~ `#1' ~ doesn't ~ exist
      }
      {
        Check ~ that ~ you ~ have ~ correctly ~ spelt ~ the ~ database
        ~ name ~ in ~ the ~ argument ~ of \token_to_str:N \glsaddall
      }
    }
  }
  \MFUexcl { \glsaddall }
```

Version 3.0 has added grouping to the following to prevent leakage of scratch token list variables.

\glslink

```
\glslink{⟨label⟩}{⟨text⟩}
```

Use given entry but user supplies text.

```
\NewDocumentCommand \glslink { m m }
{
 \group_begin:
  \datagidx@parse@formatlabel {#1}
  \datagidxlink { \l__datagidx_term_label_tl }
  {
    \__datagidx_use_entry:Nn \glslink { #2 }
  }
 \group_end:
}
\NewDocumentCommand \Glslink { m m }
{
  \glslink { #1 } { \makefirstuc { #2 } }
}
\MFUaddmap { \glslink } { \Glslink }
```

\useentry

\useentry{⟨*label*⟩}{⟨*field*⟩}

Fetch and use the given field for the given entry.

```
\NewDocumentCommand \useentry { m m }
{
 \group_begin:
  \datagidx@parse@formatlabel { #1 }
  \DTLgidxFetchEntry
    \l__datagidx_value_tl
    { \l__datagidx_term_label_tl } { #2 }
```

Just test for null, `\DTLgidxFetchEntry` will have already triggered an error.

```
  \datatool_if_null:NF \l__datagidx_value_tl
  {
    \datagidxlink { \l__datagidx_term_label_tl }
    {
      \__datagidx_use_entry:Nn \useentry
        { \l__datagidx_value_tl }
    }
  }
 \group_end:
}
```

\Useentry

\Useentry{⟨*label*⟩}{⟨*field*⟩}

As `\useentry`, but capitalise the first word.

```
\NewDocumentCommand \Useentry { m m }
{
 \group_begin:
  \datagidx@parse@formatlabel { #1 }
  \DTLgidxFetchEntry
    \l__datagidx_value_tl
     { \l__datagidx_term_label_tl } { #2 }
  \datatool_if_null:NF \l__datagidx_value_tl
  {
    \datagidxlink { \l__datagidx_term_label_tl }
    {
      \__datagidx_use_entry:Nn \Useentry
        { \xmakefirstuc {  \l__datagidx_value_tl } }
    }
  }
 \group_end:
}
\MFUaddmap { \useentry } { \Useentry }
```

\USEentry

\USEentry{⟨*label*⟩}{⟨*field*⟩}

As `\useentry`, but make the whole term upper case.

```
\NewDocumentCommand \USEentry { m m }
{
 \group_begin:
  \datagidx@parse@formatlabel { #1 }
  \DTLgidxFetchEntry
    \l__datagidx_value_tl
     { \l__datagidx_term_label_tl } { #2 }
  \datatool_if_null:NF \l__datagidx_value_tl
  {
    \datagidxlink { \l__datagidx_term_label_tl }
    {
      \__datagidx_use_entry:Nn \USEentry
       { \text_uppercase:n { \l__datagidx_value_tl } }
    }
  }
 \group_end:
}
\MFUexcl { \USEentry }
```

`\useentrynl` `\useentrynl{⟨label⟩}{⟨field⟩}`

Fetch and use the given field for the given entry without creating a hyperlink.

```
\NewDocumentCommand \useentrynl { m m }
{
 \group_begin:
  \datagidx@parse@formatlabel { #1 }
  \DTLgidxFetchEntry
    \l__datagidx_value_tl
      { \l__datagidx_term_label_tl } { #2 }
  \datatool_if_null:NF \l__datagidx_value_tl
  {
    \__datagidx_use_entry:Nn \useentrynl
      { \l__datagidx_value_tl }
  }
 \group_end:
}
```

`\Useentrynl` `\Useentrynl{⟨label⟩}{⟨field⟩}`

As `\useentry`, but capitalise the first word.

```
\NewDocumentCommand \Useentrynl { m m }
{
 \group_begin:
  \datagidx@parse@formatlabel { #1 }
  \DTLgidxFetchEntry
```

```
    \l__datagidx_value_tl
     { \l__datagidx_term_label_tl } { #2 }
   \datatool_if_null:NF \l__datagidx_value_tl
   {
     \__datagidx_use_entry:Nn \Useentrynl
      { \xmakefirstuc {  \l__datagidx_value_tl } }
   }
  \group_end:
 }
 \MFUaddmap { \useentrynl } { \Useentrynl }
```

\USEentrynl

\USEentrynl{⟨*label*⟩}{⟨*field*⟩}

As \useentry, but make the whole term upper case.

```
\NewDocumentCommand \USEentrynl { m m }
{
 \group_begin:
  \datagidx@parse@formatlabel { #1 }
  \DTLgidxFetchEntry
    \l__datagidx_value_tl
     { \l__datagidx_term_label_tl } { #2 }
  \datatool_if_null:NF \l__datagidx_value_tl
  {
    \__datagidx_use_entry:Nn \USEentrynl
     { \text_uppercase:n {  \l__datagidx_value_tl } }
  }
 \group_end:
}
\MFUexcl { \USEentrynl }
```

Short cuts to common fields.

\gls

```
\newrobustcmd*{\gls}[1]{\useentry{#1}{Text}}
```

\glspl

```
\newrobustcmd*{\glspl}[1]{\useentry{#1}{Plural}}
```

\Gls

```
\newrobustcmd*{\Gls}[1]{\Useentry{#1}{Text}}
\MFUaddmap { \gls } { \Gls }
```

\Glspl

```
\newrobustcmd*{\Glspl}[1]{\Useentry{#1}{Plural}}
\MFUaddmap { \glspl } { \Glspl }
```

\glsnl

```
\newrobustcmd*{\glsnl}[1]{\useentrynl{#1}{Text}}
```

```
\glsplnl
          \newrobustcmd*{\glsplnl}[1]{\useentrynl{#1}{Plural}}

  \Glsnl
          \newrobustcmd*{\Glsnl}[1]{\Useentrynl{#1}{Text}}
          \MFUaddmap { \glsnl } { \Glsnl }

\Glsplnl
          \newrobustcmd*{\Glsplnl}[1]{\Useentrynl{#1}{Plural}}
          \MFUaddmap { \glsplnl } { \Glsplnl }

 \glssym
          \newrobustcmd*{\glssym}[1]{\useentry{#1}{Symbol}}

 \Glssym
          \newrobustcmd*{\Glssym}[1]{\Useentry{#1}{Symbol}}
          \MFUaddmap { \glssym } { \Glssym }
```

23.1 Using Acronyms

\DTLgidxFormatAcr

```
\DTLgidxFormatAcr{⟨label⟩}{⟨long field⟩}{⟨short field⟩}
```

```
\newcommand*{\DTLgidxFormatAcr}[3]{%
  \DTLgidxAcrStyle{\glsdispentry{#1}{#2}}{\useentry{#1}{#3}}%
}
```

\DTLgidxFormatAcrUC

```
\DTLgidxFormatAcr{⟨label⟩}{⟨long field⟩}{⟨short field⟩}
```

As previous but capitalise first word.

```
\newcommand*{\DTLgidxFormatAcrUC}[3]{%
  \DTLgidxAcrStyle{\Glsdispentry{#1}{#2}}{\useentry{#1}{#3}}%
}
```

```
   \acr
          \newrobustcmd*{\acr}[1]{%
            \ifentryused{#1}%
            {\useentry{#1}{Short}}%
            {\DTLgidxFormatAcr{#1}{Long}{Short}}%
          }

  \acrpl
          \newrobustcmd*{\acrpl}[1]{%
            \ifentryused{#1}%
```

```
    {\useentry{#1}{ShortPlural}}%
    {\DTLgidxFormatAcr{#1}{LongPlural}{ShortPlural}}%
  }

\Acr
  \newrobustcmd*{\Acr}[1]{%
    \ifentryused{#1}%
    {\Useentry{#1}{Short}}%
    {\DTLgidxFormatAcrUC{#1}{Long}{Short}}%
  }
  \MFUaddmap { \acr } { \Acr }

\Acrpl
  \newrobustcmd*{\Acrpl}[1]{%
    \ifentryused{#1}%
    {\Useentry{#1}{ShortPlural}}%
    {\DTLgidxFormatAcrUC{#1}{LongPlural}{ShortPlural}}%
  }
  \MFUaddmap { \acrpl } { \Acrpl }
```

24 Displaying Glossaries, Lists of Acronyms, Indices

`\printtermsstartpar`

```
\newcommand{\printtermsstartpar}{\par}
```

`\datagidx@prestart` Version 3.0: replaced `\datagidx@prestart` with:

```
\tl_new:N \l__datagidx_prestart_tl
```

`printterms@setupmulticol`

```
\newcommand*{\printterms@setupmulticol}{
 \tl_if_empty:NTF \l__datagidx_post_heading_tl
 {
   \tl_set:Nx \l__datagidx_prestart_tl
    {
      \exp_not:N \l__datagidx_heading_tl
        { \exp_not:N \l__datagidx_title_tl }
      \exp_not:N \begin
        { \l__datagidx_multicols_tl }
        { \int_use:N \l__datagidx_columns_int }
    }
 }
 {
   \tl_set:Nx \l__datagidx_prestart_tl
    {
      \exp_not:N \l__datagidx_heading_tl
        {\exp_not:N \l__datagidx_title_tl }
      \exp_not:N \begin { \l__datagidx_multicols_tl }
        { \int_use:N \l__datagidx_columns_int }
```

```
      [ \exp_not:N \l__datagidx_post_heading_tl ]
    }
  }
  \tl_set:Nx \datagidx@postend
   {
     \exp_not:N \end { \l__datagidx_multicols_tl }
   }
}

\printterms@setuptwocol
\newcommand*{\printterms@setuptwocol}{
 \tl_set:Nn \l__datagidx_prestart_tl
  {
    \twocolumn
     [
       \l__datagidx_heading_tl
         { \l__datagidx_title_tl }
       \l__datagidx_post_heading_tl
     ]
  }
 \if@twocolumn
   \tl_clear:N \datagidx@postend
 \else
   \tl_set:Nn \datagidx@postend
    { \printtermsrestoreonecolumn }
 \fi
}

\inttermsrestoreonecolumn
\newcommand{\printtermsrestoreonecolumn}{\onecolumn}
```

\printterms

```
\printterms[options]
```

Print the list of terms

```
\NewDocumentCommand \printterms { o }
{
\group_begin:
```

Ensure that sorting has a global effect:

```
  \bool_set_true:N \l__datatool_db_global_bool
```

Initialise key list for style:

```
  \clist_clear:N \l__datagidx_styles_clist
```

Set options. If the database key is used, this will set `\l__datagidx_default_database_tl`

```
  \IfValueT { #1 }
   {
     \keys_set_filter:nnnN
       { datatool / index } { general } { #1 }
```

```
      \l__datagidx_remainder_tl
    \tl_if_empty:NF \l__datagidx_remainder_tl
     {
       \PackageError { datagidx }
        {
          Invalid ~ \token_to_str:N \printterms \c_space_tl option(s): ~
           \tl_to_str:N \l__datagidx_remainder_tl
        }
        {
          The ~ listed ~ option ~ or ~ options ~ can't ~ be ~ passed ~
          to ~ \token_to_str:N \printterms . \MessageBreak
          Try ~ \token_to_str:N \DTLsetup
           { index = { \tl_to_str:N \l__datagidx_remainder_tl } }
        }
     }
  }
```

Set the database.

```
\tl_set_eq:NN
 \l__datagidx_term_database_tl
 \l__datagidx_default_database_tl
```

Check if database exists.

```
\DTLifdbexists { \l__datagidx_term_database_tl }
{
```

Provide user the means to access the current database name.

```
  \tl_set_eq:NN
    \DTLgidxCurrentdb
    \l__datagidx_term_database_tl
```

Get the fields from datagidx:

```
  \__datatool_get_row_for_value:xxxTF
    { datagidx }
    { \dtlcolumnindex{datagidx}{Glossary} }
    { \l__datagidx_term_database_tl }
  {
   \dtlgetentryfromcurrentrow
     \l__datagidx_title_tl
     {\dtlcolumnindex{datagidx}{Title}}%
   \dtlgetentryfromcurrentrow
     \l__datagidx_heading_tl
     {\dtlcolumnindex{datagidx}{Heading}}%
   \dtlgetentryfromcurrentrow
     \l__datagidx_post_heading_tl
     { \dtlcolumnindex {datagidx} {PostHeading} }
   \dtlgetentryfromcurrentrow
     \l__datagidx_multicols_tl
     { \dtlcolumnindex {datagidx} {MultiCols} }
   \dtlgetentryfromcurrentrow
     \l__datagidx_sort_tl
```

```
  { \dtlcolumnindex {datagidx} {Sort} }
\dtlgetentryfromcurrentrow
  {\datagidx@style}%
  {\dtlcolumnindex{datagidx}{Style}}%
\dtlgetentryfromcurrentrow
  {\datagidx@showgroups}%
  {\dtlcolumnindex{datagidx}{ShowGroups}}%
```

Allow user to override style here.

```
\keys_set_groups:nnV { datatool / index } { print }
   \l__datagidx_styles_clist
```

Do we need to use multicols?

```
\int_case:nnF { \l__datagidx_columns_int }
 {
   { \c_one_int }
     {
       \tl_clear:N \l__datagidx_post_heading_tl
       \tl_clear:N \datagidx@postend
     }
   { 2 }
     {
       \ifdatagidxbalance
         \printterms@setupmulticol
       \else
         \printterms@setuptwocol
       \fi
     }
 }
 {
   \printterms@setupmulticol
 }
\tl_set_eq:NN \@dtl@dbname \DTLgidxCurrentdb
```

Set the style

```
\csuse{datagidxshowgroups\datagidx@showgroups}%
\datagidxsetstyle{\datagidx@style}%
```

Now display the glossary/index:

```
\int_compare:nNnT
  { \l__datagidx_columns_int } = { \c_one_int }
 {
   \l__datagidx_heading_tl
     { \l__datagidx_title_tl }
   \l__datagidx_post_heading_tl
 }
\datagidx@do@sort
\l__datagidx_prestart_tl
```

:

```
\printtermsstartpar
\datagidxstart
```

```
    \let\DTLgidxName\datagidx@invert
    \let\DTLgidxPlace\datagidx@invert
    \let\DTLgidxSubject\datagidx@invert
    \let\DTLgidxOffice\datagidx@invert
    \DTLgidxForeachEntry
     {
       \datagidxitem
     }
    \datagidxend
    \datagidx@postend
   }
   {
    \PackageError { datagidx }
     {
       \token_to_str:N \printterms : ~
       Unable ~ to ~ access ~ `Glossary' ~
       column ~ in ~ ` datagidx ' ~
       database ~ matching ~ ` \l__datagidx_term_database_tl '
     }
     { }
  }
 }
 {
```

Database doesn’t exist.

```
    \PackageError{datagidx}%
    {
      Glossary/index ~ database ~
      `\l__datagidx_term_database_tl' ~ doesn't ~ exist
    }
    {
      You ~ must ~ define ~ the ~ glossary/index ~
      database ~ before ~ you ~ can ~ use ~ it.
    }
  }
 \expandafter \group_end: \if@endpe \@doendpe \fi
}
```

`\datagidx@getgroup` Version 3.0: removed `\datagidx@getgroup`

`\DTLgidxGroupHeaderTitle` Produce the group title from the group label.

```
\newcommand*{\DTLgidxGroupHeaderTitle}[1]{%
  \cs_if_exist_use:cF
  { datagidx #1 name }
  { #1 }
}
```

`\DTLgidxForeachEntry`

`\DTLgidxForeachEntry{`⟨*body*⟩`}`

Iterate through the current database, but only do ⟨*body*⟩ if there is a location or cross-reference.

```
\NewDocumentCommand \DTLgidxForeachEntry { m }
{
 \tl_clear:N \datagidxprevgroup
 \DTLmapdata [ name = \DTLgidxCurrentdb , read-only ]
  {
```

Don't trigger an error if a column key is undefined. This may simply mean that the column was only added for some glossary/index databases and not others. The return value will be null in that case.

```
    \exp_args:NV \__datatool_map_get_values_noerr:n
       \DTLgidxAssignList
       \__datagidx_filter:T
     {
```

Iterate through top-level entries.

```
        \datatool_if_null:NT \Parent
         {
```

The letter group needs to be expanded to allow non-letter groups to merge if they expand to a word (such as Symbols) regardless of their argument.

```
          \legacy_if:nT { datagidxshowgroups }
           {
             \tl_set:Ne \datagidxcurrentgroup
              { \datagidxcurrentgroup }
           }
          \datagidx@doifdisplayed
           {
             \seq_gput_right:NV
               \g__datagidx_refd_labels_seq
               \Label
```

Initialise level.

```
             \int_set_eq:NN \datagidx@level \c_one_int
             #1
             \cs_gset_eq:NN \datagidxprevgroup \datagidxcurrentgroup
           }
         }
     }
  }
}
```

`gidx@checklocationchange` Version 3.0: Removed `\dtlgidx@checklocationchange`

`\datagidx@doifdisplayed`

`\datagidx@doifdisplayed{`⟨*body*⟩`}`

Do ⟨*body*⟩ if entry should appear in the glossary/index. `\Location`, `\See` and `\SeeAlso` must be set before use.

```
\newcommand{\datagidx@doifdisplayed}[1]{
  \__datagidx_unwrap_location:N \Location
  \DTLifnull \Location
  {
     \datatool_if_null_or_empty:NTF \See
     {
       \datatool_if_null_or_empty:NF \SeeAlso { #1 }
     }
     {
```

See is not null, but have any of the cross-referenced items been used?

```
       \clist_map_inline:Nn \See
       {
```

Does the cross-referenced term exist?

```
           \iftermexists { ##1 }
           {
```

Has it been used?

```
             \ifentryused { ##1 }
             {
               #1
```

Break out of loop.

```
               \clist_map_break:
             }
             { }
           }
           { }
       }
     }
  }
  { #1 }
}
```

`\datagidx@level` Keep track of current level

```
\newcount\datagidx@level

\ExplSyntaxOff
```

25 databib.sty

25.1 Package Declaration

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{databib-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{databib}[2026/07/31 v3.4.4 (NLCT)]
```

Version 3.0: no longer using xkeyval.

`\andname` Version 3.0: removed definition of `\andname` (datatool-base now provides `\DTLandname`)

25.2 Package Options

```
\ExplSyntaxOn
```

`\dtlbib@style` The default bib style. Version 3.0 replaced `\dtlbib@style` with

```
\tl_new:N \l__databib_style_tl
\tl_set:Nn \l__databib_style_tl { plain }
```

Provide option to run `bibtex` from the shell escape. Off by default.

```
\bool_new:N \l_databib_auto_build_bool
\bool_set_false:N \l_databib_auto_build_bool
\cs_new:Nn \__databib_auto_build:n
{
 \bool_if:NT \l_databib_auto_build_bool
  {
    \sys_if_shell:TF
     {
       \IfFileExists { #1.aux }
        {
          \sys_shell_now:e { bibtex ~ #1 }
        }
        {
          \PackageWarning { databib }
           {
             File ~ `#1' ~ does ~ not ~ exist. ~
             Rerun ~ may ~ be ~ required
           }
        }
     }
     {
       \PackageWarning { databib }
        {
          Can't ~ run ~ `bibtex #1`: shell ~ not ~ enabled
        }
     }
  }
}
\keys_define:nn { datatool }
{
   style .choice: ,
   style / plain .code:n =
     { \tl_set:Nn \l__databib_style_tl { plain } } ,
   style / abbrv .code:n =
     { \tl_set:Nn \l__databib_style_tl { abbrv } } ,
```

```
    style / alpha .code:n =
      { \tl_set:Nn \l__databib_style_tl { alpha } } ,
    auto .bool_set:N = \l_databib_auto_build_bool ,
  }
```

Switch off LaTeX3 syntax to prevent interference with package loading.

```
\ExplSyntaxOff
```

Process options:

```
\IfPackageLoadedTF{datatool}
{
  \ProcessKeyOptions[datatool]
}
{
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datatool}}
  \ProcessOptions
}
\RequirePackage{datatool}
```

Switch LaTeX3 syntax back on.

```
\ExplSyntaxOn
```

Remove the package option keys so they can't be used with `\DTLsetup` (otherwise they may conflict with databar etc).

```
\keys_define:nn { datatool }
{
  style .undefine: ,
  auto .undefine: ,
}
```

25.3 Loading BBL file

`\DTLloadbbl`

```
\DTLloadbbl[⟨bbl file⟩]{⟨db name⟩}{⟨bib list⟩}
```

```
\NewDocumentCommand \DTLloadbbl { O{\jobname.bbl} m m }
{
  \bibliographystyle {databib}
```

The code below that writes to the aux file is adapted from the definition of `\bibliographystyle`.

```
  \ifx \@begindocumenthook \@undefined
    \bool_if:NT \l_databib_auto_build_bool
    {
      \PackageError { databib }
       {
         \token_to_str:N \DTLloadbbl \c_space_tl
         may ~ only ~ be ~ used ~ in ~ the ~ preamble ~
         with `auto=true'
       }
```

```
        {
          Move ~ \token_to_str:N \DTLloadbbl \c_space_tl
          to ~ the ~ preamble ~ or ~ switch ~ off ~ the ~
          `auto' ~ option
        }
     }
   \else
     \__databib_auto_build:n { \c_sys_jobname_str }
     \expandafter \AtBeginDocument \fi
    {
     \if@filesw
       \immediate \write \@auxout { \string \bibdata { #3 } }
     \fi
    }
```

Version 3.0: allow empty name to indicate the default database.

```
   \tl_if_blank:nTF { #2 }
    {
      \tl_set_eq:NN \DTLBIBdbname \l__datatool_default_dbname_tl
    }
    {
      \tl_set:Nx \DTLBIBdbname { \tl_trim_spaces:n { #2 } }
    }
   \DTLnewdb { \DTLBIBdbname }
   \@input@ { #1 }
 }
```

\DTLnewbibrow `\DTLnewbibrow` adds a new row to the bibliography database. (`\DTLBIBdbname` must be set prior to use to the name of the datatool database which must exist. Any check to determine its existence should be performed when `\DTLBIBdbname` is set.)

```
\NewDocumentCommand \DTLnewbibrow { }
 {
   \@sDTLnewrow { \DTLBIBdbname }
 }
```

\DTLnewbibitem `\DTLnewbibitem{`⟨*key*⟩`}{`⟨*value*⟩`}`

Adds a new database entry with the given key and value.

```
\NewDocumentCommand \DTLnewbibitem { m m }
{
  \@sDTLnewdbentry { \DTLBIBdbname } { #1 } { #2 }
}
```

\DTLnewbibliteralitem `\DTLnewliteralbibitem{`⟨*key*⟩`}{`⟨*value*⟩`}`

For use with fields where the value needs detokenizing. The percent is the most awkward. The rest can be detokenize with `\detokenize` but it will need balanced braces if any occur.

```
\NewDocumentCommand \DTLnewbibliteralitem { m }
{
  \group_begin:
   \char_set_catcode_other:N \%
   \__databib_literal_item:nn { #1 }
}
\cs_new:Nn \__databib_literal_item:nn
{
  \exp_args:NNne
   \group_end:
   \DTLnewbibitem { #1 } { \detokenize { #2 } }
}
```

`\DTLbibsetlongestlabel`

```
\NewDocumentCommand \DTLbibsetlongestlabel { O { \DTLBIBdbname } m }
{
  \tl_set:cn { __databib_longest_label_ #1 _ tl } { #2 }
}
```

`\DTLbibgetlongestlabel`

```
\newcommand{\DTLbibgetlongestlabel}[1]{
  \tl_if_exist:cT {  __databib_longest_label_ #1 _ tl }
   { \tl_use:c { __databib_longest_label_ #1 _ tl } }
}
```

```
\tl_new:N \__databib_longest_label_tl
```

25.4 Predefined text

`\ofname`

```
\providecommand*{\ofname}{of}
```

`\inname`

```
\providecommand*{\inname}{in}
```

`\etalname`

```
\providecommand*{\etalname}{et ~ al.}
```

`\editorname`

```
\providecommand*{\editorname}{editor}
```

`\editorsname`

```
\providecommand*{\editorsname}{editors}
```

`\volumename`

```
\providecommand*{\volumename}{volume}
```

```
\numbername
\providecommand*{\numbername}{number}
\pagesname
\providecommand*{\pagesname}{pages}
\pagename
\providecommand*{\pagename}{page}
\editionname
\providecommand*{\editionname}{edition}
\techreportname
\providecommand*{\techreportname}{Technical ~ report}
\mscthesisname
\providecommand*{\mscthesisname}{Master's ~ thesis}
\phdthesisname
\providecommand*{\phdthesisname}{PhD ~ thesis}
```

25.5 Displaying the bibliography

```
\DTLbibliography{⟨bib dbname⟩}
```

Displays the bibliography for the database ⟨*bib dbname*⟩ which must have previously been loaded using `\DTLloadbbl`.

`\DTLbibliography` Version 3.0: switched to read-only loop.

```
\NewDocumentCommand \DTLbibliography { O{\boolean{true}} m }
{
  \begin{DTLthebibliography} [ #1 ] { #2 }
  \DTLforeachbibentry * [ #1 ] { #2 }
   {
     \DTLbibitem \DTLformatbibentry \DTLendbibitem
   }
  \end{DTLthebibliography}
}
\cs_new:Nn \databib_check_fmt:nT
{
  \tl_if_exist:cTF { DTLformat #1 }
   { #2 }
   {
     \PackageError { databib }
     {
       Don't ~ know ~ how ~ to ~ format ~ bibliography ~
       entries ~ of ~ type ~ ` #1 '
```

```
      }
      {
        The ~ current ~ bibliography ~ format ~ doesn't ~ seem ~ to ~
        support ~ the ~ given ~ entry ~ type
      }
    }
  }
```

\DTLformatbibentry

```
\DTLformatbibentry
```

Formats the current bib entry.

```
\NewDocumentCommand \DTLformatbibentry { }
{
```

Check format for this type is defined.

```
  \databib_check_fmt:nT { \DBIBentrytype }
  {
```

Print information to terminal and log file if in verbose mode.

```
    \dtl@message { [ \DBIBcitekey ] }
```

Initialise:

```
    \legacy_if_set_false:n { DTLstartsentence }
    \legacy_if_set_false:n { DTLmidsentence }
    \legacy_if_set_false:n { DTLperiod }
```

Format this entry

```
    \tl_use:c { DTLformat \DBIBentrytype }
  }
}
```

\gDTLformatbibentry

```
\gDTLformatbibentry
```

Global version.

```
\NewDocumentCommand \gDTLformatbibentry { }
{
```

Check format for this type is defined.

```
  \databib_check_fmt:nT { \DBIBentrytype }
  {
```

Print information to terminal and log file if in verbose mode.

```
    \dtl@message { [ \DBIBcitekey ] }
```

Initialise:

```
    \legacy_if_gset_false:n { DTLstartsentence }
    \legacy_if_gset_false:n { DTLmidsentence }
    \legacy_if_gset_false:n { DTLperiod }
```

Format this entry

```
    \tl_use:c { DTLformat \DBIBentrytype }
  }
}
```

\DTLformatthisbibentry

`\DTLformatthisbibentry{`⟨*db*⟩`}{`⟨*cite key*⟩`}`

Just does `\DTLformatbibentry` for a given entry.

```
\NewDocumentCommand \DTLformatthisbibentry { m m }
{
  \tl_set:Ne \DBIBname { \exp_args:Ne \tl_to_str:n { #1 } }
  \tl_set:Ne \DBIBcitekey { \exp_args:Ne \tl_to_str:n { #2 } }
  \edtlgetrowforvalue { \DBIBname }
    { \dtlcolumnindex { \DBIBname } { CiteKey } }
    { \DBIBcitekey }
  \dtl@gathervalues { \DBIBname } { \dtlcurrentrow }
  \tl_set_eq:Nc \DBIBentrytype { @dtl@key@EntryType }
  \DTLformatbibentry
}
```

\DTLendbibitem Hook to add extra information at the end of a bibliography item. This does nothing by default.

```
\newcommand*{\DTLendbibitem}{}
```

\dtl@widest Define a length to store the widest bib entry label. Version 3.0 replaced `\dtl@widest` with:

```
\dim_new:N \l__databib_widest_dim
```

\DTLcomputewidestbibentry

`\DTLcomputewidestbibentry{`⟨*conditions*⟩`}{`⟨*db name*⟩`}{`⟨*bib label*⟩`}{`⟨*tl var*⟩`}`

Computes the widest bibliography entry over all entries satisfying ⟨*condition*⟩ for the database called ⟨*db name*⟩, where the bibliography label is formatted according to ⟨*bib label*⟩ and stores the result in the token list variable ⟨*tl var*⟩. Version 3.0: switched to read-only loop.

```
\NewDocumentCommand \DTLcomputewidestbibentry { m m m m }
{
 \dim_zero:N \l__databib_widest_dim
 \tl_clear:N #4
 \DTLforeachbibentry * [ #1 ] { #2 }
 {
   \datatool_measure_width:Nn \dtl@tmplength { #3 }
   \dim_compare:nNnT
     { \dtl@tmplength } > { \l__databib_widest_dim }
    {
```

```
      \dim_set_eq:NN \l__databib_widest_dim \dtl@tmplength
      \tl_set:Ne #4 { #3 }
    }
  }
}
```

\DTLforeachbibentry

\DTLforeachbibentry[⟨*condition*⟩]{⟨*db name*⟩}{⟨*body*⟩}

\DTLforeachbibentry*[⟨*condition*⟩]{⟨*db name*⟩}{⟨*body*⟩}

Iterates through the database called ⟨*db name*⟩ and does ⟨*text*⟩ if ⟨*condition*⟩ is met. As with `\DTLforeach`, the starred version is read only. Version 3.0 bug fix: only locally assign `\DBIBcitekey` and `\DBIBentrytype`.

```
\NewDocumentCommand \DTLforeachbibentry { s o m m }
{
  \IfBooleanTF { #1 }
  {
```

Iterate through the database (read only).

```
    \__databib_foreach_entry:Nnnn
     \@sDTLforeach { #2 } { #3 } { #4 }
  }
  {
```

Unstarred version (edits allowed).

```
    \__databib_foreach_entry:Nnnn
      \@DTLforeach { #2 } { #3 } { #4 }
  }
}

\cs_new:Nn \__databib_foreach_entry:Nnnn
{
```

Store database name.

```
  \tl_set:Ne \DBIBname { \exp_args:Ne \tl_to_str:n { #3 } }
```

Reset row counter.

```
  \int_zero:c { c@DTLbibrow }
```

Condition.

```
  \IfValueTF { #2 }
   {
     \tl_if_empty:nTF { #2 }
      {
        \cs_set_eq:NN \__databib_do_foreach:n \use:n
      }
      {
        \cs_set:Nn \__databib_do_foreach:n
```

```
          {
            \ifthenelse { #2 } { ##1 } { }
          }
        }
      }
      {
        \cs_set_eq:NN \__databib_do_foreach:n \use:n
      }
     #1 { #3 } {  }
      {
         \dtl@gathervalues { #3 } { \dtlcurrentrow }
         \tl_set_eq:Nc \DBIBcitekey { @dtl@key@CiteKey }
         \tl_set_eq:Nc \DBIBentrytype { @dtl@key@EntryType }
         \__databib_do_foreach:n
          {
            \refstepcounter{DTLbibrow}
            #4
          }
      }
}
```

\@DTLforeachbibentry Version 3.0: removed `\@DTLforeachbibentry`

\@sDTLforeachbibentry Version 3.0: removed `\@sDTLforeachbibentry`

\gDTLforeachbibentry

`\gDTLforeachbibentry[⟨condition⟩]{⟨db name⟩}{⟨body⟩}`

`\gDTLforeachbibentry*[⟨condition⟩]{⟨db name⟩}{⟨body⟩}`

Global version.

```
\NewDocumentCommand \gDTLforeachbibentry { s O{\boolean{true}} m m }
{
  \IfBooleanTF { #1 }
  {
    \__databib_gforeach_entry:Nnnn
     \@sDTLforeach { #2 } { #3 } { #4 }
  }
  {
    \__databib_gforeach_entry:Nnnn
     \@DTLforeach { #2 } { #3 } { #4 }
  }
}

\cs_new:Nn \__databib_gforeach_entry:Nnnn
{
```

Store database name.

```
  \tl_gset:Ne \DBIBname { \exp_args:Ne \tl_to_str:n { #3 } }
```

Reset row counter.

```
\int_gzero:c { c@DTLbibrow }
```

Iterate through the database.

```
#1 { #3 }
    { \DBIBcitekey = CiteKey , \DBIBentrytype = EntryType }
 {
   \dtl@g@gathervalues { #3 } { \dtlcurrentrow }
   \ifthenelse { #2 }
    {
      \refstepcounter{DTLbibrow}
      #4
    }
    { }
 }
}
```

`\@gDTLforeachbibentry` Version 3.0: removed `\@gDTLforeachbibentry`

`\@sgDTLforeachbibentry` Version 3.0: removed `\@sgDTLforeachbibentry`

`DTLbibrow` The counter DTLbibrow keeps track of the current row in the body of `\DTLforeachbibentry`. (You can't rely on DTLrowi, DTLrowii and DTLrowiii, as `\DTLforeachbibentry` pass the conditions to the optional argument of `\DTLforeach`, but instead uses `\ifthenelse`, which means that DTLrowi etc will be incremented, even when the given condition is not met.)

```
\newcounter{DTLbibrow}
```

`\theHDTLbibrow` Keep hyperref happy:

```
\def\theHDTLbibrow{\theHDTLrow.bib.\arabic{DTLbibrow}}%
```

`\DTLbibfield`

```
\DTLbibfield{⟨field name⟩}
```

Gets the value assigned to the field ⟨*field name*⟩ for the current row of `\DTLforeachbibentry`. This should be expandable but not fully in case the field contains fragile commands.

```
\newcommand*{\DTLbibfield}[1]{
 \tl_if_exist:cT { @dtl@key@#1 }
  {
    \exp_not:v { @dtl@key@#1 }
  }
}
```

`\DTLbibdatefield`

```
\DTLbibdatefield{⟨field name⟩}
```

Used for fields that have dates (Date and UrlDate). This may be redefined to format the date.

```
\newcommand*{\DTLbibdatefield}[1]{
  \DTLbibfield { #1 }
}
```

Provide a means of passing the actual field value to another command. This may be needed for commands that require their argument in a specific format.

\DTLencapbibfield

\DTLencapbibfield{⟨*cs*⟩}{⟨*field*⟩}

```
\newcommand*{\DTLencapbibfield}[2]{
 \tl_if_exist:cT { @dtl@key@#2 }
  {
    \exp_args:Nv #1 { @dtl@key@#2 }
  }
}
```

\DTLbibfieldlet

\DTLbibfield{⟨*cs*⟩}{⟨*field name*⟩}

Gets the value assigned to the field ⟨*field name*⟩ for the current row of `\DTLforeachbibentry` and assigns it to the control sequence ⟨*cs*⟩. (Doesn't check if the field exists, or if it is being used within `\DTLforeachbibentry`.)

```
\NewDocumentCommand \DTLbibfieldlet { m m }
{
  \tl_set_eq:Nc #1 { @dtl@key@#2 }
}
```

\DTLifbibfieldexists

\DTLifbibfieldexists{⟨*field name*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines whether the given field name exists for the current row of `\DTLforeachbibentry`.

```
\newcommand*{\DTLifbibfieldexists}[3]{%
 \tl_if_exist:cTF { @dtl@key@#1 }
  {
    \datatool_if_null:cTF { @dtl@key@ #1 }
     { #3 } { #2 }
  }
  { #3 }
}
```

\DTLifanybibfieldexists

\DTLifanybibfieldexists{⟨*list of field names*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines whether any of the listed fields exist for the current row of `\DTLforeachbibentry`.

```
\NewDocumentCommand \DTLifanybibfieldexists { m m m }
{
  \cs_set_eq:NN \__databib_do:nn \use_ii:nn
  \clist_map_inline:nn { #1 }
   {
     \DTLifbibfieldexists { ##1 }
     {
       \cs_set_eq:NN \__databib_do:nn \use_i:nn
       \clist_map_break:
     }
     { }
   }
  \__databib_do:nn { #2 } { #3 }
}
```

`\ifDTLperiod` The conditional `\ifDTLperiod` is used to keep track of any abbreviations ending with a period, this is to ensure that abbreviations aren't followed by a full stop if they already have a full stop terminating the abbreviation.

```
\newif\ifDTLperiod

\regex_new:N \l_databib_end_sentence_regex
\regex_set:Nn \l_databib_end_sentence_regex
  { [\.?!] \Z }
```

`\DTLcheckendsperiod`

```
\DTLcheckendperiod{⟨string⟩}
```

Checks if ⟨*string*⟩ ends with an end of sentence punctuation mark. This sets `\ifDTLperiod`.

```
\NewDocumentCommand \DTLcheckendsperiod { m }
{
 \regex_match:NnTF \l_databib_end_sentence_regex { #1 }
  { \legacy_if_gset_true:n { DTLperiod } }
  { \legacy_if_gset_false:n { DTLperiod } }
}
```

`Lcheckbibfieldendsperiod`

```
\DTLcheckbibfieldendperiod{⟨label⟩}
```

Checks if the bib field ⟨*label*⟩ ends with a full stop. This sets `\ifDTLperiod`.

```
\NewDocumentCommand \DTLcheckbibfieldendsperiod { m }
{
  \exp_args:Ne \DTLcheckendsperiod { \DTLbibfield { #1 } }
}
```

```
\cs_new:Nn \databib_do_and_check:n
 {
   \legacy_if:nTF { DTLmidsentence }
    { #1 }
    {
      \DTLstartsentencespace
      \text_titlecase:n { #1 }
    }
   \DTLcheckendsperiod { #1 }
 }
\cs_generate_variant:Nn \databib_do_and_check:n { e }
```

\ifDTLmidsentence

\ifDTLmidsentence

Determine whether we are in the middle of a sentence.

```
\newif\ifDTLmidsentence
```

\ifDTLstartsentence

\ifDTLstartsentence

Determine whether we are at the start of a sentence.

```
\newif\ifDTLstartsentence
```

\DTLaddperiod

\DTLaddperiod

Adds a full stop and sets \DTLmidsentencefalse, \DTLstartsentencetrue and \DTLperiodfalse.

```
\newcommand*{\DTLaddperiod}{
 \ifDTLperiod\else.\fi
 \DTLmidsentencefalse
 \DTLperiodfalse
 \DTLstartsentencetrue
}
```

\DTLaddcomma

\DTLaddcomma

Adds a comma and sets \DTLmidsentencetrue, \DTLperiodfalse and \DTLstartsentencefalse

```
\newcommand*{\DTLaddcomma}{
 , ~
 \DTLmidsentencetrue
 \DTLperiodfalse
 \DTLstartsentencefalse
}
```

```
\ExplSyntaxOff
```

\DTLstartsentencespace Adds a space if at the start of the sentence, otherwise does nothing. (The space between sentences is added this way, rather than in \DTLaddperiod otherwise spurious extra space can occur at the end of the bib item. The spacefactor needs to be set manually, because there's stuff in the way of the previous sentence's full stop and this space which confuses the inter sentence spacing (and, of course, the previous sentence could have ended with a capital letter.)

```
\newcommand*{\DTLstartsentencespace}{%
\ifDTLstartsentence\spacefactor=\sfcode`\.\relax\space
\fi\DTLstartsentencefalse}
```

\DTLtwoand In a list of only two author (or editor) names, the text between the two names is given by \DTLtwoand:

```
\newcommand*{\DTLtwoand}{\ \DTLlistand \ }
```

\DTLandlast In a list of author (or editor) names, the text between the penultimate and last name is given by \DTLandlast:

```
\newcommand*{\DTLandlast}{, \DTLlistand \ }

\ExplSyntaxOn
```

\DTLandnotlast In a list of author (or editor) names, the text between the names (except the penultimate and last name) is given by \DTLandnotlast:

```
\newcommand*{\DTLandnotlast}{, ~ }
```

\dtl@authorcount Define a integer variable to keep track of the number of authors. Version 3.0: replaced \dtl@authorcount with:

```
\int_new:N \l__databib_author_count_int
```

DTLmaxauthors The counter DTLmaxauthors indicates the maximum number of author names to display, if there are more than that number, \etalname is used.

```
\newcounter{DTLmaxauthors}
\setcounter{DTLmaxauthors}{10}
```

\DTLformatbibnamelist \DTLformatbibnamelist{⟨*clist*⟩}{⟨*max*⟩}{⟨*formatting cmd*⟩}
Generic formatting command for list of names.

```
\NewDocumentCommand \DTLformatbibnamelist { m m m }
{
 \int_set:Nn \l__databib_author_count_int
   { \clist_count:n { #1 } }
 \int_zero:N \l__datatool_count_int
 \int_compare:nNnTF
    { \l__databib_author_count_int } > { #2 }
  {
    \clist_map_inline:nn { #1 }
    {
      \int_incr:N \l__datatool_count_int
```

```
    \int_compare:nNnTF
       { \l__datatool_count_int } = { \c_one_int }
     {
       \__databib_format_name:Nn #3 { ##1 }
     }
     {
       \int_compare:nNnTF
         { \l__datatool_count_int } > { #2 }
        {
          \DTLandnotlast
          \etalname
          \exp_args:NV \DTLcheckendsperiod \etalname
          \clist_map_break:
        }
        {
          \DTLandnotlast
          \__databib_format_name:Nn #3 { ##1 }
        }
     }
  }
}
{
  \clist_map_inline:nn { #1 }
   {
     \int_incr:N \l__datatool_count_int
     \int_compare:nNnTF { \l__datatool_count_int } = { \c_one_int }
      {
        \__databib_format_name:Nn #3 { ##1 }
      }
      {
        \int_compare:nNnTF
           { \l__datatool_count_int } = { \l__databib_author_count_int }
         {
           \int_compare:nNnTF
             { \l__databib_author_count_int } = { 2 }
            {
              \DTLtwoand
            }
            {
              \DTLandlast
            }
           \__databib_format_name:Nn #3 { ##1 }
         }
         {
           \DTLandnotlast
           \__databib_format_name:Nn #3 { ##1 }
         }
      }
   }
}
```

```
}
```

`\DTLformatauthorlist` Format a list of author names (the list is stored in `\@dtl@key@Author`):

```
\newcommand*{\DTLformatauthorlist}{
 \DTLifbibfieldexists{Author}
 {
   \DTLstartsentencespace
   \exp_args:NV \DTLformatbibnamelist \@dtl@key@Author
    { \c@DTLmaxauthors } { \DTLformatauthor }
 }
 { }
}
```

`DTLmaxeditors` The counter DTLmaxeditors indicates the maximum number of editor names to display, if there are more than that number, `\etalname` is used.

```
\newcounter{DTLmaxeditors}
\setcounter{DTLmaxeditors}{10}
```

`\DTLformateditorlist` Format a list of editor names (the list is stored in `\@dtl@key@Editor`):

```
\newcommand*{\DTLformateditorlist}{%
 \DTLifbibfieldexists {Editor}
  {
   \DTLstartsentencespace
   \exp_args:NV \DTLformatbibnamelist \@dtl@key@Editor
    { \c@DTLmaxeditors } { \DTLformateditor }
    , ~
   \int_compare:nNnTF
      { \l__databib_author_count_int } = { \c_one_int }
    {
      \editorname
      \exp_args:NV \DTLcheckendsperiod \editorname
    }
    {
      \editorsname
      \exp_args:NV \DTLcheckendsperiod \editorsname
    }
  }
  { }
}

\ExplSyntaxOff
```

`\DTLpostvon` Space to insert after ⟨*von part*⟩.

```
\newcommand{\DTLpostvon}{~}
```

`\DTLpostvolnum`

```
\newcommand{\DTLpostvolnum}{:}
```

`\DTLpostpagename`

```
\newcommand{\DTLpostpagename}{~}
```

```
\DTLpostvolumename
                    \newcommand{\DTLpostvolumename}{~}
\DTLpostchaptername
                    \newcommand{\DTLpostchaptername}{~}
\DTLpostnumbername
                    \newcommand{\DTLpostnumbername}{~}
       \DTLprecite
                    \newcommand{\DTLprecite}{~}
      \DTLofseries
                    \newcommand{\DTLofseries}[1]{%
                      \ \ofname\ \DTLofseriesfmt{#1}%
                    }
   \DTLofseriesfmt
                    \newcommand{\DTLofseriesfmt}[1]{\emph{#1}}
      \DTLinseries
                    \newcommand{\DTLinseries}[1]{%
                      \ \inname\ #1%
                    }
    \DTLjournalfmt
                    \newcommand{\DTLjournalfmt}[1]{\emph{#1}}
\DTLinbooktitlefmt
                    \newcommand{\DTLinbooktitlefmt}[1]{\emph{#1}}
\DTLmanualtitlefmt
                    \newcommand{\DTLmanualtitlefmt}[1]{\emph{#1}}
\DTLthesistitlefmt
                    \newcommand{\DTLthesistitlefmt}[1]{\emph{#1}}
\DTLproceedingstitlefmt
                    \newcommand{\DTLproceedingstitlefmt}[1]{\emph{#1}}
                    \ExplSyntaxOn
```

\DTLformatsurnameonly{⟨*von part*⟩}{⟨*surname*⟩}{⟨*jr part*⟩}{⟨*forenames*⟩}

\DTLformatsurnameonly

This is used when only the surname should be displayed. (The final argument, ⟨*forenames*⟩, is ignored.)

```
\NewDocumentCommand \DTLformatsurnameonly { m m m m }
{
  \tl_if_empty:nF { #1 }
   { #1 \DTLpostvon }
  #2
  \tl_if_empty:nTF { #3 }
   {
     \DTLcheckendsperiod { #2 }
   }
   {
     , ~ #3
     \DTLcheckendsperiod { #3 }
   }
}
```

\DTLformatforenames

\DTLformatforenames{⟨*forenames*⟩}

The format of an author/editor's forenames. If the forenames occur at the start of sentence, a new sentence space is added. The argument is checked to determine whether it ends with a full stop (sometimes the forenames may include initials.)

```
\NewDocumentCommand \DTLformatforenames { m }
{
  #1
  \DTLcheckendsperiod { #1 }
}
```

\DTLformatabbrvforenames

\DTLformatabbrvforenames{⟨*forenames*⟩}

The format of an author/editor's abbreviated forenames. The initials may or may not end in a full stop depending on the commands governing the format of `\DTLstoreinitials`, so the initials need to be check using `\DTLcheckendsperiod`.

```
\NewDocumentCommand \DTLformatabbrvforenames { m }
{
  \DTLstoreinitials { #1 } \@dtl@tmp
  \@dtl@tmp
  \exp_args:NV \DTLcheckendsperiod \@dtl@tmp
}
```

\DTLformatvon

\DTLformatvon{⟨*von part*⟩}

The format of the "von" part. This does nothing if the argument is empty, otherwise it does the argument followed by a non-breakable space.

```
\NewDocumentCommand \DTLformatvon { m }
{
  \tl_if_empty:nF { #1 }
   {
     #1 \DTLpostvon
   }
}
```

\DTLformatsurname{⟨*surname*⟩}

\DTLformatsurname

The format of an author/editor's surname.

```
\NewDocumentCommand \DTLformatsurname { m }
{
  #1
  \DTLcheckendsperiod { #1 }
}
```

\DTLformatjr{⟨*jr part*⟩}

\DTLformatjr

The format of the "jr" part. This does nothing if the argument is empty.

```
\NewDocumentCommand \DTLformatjr { m }
{
  \tl_if_empty:nF { #1 }
   {
     , ~ #1
     \DTLcheckendsperiod { #1 }
   }
}
```

\DTLformatcrossrefeditor Format cross reference editors:

```
\NewDocumentCommand \DTLformatcrossrefeditor { }
{
 \DTLifbibfieldexists {Editor}
  {
    \DTLstartsentencespace
    \int_set:Nn \l__databib_author_count_int
     { \clist_count:N \@dtl@key@Editor }
    \int_zero:N \l__datatool_count_int
    \clist_map_inline:Nn \@dtl@key@Editor
     {
       \int_compare:nNnTF
         { \l__databib_author_count_int } = { \c_one_int }
        {
          \__databib_format_name:Nn
            \DTLformatsurnameonly { ##1 }
        }
```

```
        {
          \int_incr:N \l__datatool_count_int
          \int_compare:nNnTF
            { \l__datatool_count_int } = { \c_one_int }
           {
             \__databib_format_name:Nn
               \DTLformatsurnameonly { ##1 }
           }
           {
             \int_compare:nNnTF
               { \l__databib_author_count_int } = { 2 }
              {
                \DTLtwoand
                \__databib_format_name:Nn
                  \DTLformatsurnameonly { ##1 }
              }
              {
                \c_space_tl
                \etalname
                \exp_args:NV \DTLcheckendsperiod \etalname
              }
             \clist_map_break:
           }
        }
     }
  }
  { }
}
```

`\DTLformatvolnumpages` Format volume, number and pages (of an article).

```
\NewDocumentCommand \DTLformatvolnumpages { }
{
 \cs_set_eq:NN \__databib_do:nn \use_ii:nn
 \DTLifbibfieldexists {Volume}
  {
    \DTLstartsentencespace
    \DTLbibfield {Volume}
    \DTLperiodfalse
    \cs_set_eq:NN \__databib_do:nn \use_i:nn
  }
  { }
 \DTLifbibfieldexists {Number}
  {
    \DTLstartsentencespace
    ( \DTLbibfield{Number} )
    \DTLperiodfalse
    \cs_set_eq:NN \__databib_do:nn \use_i:nn
  }
  { }
 \DTLifbibfieldexists {Pages}
```

```
    {
      \__databib_do:nn
       { \DTLpostvolnum }
       {
         \DTLstartsentencespace
         \exp_args:Ne \DTLifnumerical
           { 0 \DTLbibfield {Pages} }
           { \pagename } { \pagesname }
         \DTLpostpagename
       }
      \DTLbibfield {Pages}
      \DTLperiodfalse
    }
    { }
}
```

`\DTLformatbvolume` Format book volume.

```
\NewDocumentCommand \DTLformatbvolume { }
{
 \DTLifbibfieldexists {Volume}
  {
    \legacy_if:nTF { DTLmidsentence }
     {
       \volumename
     }
     {
       \DTLstartsentencespace
       \text_titlecase_first:n { \volumename }
     }
    \DTLpostvolumename
    \DTLbibfield {Volume}
    \DTLifbibfieldexists {Series}
     {
       \DTLofseries { \DTLbibfield{Series} }
       \DTLcheckbibfieldendsperiod {Series}
     }
     {
       \DTLcheckbibfieldendsperiod {Volume}
     }
  }
  { }
}
```

`\DTLformatchapterpages` Format chapter and pages:

```
\NewDocumentCommand \DTLformatchapterpages { }
{
 \DTLifbibfieldexists {Chapter}
  {
    \DTLifbibfieldexists {Type}
     {
```

```
      \DTLstartsentencespace
      \DTLbibfield {Type}
    }
    {
      \DTLstartsentencespace
      \chaptername
    }
   \DTLpostchaptername
   \DTLbibfield {Chapter}
   \DTLifbibfieldexists {Pages}
    { \DTLaddcomma }
    {
      \DTLcheckbibfieldendsperiod {Chapter}
    }
  }
  { }
 \DTLstartsentencespace
 \DTLformatpages
}
```

`\DTLformatpages` Format pages:

```
\NewDocumentCommand \DTLformatpages { }
{
  \DTLifbibfieldexists{Pages}
   {
     \DTLstartsentencespace
     \exp_args:Ne \DTLifnumerical { 0\DTLbibfield{Pages} }
       { \pagename } { \pagesname }
     \DTLpostpagename
     \DTLbibfield {Pages}
     \DTLcheckbibfieldendsperiod {Pages}
   }
   { }
}
```

`\DTLformatnumberseries` Format number and series (of book)

```
\NewDocumentCommand \DTLformatnumberseries { }
{
 \DTLifbibfieldexists {Volume}
  { }
  {
   \DTLifbibfieldexists {Number}
    {
      \legacy_if:nTF { DTLmidsentence }
       {
         \numbername
       }
       {
         \DTLstartsentencespace
         \text_titlecase_first:n { \numbername }
```

```
      }
    \DTLpostnumbername
    \DTLbibfield {Number}
    \DTLifbibfieldexists {Series}
      {
        \DTLinseries { \DTLbibfield {Series} }
        \DTLcheckbibfieldendsperiod {Series}
      }
      {
        \DTLcheckbibfieldendsperiod {Number}
      }
  }
  {
    \DTLifbibfieldexists {Series}
      {
        \DTLstartsentencespace
        \DTLbibfield {Series}
        \DTLcheckbibfieldendsperiod {Series}
      }
      { }
  }
 }
```

`\DTLformatbookcrossref` Format a book cross reference.

```
\NewDocumentCommand \DTLformatbookcrossref { }
{
 \DTLifbibfieldexists {Volume}
  {
    \legacy_if:nTF { DTLmidsentence }
     {
       \volumename
     }
     {
       \DTLstartsentencespace
       \text_titlecase:n { \volumename }
     }
    \DTLpostvolumename
    \DTLbibfield {Volume}
    \c_space_tl \ofname \c_space_tl
  }
  {
    \legacy_if:nTF { ifDTLmidsentence }
     {
       \inname
     }
     {
       \DTLstartsentencespace
       \text_titlecase:n { \inname }
     }
```

```
      \c_space_tl
    }
   \DTLifbibfieldexists {Editor}
    {
      \DTLformatcrossrefeditor
    }
    {
      \DTLifbibfieldexists {Key}
       {
         \DTLbibfield {Key}
       }
       {
         \DTLifbibfieldexists {Series}
          {
            \DTLofseriesfmt { \DTLbibfield {Series} }
          }
          { }
       }
    }
   \DTLprecite
   \DTLpcite { \DTLbibfield{CrossRef} }
  }
```

`formatincollproccrossref` Format ‘incollections’ cross reference.

```
  \NewDocumentCommand \DTLformatincollproccrossref { }
  {
   \DTLifbibfieldexists {Editor}
    {
      \legacy_if:nTF { DTLmidsentence }
       {
         \inname
       }
       {
         \DTLstartsentencespace
         \text_titlecase:n { \inname }
       }
      \c_space_tl
      \DTLformatcrossrefeditor
    }
    {
      \DTLifbibfieldexists {Key}
       {
         \legacy_if:nTF { DTLmidsentence }
          {
            \inname
          }
          {
            \DTLstartsentencespace
            \text_titlecase_first:n { \inname }
          }
```

```
      \c_space_tl
      \DTLbibfield {Key}
    }
    {
      \DTLifbibfieldexists {BookTitle}
       {
         \legacy_if:nTF { DTLmidsentence }
          {
            \inname
          }
          {
            \DTLstartsentencespace
            \text_titlecase_first:n { \inname }
          }
         \c_space_tl
         \DTLformatbooktitle { \DTLbibfield {BookTitle} }
       }
       { }
    }
  }
 \DTLprecite
 \DTLpcite { \DTLbibfield {CrossRef} }
}
```

`\DTLformatinedbooktitle` Format editor and booktitle:

```
\NewDocumentCommand \DTLformatinedbooktitle { }
{
 \DTLifbibfieldexists {BookTitle}
  {
    \legacy_if:nTF { DTLmidsentence }
     {
       \inname
     }
     {
       \DTLstartsentencespace
       \text_titlecase_first:n { \inname }
     }
    \c_space_tl
    \DTLifbibfieldexists {Editor}
     {
       \DTLformateditorlist
       \DTLaddcomma
       \DTLformatbooktitle { \DTLbibfield {BookTitle} }
       \DTLcheckbibfieldendsperiod {BookTitle}
     }
     {
       \DTLformatbooktitle { \DTLbibfield {BookTitle} }
       \DTLcheckbibfieldendsperiod {BookTitle}
     }
  }
```

```
    { }
  }
```

`\DTLformatdate` Format date.

```
\NewDocumentCommand \DTLformatdate { }
{
 \DTLifbibfieldexists {Date}
  {
    \DTLstartsentencespace
    \DTLbibdatefield {Date}
  }
  {
   \DTLifbibfieldexists {Year}
    {
      \DTLifbibfieldexists {Month}
       {
         \legacy_if:nTF { DTLmidsentence }
          {
            \DTLbibfield {Month}
          }
          {
            \DTLstartsentencespace
            \text_titlecase:n { \DTLbibfield {Month} }
          }
        \c_space_tl
        \DTLmidsentencefalse
       }
       { }
      \DTLstartsentencespace
      \DTLbibfield {Year}
    }
    {
      \DTLifbibfieldexists {Month}
       {
         \legacy_if:nTF { DTLmidsentence }
          {
            \DTLbibfield {Month}
          }
          {
            \DTLstartsentencespace
            \text_titlecase_first:n { \DTLbibfield {Month} }
          }
        \DTLcheckbibfieldendsperiod {Month}
       }
       { }
    }
  }
}
```

`DTLformatarticlecrossref` Format article cross reference.

```
                \NewDocumentCommand \DTLformatarticlecrossref { }
                {
                 \DTLifbibfieldexists {Key}
                  {
                    \legacy_if:nTF { DTLmidsentence }
                     {
                       \inname
                     }
                     {
                       \DTLstartsentencespace
                       \text_titlecase_first:n { \inname }
                     }
                    \c_space_tl
                    \DTLjournalfmt { \DTLbibfield {Key} }
                  }
                  {
                    \DTLifbibfieldexists {Journal}
                     {
                       \legacy_if:nTF { DTLmidsentence }
                        {
                          \inname
                        }
                        {
                          \DTLstartsentencespace
                          \text_titlecase_first:n { \inname }
                        }
                       \c_space_tl
                       \DTLjournalfmt { \DTLbibfield{Journal} }
                     }
                     { }
                  }
                 \DTLprecite
                 \DTLpcite { \DTLbibfield{CrossRef} }
                }

        \DTLpcite
                \NewDocumentCommand \DTLpcite { m }
                {
                  \exp_args:Ne \cite { #1 }
                }

                \ExplSyntaxOff

   \DTLbibdoihome
                \newcommand{\DTLbibdoihome}{https://doi.org/}

\DTLbibpubmedhome
                \newcommand{\DTLbibpubmedhome}{https://pubmed.ncbi.nlm.nih.gov/}
```

\DTLbibdoitag

```
\newcommand{\DTLbibdoitag}{\textsc{doi}: }
```

\DTLbibpubmedtag

```
\newcommand{\DTLbibpubmedtag}{\textsc{pmid}: }
```

\DTLbiburldate

\DTLbiburldate{⟨*date*⟩}

Used for UrlDate field.

```
\newcommand*{\DTLbiburldate}[1]{%
  \space (\DTLbibaccessedname : #1)%
}
```

\DTLbibeprints

\DTLbibeprints{⟨*eprint*⟩}{⟨*eprint-type*⟩}

Used for Eprints field.

```
\newcommand*{\DTLbibeprints}[2]{%
 \ifdef\href
  {\href{#1}{#2}}%
  {%
    #2: % space intended
    \ifdef\url{\url{#1}}{\texttt{#1}}%
  }%
}
```

```
\ExplSyntaxOn
```

\DTLbibdoi

\DTLbibdoi{⟨*doi*⟩}

Used for DOI field, which should now have its content sanitized.

```
\newcommand*{\DTLbibdoi}[1]{
 \DTLbibdoitag
 \cs_if_exist:NTF \href
  {
     \exp_args:Ne \href { \DTLbibdoihome #1 } { \nolinkurl { #1 } }
  }
  {
    \cs_if_exist:NTF \url
     { \url { #1 } } { \texttt { #1 } }
  }
}
```

\DTLbiburl

\DTLbiburl{⟨*url*⟩}

Used for Url field, which should now have its content sanitized.

```
\newcommand*{\DTLbiburl}[1]{
  \cs_if_exist:NTF \url
   { \url { #1 } }
   { \texttt { #1 } }
}
```

`\DTLbibaccessedname`

```
\newcommand*{\DTLbibaccessedname}{accessed}
```

`\DTLbibpubmed` `\DTLbibpubmed{⟨id⟩}`

Used for PubMed field.

```
\newcommand*{\DTLbibpubmed}[1]{
  \DTLbibpubmedtag
  \cs_if_exist:NTF \href
   { \exp_args:Ne \href { \DTLbibpubmedhome #1 } { #1 } }
   { #1 }
}
```

`\DTLbibformatdigital` Format the digital elements.

```
\NewDocumentCommand \DTLbibformatdigital { }
{
 \DTLifbibfieldexists {PubMed}
  {
    \exp_args:Ne \DTLbibpubmed { \DTLbibfield { PubMed } }
    \legacy_if_set_true:n { DTLmidsentence }
  }
  { }
 \DTLifbibfieldexists {DOI}
  {
    \legacy_if:nT { DTLmidsentence }
     {
       \DTLaddcomma
     }
    \exp_args:Ne \DTLbibdoi { \DTLbibfield { DOI } }
    \legacy_if_set_true:n { DTLmidsentence }
  }
  { }
 \DTLifbibfieldexists {Url}
  {
    \legacy_if:nT { DTLmidsentence }
     {
       \DTLaddcomma
     }
    \exp_args:Ne \DTLbiburl { \DTLbibfield { Url } }
    \DTLifbibfieldexists {UrlDate}
     {
```

```
        \DTLbiburldate { \DTLbibdatefield { UrlDate } }
      }
      { }
     \legacy_if_set_true:n { DTLmidsentence }
   }
   { }
  \DTLifbibfieldexists {Eprints}
   {
     \legacy_if:nT { DTLmidsentence }
      {
        \DTLaddcomma
      }
     \exp_args:Nee \DTLbibeprints
      { \DTLbibfield { Eprints } }
      {
        \tl_if_exist:NTF \@dtl@key@EprintType
         { \exp_not:V \@dtl@key@EprintType } { eprint }
      }
     \legacy_if_set_true:n { DTLmidsentence }
   }
   { }
 }
```

25.5.1 ifthen conditionals

The conditionals defined in this section may be used in the optional argument of `\DTLforeachbibentry`. They may also be used in the first argument of `\ifthenelse`, but only if the command occurs within the body of `\DTLforeachbibentry`. NB these commands need to expand.

`\DTLbibfieldexists`

`\DTLbibfieldexists{`⟨*field label*⟩`}`

Checks if named bib field exists for current entry

```
\newcommand*{\DTLbibfieldexists}[1]{%
 \TE@throw
 \noexpand \dtl@testbibfieldexists {#1}
 \noexpand \if@dtl@condition
}
```

`\dtl@testbibfieldexists`

```
\newcommand*{\dtl@testbibfieldexists}[1]{%
 \DTLifbibfieldexists {#1}
  { \@dtl@conditiontrue }
  { \@dtl@conditionfalse }
}
```

\DTLbibfieldiseq

```
\DTLbibfieldiseq{⟨field label⟩}{⟨value⟩}
```

Checks if the value of the bib field given by ⟨*field label*⟩ is equal to ⟨*value*⟩. (Uses `\dtlcompare` to determine if the values are equal. If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldiseq}[2]{%
 \TE@throw
 \noexpand
 \dtl@testbibfieldiseq {#1} {#2}
 \noexpand\if@dtl@condition
}
```

\dtl@testbibfieldiseq

```
\newcommand*{\dtl@testbibfieldiseq}[2]{%
 \DTLifbibfieldexists {#1}
  {
    \exp_args:NNv \dtlcompare
       \@dtl@tmpcount
       { @dtl@key@ #1 } { #2 }
    \int_if_zero:nTF { \@dtl@tmpcount }
     {
       \@dtl@conditiontrue
     }
     {
       \@dtl@conditionfalse
     }
  }
  {
    \@dtl@conditionfalse
  }
}
```

\DTLbibfieldislt

```
\DTLbibfieldislt{⟨field label⟩}{⟨value⟩}
```

Checks if the value of the bib field given by ⟨*field label*⟩ is less than ⟨*value*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldislt}[2]{%
 \TE@throw
 \noexpand \dtl@testbibfieldislt { #1 } { #2 }
 \noexpand \if@dtl@condition
}
```

\dtl@testbibfieldislt

```
\newcommand*{\dtl@testbibfieldislt}[2]{%
 \DTLifbibfieldexists {#1}
  {
```

```
    \exp_args:NNv \dtlcompare
       \@dtl@tmpcount
       { @dtl@key@ #1 } { #2 }
    \int_compare:nNnTF { \@dtl@tmpcount } = { -1 }
     {
       \@dtl@conditiontrue
     }
     {
       \@dtl@conditionfalse
     }
  }
  {
    \@dtl@conditionfalse
  }
}
```

\DTLbibfieldisle

\DTLbibfieldisle{⟨*field label*⟩}{⟨*value*⟩}

Checks if the value of the bib field given by ⟨*field label*⟩ is less than or equal to ⟨*value*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldisle}[2]{%
 \TE@throw
 \noexpand \dtl@testbibfieldisle { #1 } { #2 }
 \noexpand \if@dtl@condition
}
```

\dtl@testbibfieldisle

```
\newcommand*{\dtl@testbibfieldisle}[2]{%
 \DTLifbibfieldexists {#1}
  {
    \exp_args:NNv \dtlcompare
       \@dtl@tmpcount
       { @dtl@key@ #1 } { #2 }
    \int_compare:nNnTF { \@dtl@tmpcount } < { \c_one_int }
     {
       \@dtl@conditiontrue
     }
     {
       \@dtl@conditionfalse
     }
  }
  {
    \@dtl@conditionfalse
  }
}
```

\DTLbibfieldisgt

\DTLbibfieldisgt{⟨*field label*⟩}{⟨*value*⟩}

Checks if the value of the bib field given by ⟨*field label*⟩ is greater than ⟨*value*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldisgt}[2]{%
 \TE@throw
 \noexpand \dtl@testbibfieldisgt { #1 } { #2 }
 \noexpand \if@dtl@condition
}
```

\dtl@testbibfieldisgt

```
\newcommand*{\dtl@testbibfieldisgt}[2]{%
 \DTLifbibfieldexists {#1}
  {
    \exp_args:NNv \dtlcompare
       \@dtl@tmpcount
       { @dtl@key@ #1 } { #2 }
    \int_compare:nNnTF { \@dtl@tmpcount } = { \c_one_int }
     {
       \@dtl@conditiontrue
     }
     {
       \@dtl@conditionfalse
     }
  }
  {
    \@dtl@conditionfalse
  }
}
```

\DTLbibfieldisge

\DTLbibfieldisge{⟨*field label*⟩}{⟨*value*⟩}

Checks if the value of the bib field given by ⟨*field label*⟩ is less than or equal to ⟨*value*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldisge}[2]{%
 \TE@throw
 \noexpand \dtl@testbibfieldisge { #1 } { #2 }
 \noexpand \if@dtl@condition
}
```

\dtl@testbibfieldisge

```
\newcommand*{\dtl@testbibfieldisge}[2]{%
 \DTLifbibfieldexists {#1}
  {
    \exp_args:NNv \dtlcompare
       \@dtl@tmpcount
```

```
        { @dtl@key@ #1 } { #2 }
      \int_compare:nNnTF { \@dtl@tmpcount } > { -1 }
       {
         \@dtl@conditiontrue
       }
       {
         \@dtl@conditionfalse
       }
    }
    {
      \@dtl@conditionfalse
    }
}
```

\DTLbibfieldcontains{⟨*field label*⟩}{⟨*sub string*⟩}

\DTLbibfieldcontains Checks if the value of the bib field given by ⟨*field label*⟩ contains ⟨*sub string*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldcontains}[2]{%
 \TE@throw
 \noexpand \dtl@testbibfieldcontains { #1 } { #2 }
 \noexpand \if@dtl@condition
}
```

dtl@testbibfieldcontains

```
\newcommand*{\dtl@testbibfieldcontains}[2]{%
 \DTLifbibfieldexists { #1 }
 {
   \exp_args:Nc \dtl@testifsubstring
     { @dtl@key@ #1 } { #2 }
 }
 { \@dtl@conditionfalse }
}
```

25.6 Bibliography Style Macros

The macros defined in this section should be redefined by bibliography styles.

DTLthebibliography (*env.*) How to format the entire bibliography:

```
\NewDocumentEnvironment { DTLthebibliography }
 { O{\boolean{true}} m }
{
  \tl_clear:N \__databib_longest_label_tl
  \tl_if_eq:nnT { #1 } { \boolean{true} }
   {
    \tl_if_exist:cT { __databib_longest_label_ #1 _ tl }
     {
```

```
        \tl_set_eq:Nc
          \__databib_longest_label_tl
         { __databib_longest_label_ #1 _ tl }
      }
    }
   \tl_if_empty:NT \__databib_longest_label_tl
    {
      \int_zero:N \l__datatool_count_int
      \@sDTLforeach [ #1 ] { #2 } { }
       { \int_incr:N \l__datatool_count_int }
      \tl_set:NV \__databib_longest_label_tl
        \l__datatool_count_int
    }
   \exp_args:NnV \begin { thebibliography }
    \__databib_longest_label_tl
}
{ \end {thebibliography} }
```

`\DTLmonthname` The monthname style. The argument must be a number from 1 to 12. By default, uses `\dtl@monthname`.

```
\newcommand*{\DTLmonthname}[1]{%
\dtl@monthname{#1}}
```

Ensure the value is the numeric argument if sorting by month.

```
\dtlSortWordCommands{\let\DTLmonthname\@firstofone}
```

`\dtl@monthname` Full month names:

```
\newcommand*{\dtl@monthname}[1]{%
  \ifcase#1%
  \or January%
  \or February%
  \or March%
  \or April%
  \or May%
  \or June%
  \or July%
  \or August%
  \or September%
  \or October%
  \or November%
  \or December%
  \fi
}
```

`\dtl@abbrvmonthname` Abbreviated months:

```
\newcommand*{\dtl@abbrvmonthname}[1]{%
  \ifcase#1%
   \or Jan.%
   \or Feb.%
```

```
    \or Mar.%
    \or Apr.%
    \or May%
    \or June%
    \or July%
    \or Aug.%
    \or Sept.%
    \or Oct.%
    \or Nov.%
    \or Dec.%
   \fi
}
```

`\DTLbibitem` Define how to start a new bibitem:

```
\newcommand*{\DTLbibitem}{\bibitem{\DBIBcitekey}}
```

`\DTLmbibitem` As `\DTLbibitem` but for `\DTLmbibliography`

```
\newcommand*{\DTLmbibitem}[1]{\bibitem{#1@\DBIBcitekey}}
```

`\DTLcustombibitem`

`\DTLcustombibitem{`⟨*item code*⟩`}{`⟨*ref text*⟩`}{`⟨*cite key*⟩`}`

As `\DTLbibitem` but user provides ⟨*item code*⟩ to use in place of `\item`. This code can access the cite key using `\DBIBcitekey`. The ⟨*ref text*⟩ is the text associated with this bib item. (For example, if used in an enumerate environment, ⟨*ref text*⟩ might be `\theenumi`.)

```
\NewDocumentCommand \DTLcustombibitem { m m m }
{%
  #1%
  \if@filesw
    \immediate\write\@auxout{\string\bibcite{#3}{#2}}%
  \fi
  \ignorespaces
}
```

`\DTLformatauthor`

`\DTLformatauthor{`⟨*von part*⟩`}{`⟨*surname*⟩`}{`⟨*junior part*⟩`}{`⟨*forenames*⟩`}`

The format of an author's name.

```
\newcommand*{\DTLformatauthor}[4]{%
 \DTLformatforenames { #4 } ~
 \DTLformatvon { #1 }
 \DTLformatsurname { #2 }
 \DTLformatjr { #3 }
}
```

Syntax: ⟨*cs*⟩ {⟨*name specs*⟩} where ⟨*name specs*⟩ is in the form {⟨*von*⟩}{⟨*surname*⟩}{⟨*jr*⟩}{⟨*forename*⟩}

```
\cs_new:Nn \__databib_format_name:Nn
 {
   #1 #2
 }
```

`\DTLformateditor` The format of an editor's name.

```
\newcommand*{\DTLformateditor}[4]{%
 \DTLformatforenames { #4 } ~
 \DTLformatvon { #1 }
 \DTLformatsurname { #2 }
 \DTLformatjr { #3 }
}
```

`\DTLformatedition` The format of an edition:

```
\newcommand*{\DTLformatedition}[1]{#1 ~ \editionname}
```

The following will be redefined by the style.

`\DTLformatarticle` The format of an article:

```
\newcommand{\DTLformatarticle}{}
```

`\DTLformatbook` The format of a book:

```
\newcommand{\DTLformatbook}{}
```

`\DTLformatbooklet` The format of a booklet:

```
\newcommand{\DTLformatbooklet}{}
```

`\DTLformatinbook` The format of an "inbook" type:

```
\newcommand{\DTLformatinbook}{}
```

`\DTLformatincollection` The format of an "incollection" type:

```
\newcommand{\DTLformatincollection}{}
```

`\DTLformatinproceedings` The format of an "inproceedings" type:

```
\newcommand{\DTLformatinproceedings}{}
```

`\DTLformatmanual` The format of a manual:

```
\newcommand{\DTLformatmanual}{}
```

`\DTLformatmastersthesis` The format of a master's thesis:

```
\newcommand{\DTLformatmastersthesis}{}
```

`\DTLformatmisc` The format of a miscellaneous entry:

```
\newcommand{\DTLformatmisc}{}
```

`\DTLformatphdthesis` The format of a Ph.D. thesis:

```
\newcommand{\DTLformatphdthesis}{}
```

\DTLformatproceedings The format of a proceedings:

```
\newcommand{\DTLformatproceedings}{}
```

\DTLformattechreport The format of a technical report:

```
\newcommand{\DTLformattechreport}{}
```

\DTLformatunpublished The format of an unpublished work:

```
\newcommand{\DTLformatunpublished}{}
```

```
\ExplSyntaxOff
```

Predefined names (these correspond to the standard BibTeX predefined strings of the same name without the leading \DTL):

\DTLacmcs

```
\newcommand*{\DTLacmcs}{ACM Computing Surveys}
```

\DTLacta

```
\newcommand*{\DTLacta}{Acta Informatica}
```

\DTLcacm

```
\newcommand*{\DTLcacm}{Communications of the ACM}
```

\DTLibmjrd

```
\newcommand*{\DTLibmjrd}{IBM Journal of Research and Development}
```

\DTLibmsj

```
\newcommand*{\DTLibmsj}{IBM Systems Journal}
```

\DTLieeese

```
\newcommand*{\DTLieeese}{IEEE Transactions on Software Engineering}
```

\DTLieeetc

```
\newcommand*{\DTLieeetc}{IEEE Transactions on Computers}
```

\DTLieeetcad

```
\newcommand*{\DTLieeetcad}{IEEE Transactions on Computer-Aided Design
of Integrated Circuits}
```

\DTLipl

```
\newcommand*{\DTLipl}{Information Processing Letters}
```

\DTLjacm

```
\newcommand*{\DTLjacm}{Journal of the ACM}
```

\DTLjcss

```
\newcommand*{\DTLjcss}{Journal of Computer and System Sciences}
```

```
          \DTLscp
                    \newcommand*{\DTLscp}{Science of Computer Programming}

        \DTLsicomp
                    \newcommand*{\DTLsicomp}{SIAM Journal on Computing}

         \DTLtocs
                    \newcommand*{\DTLtocs}{ACM Transactions on Computer Systems}

         \DTLtods
                    \newcommand*{\DTLtods}{ACM Transactions on Database Systems}

          \DTLtog
                    \newcommand*{\DTLtog}{ACM Transactions on Graphics}

         \DTLtoms
                    \newcommand*{\DTLtoms}{ACM Transactions on Mathematical Software}

         \DTLtoois
                    \newcommand*{\DTLtoois}{ACM Transactions on Office Information
                    Systems}

        \DTLtoplas
                    \newcommand*{\DTLtoplas}{ACM Transactions on Programming Languages
                    and Systems}

          \DTLtcs
                    \newcommand*{\DTLtcs}{Theoretical Computer Science}

\DTLresetpredefined
                    \newcommand{\DTLresetpredefined}{%
                      \renewcommand*{\DTLacmcs}{ACM Computing Surveys}
                      \renewcommand*{\DTLacta}{Acta Informatica}
                      \renewcommand*{\DTLcacm}{Communications of the ACM}
                      \renewcommand*{\DTLibmjrd}{IBM Journal of Research and Development}
                      \renewcommand*{\DTLibmsj}{IBM Systems Journal}
                      \renewcommand*{\DTLieeese}{IEEE Transactions on Software Engineering}
                      \renewcommand*{\DTLieeetc}{IEEE Transactions on Computers}
                      \renewcommand*{\DTLieeetcad}{IEEE Transactions on Computer-Aided Design
                    of Integrated Circuits}
                      \renewcommand*{\DTLipl}{Information Processing Letters}
                      \renewcommand*{\DTLjacm}{Journal of the ACM}
                      \renewcommand*{\DTLjcss}{Journal of Computer and System Sciences}
                      \renewcommand*{\DTLscp}{Science of Computer Programming}
                      \renewcommand*{\DTLsicomp}{SIAM Journal on Computing}
                      \renewcommand*{\DTLtocs}{ACM Transactions on Computer Systems}
                      \renewcommand*{\DTLtods}{ACM Transactions on Database Systems}
                      \renewcommand*{\DTLtog}{ACM Transactions on Graphics}
```

```
  \renewcommand*{\DTLtoms}{ACM Transactions on Mathematical Software}
  \renewcommand*{\DTLtoois}{ACM Transactions on Office Information
Systems}
  \renewcommand*{\DTLtoplas}{ACM Transactions on Programming Languages
and Systems}
  \renewcommand*{\DTLtcs}{Theoretical Computer Science}
}
```

`\DTLresetpredefinedabbrv`

```
\newcommand{\DTLresetpredefinedabbrv}{%
  \renewcommand*{\DTLacmcs}{ACM Comput.\ Surv.}%
  \renewcommand*{\DTLacta}{Acta Inf.}%
  \renewcommand*{\DTLcacm}{Commun.\ ACM}%
  \renewcommand*{\DTLibmjrd}{IBM J.\ Res.\ Dev.}%
  \renewcommand*{\DTLibmsj}{IBM Syst.~J.}%
  \renewcommand*{\DTLieeese}{IEEE Trans. Softw.\ Eng.}%
  \renewcommand*{\DTLieeetc}{IEEE Trans.\ Comput.}%
  \renewcommand*{\DTLieeetcad}{IEEE Trans.\ Comput.-Aided Design
     Integrated Circuits}%
  \renewcommand*{\DTLipl}{Inf.\ Process.\ Lett.}%
  \renewcommand*{\DTLjacm}{J.~ACM}%
  \renewcommand*{\DTLjcss}{J.~Comput.\ Syst.\ Sci.}%
  \renewcommand*{\DTLscp}{Sci.\ Comput.\ Programming}%
  \renewcommand*{\DTLsicomp}{SIAM J.~Comput.}%
  \renewcommand*{\DTLtocs}{ACM Trans.\ Comput.\ Syst.}%
  \renewcommand*{\DTLtods}{ACM Trans.\ Database Syst.}%
  \renewcommand*{\DTLtog}{ACM Trans.\ Gr.}%
  \renewcommand*{\DTLtoms}{ACM Trans.\ Math. Softw.}%
  \renewcommand*{\DTLtoois}{ACM Trans. Office Inf.\ Syst.}%
  \renewcommand*{\DTLtoplas}{ACM Trans.\ Prog. Lang.\ Syst.}%
  \renewcommand*{\DTLtcs}{Theoretical Comput.\ Sci.}%
}

\ExplSyntaxOn
```

25.7 Bibliography Styles

Each bibliography style is set by the command `\dtlbst@`⟨*style*⟩, where ⟨*style*⟩ is the name of the bibliography style.

`\dtlbst@plain` The ‘plain’ style:

```
\newcommand{\dtlbst@plain}{%
```

Set how to format the entire bibliography:

```
  \RenewDocumentEnvironment { DTLthebibliography }
   { O{\boolean{true}} m }
  {
    \int_zero:N \l__datatool_count_int
    \@sDTLforeach [ ##1 ] { ##2 } { }
     { \int_incr:N \l__datatool_count_int }
```

```
    \exp_args:NnV \begin { thebibliography }
      \l__datatool_count_int
  }
  { \end {thebibliography} }
```

Set how to start the bibliography entry:

```
  \renewcommand*{\DTLbibitem}{\bibitem{\DBIBcitekey}}%
  \renewcommand*{\DTLmbibitem}[1]{\bibitem{##1@\DBIBcitekey}}%
```

Sets the author name format.

```
  \renewcommand*{\DTLformatauthor}[4]{
    \DTLformatforenames {##4} ~
    \DTLformatvon {##1}
    \DTLformatsurname {##2}
    \DTLformatjr {##3}
  }
```

Sets the editor name format.

```
  \renewcommand*{\DTLformateditor}[4]{
    \DTLformatforenames {##4} ~
    \DTLformatvon {##1}
    \DTLformatsurname {##2}
    \DTLformatjr {##3}
  }
```

Sets the edition format.

```
  \renewcommand*{\DTLformatedition}[1]{##1 ~ \editionname}%
```

Sets the monthname format.

```
  \let\DTLmonthname\dtl@monthname
```

Sets other predefined names:

```
  \DTLresetpredefined
```

The format of an article.

```
  \renewcommand*{\DTLformatarticle}{%
    \DTLformatauthorlist
    \DTLifbibfieldexists {Author}
     { \DTLaddperiod } { }
    \DTLifbibfieldexists {Title}
     {
       \DTLstartsentencespace
       \DTLbibfield {Title}
       \DTLcheckbibfieldendsperiod {Title}
       \DTLaddperiod
     }
     { }
    \DTLifbibfieldexists{CrossRef}
     {
```

Cross ref field

```
       \DTLformatarticlecrossref
       \DTLifbibfieldexists {Pages}
```

```
          {\DTLaddcomma} { }
        \DTLformatpages
        \DTLaddperiod
      }
      {
```

No cross ref field

```
        \DTLifbibfieldexists {Journal}
         {
           \DTLstartsentencespace
           \DTLjournalfmt { \DTLbibfield{Journal} }
           \DTLcheckbibfieldendsperiod {Journal}
           \DTLifanybibfieldexists {Number,Volume,Pages,Month,Year}
            { \DTLaddcomma }
            { \DTLaddperiod }
         }
         { }
        \DTLformatvolnumpages
        \DTLifanybibfieldexists {Volume,Number,Pages}
         {
           \DTLifanybibfieldexists {Year,Month}
            { \DTLaddcomma }
            { \DTLaddperiod }
           \DTLmidsentencefalse
         }
         { }
        \DTLformatdate
        \DTLifanybibfieldexists {Year,Month}
         { \DTLaddperiod } { }
      }
     \DTLifbibfieldexists {Note}
      {
       \DTLstartsentencespace
       \DTLbibfield {Note}
       \DTLcheckbibfieldendsperiod {Note}
       \DTLaddperiod
      }
      { }
  }
```

The format of a book.

```
  \renewcommand*{\DTLformatbook}{
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
      \DTLaddperiod
    }
    {
      \DTLformateditorlist
      \DTLifbibfieldexists {Editor}
      {
```

```
        \DTLaddperiod
       }
       { }
     }
   \DTLifbibfieldexists {Title}
    {
     \DTLstartsentencespace
     \DTLformatbooktitle { \DTLbibfield {Title} }
     \DTLcheckbibfieldendsperiod {Title}
    }
    { }
   \DTLifbibfieldexists{CrossRef}
    {
```
Cross ref field
```
     \DTLifbibfieldexists {Title}
       { \DTLaddperiod } { }
     \DTLformatbookcrossref
     \DTLifanybibfieldexists {Edition,Month,Year}
      { \DTLaddcomma }
      { \DTLaddperiod }
    }
    {
```
no cross ref field
```
     \DTLifbibfieldexists {Title}
      {
       \DTLifbibfieldexists {Volume}
         { \DTLaddcomma }
         { \DTLaddperiod }
      }
      { }
     \DTLformatbvolume
     \DTLformatnumberseries
     \DTLifanybibfieldexists {Number,Series,Volume}
      { \DTLaddperiod } { }
     \DTLifbibfieldexists {Publisher}
      {
       \DTLstartsentencespace
       \DTLbibfield {Publisher}
       \DTLcheckbibfieldendsperiod {Publisher}
       \DTLifbibfieldexists {Address}
        { \DTLaddcomma }
        {
          \DTLifanybibfieldexists {Month,Year}
          { \DTLaddcomma }
          { \DTLaddperiod }
        }
      }
      { }
     \DTLifbibfieldexists{Address}
```

```
      {
       \DTLstartsentencespace
       \DTLbibfield {Address}
       \DTLcheckbibfieldendsperiod {Address}
       \DTLifanybibfieldexists {Month,Year}
        { \DTLaddcomma }
        { \DTLaddperiod }
      }
      { }
    }
   \DTLifbibfieldexists{Edition}
    {
     \databib_do_and_check:e
       { \DTLformatedition { \DTLbibfield {Edition} } }
     \DTLifanybibfieldexists { Month, Year }
       { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLformatdate
   \DTLifanybibfieldexists {Year,Month}
     { \DTLaddperiod } { }
   \DTLifbibfieldexists {Note}
    {
      \DTLstartsentencespace
      \DTLbibfield {Note}
      \DTLcheckbibfieldendsperiod{Note}
      \DTLaddperiod
    }
    { }
  }
```

The format of a booklet.

```
   \renewcommand*{\DTLformatbooklet}{
     \DTLifbibfieldexists {Author}
      {
        \DTLformatauthorlist
        \DTLaddperiod
      }
      { }
     \DTLifbibfieldexists {Title}
      {
        \DTLstartsentencespace
        \DTLbibfield {Title}
        \DTLcheckbibfieldendsperiod {Title}
        \DTLaddperiod
      }
      { }
     \DTLifbibfieldexists {HowPublished}
      {
        \DTLstartsentencespace
```

```
      \DTLbibfield {HowPublished}
      \DTLcheckbibfieldendsperiod {HowPublished}
      \DTLifanybibfieldexists {Address,Month,Year}
       { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLifbibfieldexists {Address}
    {
      \DTLstartsentencespace
      \DTLbibfield {Address}
      \DTLcheckbibfieldendsperiod {Address}
      \DTLifanybibfieldexists {Month,Year}
        { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLformatdate
   \DTLifanybibfieldexists {Year,Month}
    { \DTLaddperiod } { }
   \DTLifbibfieldexists {Note}
    {
      \DTLstartsentencespace
      \DTLbibfield{Note}
      \DTLcheckbibfieldendsperiod {Note}
      \DTLaddperiod
    }
    { }
  }
```

The format of an ‘inbook’ entry.

```
\renewcommand*{\DTLformatinbook}
 {
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
      \DTLaddperiod
    }
    {
      \DTLifbibfieldexists {Editor}
       {
         \DTLformateditorlist
         \DTLaddperiod
       }
       { }
    }
   \DTLifbibfieldexists {Title}
    {
      \DTLstartsentencespace
      \DTLinbooktitlefmt { \DTLbibfield{Title} }
      \DTLcheckbibfieldendsperiod {Title}
    }
```

```
      { }
     \DTLifbibfieldexists {CrossRef}
      {
```
Cross ref entry
```
        \DTLifbibfieldexists {Title}
         {
           \DTLifbibfieldexists {Chapter}
            { \DTLaddcomma } { \DTLaddperiod }
         }
         { }
        \DTLformatchapterpages
        \DTLifanybibfieldexists {Chapter,Pages}
         { \DTLaddperiod } { }
        \DTLformatbookcrossref
      }
      {
```
No cross ref
```
        \DTLifbibfieldexists {Title}
         {
           \DTLifanybibfieldexists {Chapter,Volume}
            { \DTLaddcomma } { \DTLaddperiod }
         }
         { }
        \DTLformatbvolume
        \DTLifanybibfieldexists {Volume,Series}
         {
           \DTLifanybibfieldexists {Chapter,Pages}
            { \DTLaddcomma } { \DTLaddperiod }
         }
         { }
        \DTLformatchapterpages
        \DTLifanybibfieldexists {Chapter,Pages}
         { \DTLaddperiod } { }
        \DTLifbibfieldexists {Publisher}
         {
           \DTLstartsentencespace
           \DTLbibfield {Publisher}
           \DTLcheckbibfieldendsperiod {Publisher}
           \DTLifbibfieldexists {Address}
            { \DTLaddcomma } { }
         }
         { }
        \DTLifbibfieldexists{Address}
         {
           \DTLstartsentencespace
           \DTLbibfield {Address}
           \DTLcheckbibfieldendsperiod {Address}
         }
         { }
```

```
      }
    \DTLifanybibfieldexists {Edition,Month,Year}
     { \DTLaddcomma } { \DTLaddperiod }
    \DTLifbibfieldexists {Edition}
     {
       \databib_do_and_check:e
        { \DTLformatedition { \DTLbibfield{Edition} } }
       \DTLifanybibfieldexists {Month,Year}
        { \DTLaddcomma } { \DTLaddperiod }
     }
     { }
    \DTLformatdate
    \DTLifanybibfieldexists {Month,Year}
     { \DTLaddperiod } { }
    \DTLifbibfieldexists {Note}
     {
       \DTLstartsentencespace
       \DTLbibfield {Note}
       \DTLcheckbibfieldendsperiod {Note}
       \DTLaddperiod
     }
     { }
  }
```

The format of an ‘incollection’ entry.

```
\renewcommand*{\DTLformatincollection}
 {
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Title}
    {
      \DTLstartsentencespace
      \DTLbibfield {Title}
      \DTLcheckbibfieldendsperiod {Title}
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {CrossRef}
    {
```

Cross ref entry

```
      \DTLformatincollproccrossref
      \DTLifanybibfieldexists {Chapter,Pages}
       { \DTLaddcomma } { }
      \DTLformatchapterpages
      \DTLaddperiod
    }
```

```
        {
No cross ref entry
          \DTLformatinedbooktitle
          \DTLifbibfieldexists {BookTitle}
           {
             \DTLifanybibfieldexists
               {Volume,Series,Chapter,Pages,Number}
              { \DTLaddcomma } { \DTLaddperiod }
           }
           { }
          \DTLformatbvolume
          \DTLifbibfieldexists {Volume}
           {
             \DTLifanybibfieldexists
               {Number,Series,Chapter,Pages}
              { \DTLaddcomma } { \DTLaddperiod }
           }
           { }
          \DTLformatnumberseries
          \DTLifanybibfieldexists {Number,Series}
           {
            \DTLifanybibfieldexists {Chapter,Pages}
             { \DTLaddcomma } { \DTLaddperiod }
           }
           { }
          \DTLformatchapterpages
          \DTLifanybibfieldexists {Chapter,Pages}
           { \DTLaddperiod } { }
          \DTLifbibfieldexists {Publisher}
           {
             \DTLstartsentencespace
             \DTLbibfield {Publisher}
             \DTLcheckbibfieldendsperiod {Publisher}
             \DTLifanybibfieldexists {Address,Edition,Month,Year}
              { \DTLaddcomma } { \DTLaddperiod }
           }
           { }
          \DTLifbibfieldexists{Address}
           {
             \DTLstartsentencespace
             \DTLbibfield {Address}
             \DTLcheckbibfieldendsperiod {Address}
             \DTLifanybibfieldexists {Edition,Month,Year}
              { \DTLaddcomma } { \DTLaddperiod }
           }
           { }
          \DTLifbibfieldexists {Edition}
           {
             \databib_do_and_check:e
```

```
            { \DTLformatedition { \DTLbibfield {Edition} } }
           \DTLifanybibfieldexists {Month,Year}
            { \DTLaddcomma } { \DTLaddperiod }
         }
         { }
        \DTLformatdate
        \DTLifanybibfieldexists {Month,Year}
         { \DTLaddperiod } { }
      }
     \DTLifbibfieldexists {Note}
      {
        \DTLstartsentencespace
        \DTLbibfield {Note}
        \DTLcheckbibfieldendsperiod {Note}
        \DTLaddperiod
      }
      { }
  }%
```

The format of an ‘inproceedings’ entry.

```
\renewcommand*{\DTLformatinproceedings}
 {
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Title}
    {
      \DTLstartsentencespace
      \DTLbibfield {Title}
      \DTLcheckbibfieldendsperiod {Title}
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {CrossRef}
    {
```

Cross ref entry

```
     \DTLformatincollproccrossref
     \DTLifbibfieldexists {Pages}
      { \DTLaddcomma } { \DTLaddperiod }
     \DTLformatpages
     \DTLifbibfieldexists {Pages}
      { \DTLaddperiod } { }
    }
    {
```

No cross ref

```
      \DTLformatinedbooktitle
```

```
\DTLifbibfieldexists {BookTitle}
 {
   \DTLifanybibfieldexists
    { Volume,Series,Pages,Number,Address,Month,Year}
   { \DTLaddcomma } { \DTLaddperiod }
 }
 { }
\DTLformatbvolume
\DTLifbibfieldexists {Volume}
 {
   \DTLifanybibfieldexists
    { Number,Series,Pages,Address,Month,Year }
   { \DTLaddcomma } { \DTLaddperiod }
 }
 { }
\DTLformatnumberseries
\DTLifanybibfieldexists {Number,Series}
 {
   \DTLifanybibfieldexists
     {Pages,Address,Month,Year}
    { \DTLaddcomma } { \DTLaddperiod }
 }
 { }
\DTLformatpages
\DTLifbibfieldexists {Pages}
 {
   \DTLifanybibfieldexists {Address,Month,Year}
    { \DTLaddcomma } { \DTLaddperiod }
 }
 { }
\DTLifbibfieldexists {Address}
 {
   \DTLstartsentencespace
   \DTLbibfield {Address}
   \DTLcheckbibfieldendsperiod {Address}
   \DTLifanybibfieldexists {Month,Year}
    { \DTLaddcomma } { \DTLaddperiod }
   \DTLformatdate
   \DTLifanybibfieldexists {Month,Year}
    { \DTLaddperiod } { }
   \DTLifbibfieldexists {Organization}
    {
      \DTLstartsentencespace
      \DTLbibfield {Organization}
      \DTLcheckbibfieldendsperiod {Organization}
      \DTLifbibfieldexists {Publisher}
       { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLifbibfieldexists {Publisher}
```

```
            {
              \DTLstartsentencespace
              \DTLbibfield {Publisher}
              \DTLcheckbibfieldendsperiod {Publisher}
              \DTLaddperiod
            }
            { }
          }
          {
            \DTLifanybibfieldexists {Publisher,Organization}
             { \DTLaddperiod } { }
            \DTLifbibfieldexists {Organization}
             {
               \DTLstartsentencespace
               \DTLbibfield {Organization}
               \DTLcheckbibfieldendsperiod {Organization}
               \DTLifanybibfieldexists {Publisher,Month,Year}
                { \DTLaddcomma } { }
             }
             { }
            \DTLifbibfieldexists {Publisher}
             {
               \DTLstartsentencespace
               \DTLbibfield{Publisher}
               \DTLcheckbibfieldendsperiod {Publisher}
               \DTLifanybibfieldexists {Month,Year}
                { \DTLaddcomma } { \DTLaddperiod }
             }
             { }
            \DTLformatdate
            \DTLifanybibfieldexists {Month,Year}
             { \DTLaddperiod } { }
          }
        }
       \DTLifbibfieldexists{Note}
        {
          \DTLstartsentencespace
          \DTLbibfield {Note}
          \DTLcheckbibfieldendsperiod {Note}
          \DTLaddperiod
        }
        { }
      }
```

The format of a manual.

```
    \renewcommand*{\DTLformatmanual}
     {
       \DTLifbibfieldexists {Author}
        {
          \DTLformatauthorlist
```

```
    \DTLaddperiod
  }
  {
    \DTLifbibfieldexists {Organization}
     {
       \DTLstartsentencespace
       \DTLbibfield {Organization}
       \DTLcheckbibfieldendsperiod {Organization}
       \DTLifbibfieldexists {Address}
        {
          \DTLaddcomma
          \DTLbibfield {Address}
          \DTLcheckbibfieldendsperiod {Address}
        }
        { }
       \DTLaddperiod
     }
     { }
  }
\DTLifbibfieldexists {Title}
  {
    \DTLstartsentencespace
    \DTLmanualtitlefmt { \DTLbibfield{Title} }
    \DTLcheckbibfieldendsperiod {Title}
    \DTLifbibfieldexists {Author}
     {
       \DTLifanybibfieldexists {Organization,Address}
        { \DTLaddperiod } { \DTLaddcomma }
     }
     {
       \DTLifanybibfieldexists
         {Organization,Address,Edition,Month,Year}
        { \DTLaddcomma } { \DTLaddperiod }
     }
  }
  { }
\DTLifbibfieldexists {Author}
  {
    \DTLifbibfieldexists {Organization}
     {
       \DTLstartsentencespace
       \DTLbibfield{Organization}%
       \DTLcheckbibfieldendsperiod{Organization}%
       \DTLifanybibfieldexists {Address,Edition,Month,Year}
        { \DTLaddcomma } { \DTLaddperiod }
     }
     { }
    \DTLifbibfieldexists {Address}
     {
       \DTLstartsentencespace
```

```
            \DTLbibfield {Address}
            \DTLcheckbibfieldendsperiod {Address}
            \DTLifanybibfieldexists
              {Edition,Month,Year}
             { \DTLaddcomma } { \DTLaddperiod }
          }
          { }
        }
        {
          \DTLifbibfieldexists {Organization} { }
           {
             \DTLifbibfieldexists {Address}
              {
                \DTLstartsentencespace
                \DTLbibfield {Address}
                \DTLcheckbibfieldendsperiod {Address}
                \DTLifanybibfieldexists
                  {Edition,Month,Year}
                 { \DTLaddcomma } { \DTLaddperiod }
              }
              { }
           }
        }
       \DTLifbibfieldexists {Edition}
        {
         \databib_do_and_check:e
          { \DTLformatedition { \DTLbibfield {Edition} } }
         \DTLifanybibfieldexists {Month,Year}
          { \DTLaddcomma } { \DTLaddperiod }
        }
        { }
       \DTLformatdate
       \DTLifanybibfieldexists {Month,Year}
        { \DTLaddperiod } { }
       \DTLifbibfieldexists {Note}
        {
          \DTLstartsentencespace
          \DTLbibfield {Note}
          \DTLcheckbibfieldendsperiod {Note}
          \DTLaddperiod
        }
        { }
     }
```

The format of a master's thesis.

```
\renewcommand*{\DTLformatmastersthesis}
 {
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
```

```
    \DTLaddperiod
  }
  { }
\DTLifbibfieldexists {Title}
  {
    \DTLstartsentencespace
    \DTLbibfield {Title}
    \DTLcheckbibfieldendsperiod {Title}
    \DTLaddperiod
  }
  { }
\DTLifbibfieldexists{Type}
  {
    \DTLstartsentencespace
    \DTLbibfield {Type}
    \DTLcheckbibfieldendsperiod {Type}
    \DTLifanybibfieldexists
      {School,Address,Month,Year}
     { \DTLaddcomma } { \DTLaddperiod }
  }
  { }
\DTLifbibfieldexists {School}
  {
    \DTLstartsentencespace
    \DTLbibfield {School}
    \DTLcheckbibfieldendsperiod {School}
    \DTLifanybibfieldexists
      {Address,Month,Year}
     { \DTLaddcomma } { \DTLaddperiod }
  }
  { }
\DTLifbibfieldexists {Address}
  {
    \DTLstartsentencespace
    \DTLbibfield {Address}
    \DTLcheckbibfieldendsperiod {Address}
    \DTLifanybibfieldexists {Month,Year}
     { \DTLaddcomma } { \DTLaddperiod }
  }
  { }
\DTLformatdate
\DTLifanybibfieldexists {Month,Year}
  { \DTLaddperiod } { }
\DTLifbibfieldexists {Note}
  {
    \DTLstartsentencespace
    \DTLbibfield {Note}
    \DTLcheckbibfieldendsperiod {Note}
    \DTLaddperiod
  }
```

```
    { }
  }
```

The format of a miscellaneous entry.

```
\renewcommand*{\DTLformatmisc}
 {
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Title}
    {
      \DTLstartsentencespace
      \DTLbibfield {Title}
      \DTLcheckbibfieldendsperiod {Title}
      \DTLifbibfieldexists {HowPublished}
       {
         \DTLaddperiod
       }
       {
         \DTLifanybibfieldexists {Month,Year}
           { \DTLaddcomma } { \DTLaddperiod }
       }
      \DTLmidsentencefalse
    }
    { }
   \DTLifbibfieldexists {HowPublished}
    {
      \DTLstartsentencespace
      \DTLbibfield {HowPublished}
      \DTLcheckbibfieldendsperiod {HowPublished}
      \DTLifanybibfieldexists {Month,Year}
       { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLformatdate
   \DTLifanybibfieldexists {Month,Year}
    { \DTLaddperiod } { }
   \DTLifbibfieldexists {Note}
    {
      \DTLstartsentencespace
      \DTLbibfield {Note}
      \DTLcheckbibfieldendsperiod {Note}
      \DTLaddperiod
    }
    { }
 }
```

The format of a PhD thesis.

```
\renewcommand*{\DTLformatphdthesis}
 {
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Title}
    {
      \DTLstartsentencespace
      \DTLthesistitlefmt { \DTLbibfield{Title} }
      \DTLcheckbibfieldendsperiod {Title}
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Type}
    {
      \DTLstartsentencespace
      \DTLbibfield {Type}
      \DTLcheckbibfieldendsperiod {Type}
      \DTLifanybibfieldexists {School,Address,Month,Year}
        { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLifbibfieldexists {School}
    {
      \DTLstartsentencespace
      \DTLbibfield {School}
      \DTLcheckbibfieldendsperiod {School}
      \DTLifanybibfieldexists {Address,Month,Year}
        { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLifbibfieldexists {Address}
    {
      \DTLstartsentencespace
      \DTLbibfield {Address}
      \DTLcheckbibfieldendsperiod {Address}
      \DTLifanybibfieldexists {Month,Year}
        { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLformatdate
   \DTLifanybibfieldexists {Month,Year}
    { \DTLaddperiod } { }
   \DTLifbibfieldexists {Note}
    {
      \DTLstartsentencespace
      \DTLbibfield {Note}
```

```
      \DTLcheckbibfieldendsperiod {Note}
      \DTLaddperiod
    }
    { }
  }
```

The format of a proceedings.

```
\renewcommand*{\DTLformatproceedings}
 {
   \DTLifbibfieldexists {Editor}
    {
      \DTLformateditorlist
      \DTLaddperiod
    }
    {
      \DTLifbibfieldexists {Organization}
       {
         \DTLstartsentencespace
         \DTLbibfield {Organization}
         \DTLcheckbibfieldendsperiod {Organization}
         \DTLaddperiod
       }
       { }
    }
   \DTLifbibfieldexists {Title}
    {
      \DTLstartsentencespace
      \DTLproceedingstitlefmt { \DTLbibfield{Title} }
      \DTLcheckbibfieldendsperiod {Title}
      \DTLifanybibfieldexists
        {
          Volume,Number,Address,Editor,Publisher,
          Month,Year
        }
      { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLformatbvolume
   \DTLifbibfieldexists {Volume}
    {
      \DTLifanybibfieldexists
        {
          Number,Address,Editor,Publisher,
          Month,Year
        }
       { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLformatnumberseries
   \DTLifbibfieldexists {Number}
```

```
        {
          \DTLifanybibfieldexists
           {
             Address,Editor,Publisher,Month,Year
           }
           { \DTLaddcomma } { \DTLaddperiod }
        }
        { }
      \DTLifbibfieldexists {Address}
        {
          \DTLstartsentencespace
          \DTLbibfield {Address}
          \DTLcheckbibfieldendsperiod {Address}
          \DTLifanybibfieldexists {Month,Year}
           { \DTLaddcomma } { \DTLaddperiod }
          \DTLformatdate
          \DTLifanybibfieldexists {Month,Year}
           { \DTLaddperiod } { }
          \DTLifbibfieldexists {Editor}
           {
             \DTLifbibfieldexists {Organization}
              {
                \DTLstartsentencespace
                \DTLbibfield {Organization}
                \DTLcheckbibfieldendsperiod {Organization}
                \DTLifbibfieldexists {Publisher}
                 { \DTLaddcomma } { \DTLaddperiod }
              }
              { }
           }
           { }
          \DTLifbibfieldexists {Publisher}
           {
             \DTLstartsentencespace
             \DTLbibfield {Publisher}
             \DTLcheckbibfieldendsperiod {Publisher}
             \DTLaddperiod
           }
           { }
        }
        {
```

no address

```
          \DTLifbibfieldexists {Editor}
           {
             \DTLifbibfieldexists {Organization}
              {
                \DTLstartsentencespace
                \DTLbibfield {Organization}
                \DTLcheckbibfieldendsperiod {Organization}
```

```
            \DTLifanybibfieldexists {Publisher,Month,Year}
             { \DTLaddcomma } { \DTLaddperiod }
          }
          { }
        }
        { }
       \DTLifbibfieldexists {Publisher}
        {
          \DTLstartsentencespace
          \DTLbibfield {Publisher}
          \DTLcheckbibfieldendsperiod {Publisher}
          \DTLifanybibfieldexists {Month,Year}
           { \DTLaddcomma } { \DTLaddperiod }
        }
        { }
       \DTLformatdate
       \DTLifanybibfieldexists {Month,Year}
        { \DTLaddperiod } { }
     }
    \DTLifbibfieldexists {Note}
     {
       \DTLstartsentencespace
       \DTLbibfield {Note}
       \DTLcheckbibfieldendsperiod {Note}
       \DTLaddperiod
     }
     { }
  }
```

The format of a technical report.

```
\renewcommand*{\DTLformattechreport}
 {
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Title}
    {
      \DTLstartsentencespace
      \DTLbibfield {Title}
      \DTLcheckbibfieldendsperiod {Title}
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Type}
    {
      \DTLstartsentencespace
      \DTLbibfield {Type}
```

```
      \DTLcheckbibfieldendsperiod {Type}
      \DTLifbibfieldexists {Number}
       { \DTLpostnumbername } { }
    }
    { }
  \DTLifbibfieldexists {Number}
    {
      \DTLstartsentencespace
      \DTLbibfield {Number}
      \DTLcheckbibfieldendsperiod {Number}
    }
    { }
  \DTLifanybibfieldexists {Type,Number}
    {
      \DTLifanybibfieldexists
        {Institution,Address,Month,Year}
       { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
  \DTLifbibfieldexists {Institution}
    {
      \DTLstartsentencespace
      \DTLbibfield {Institution}
      \DTLcheckbibfieldendsperiod {Institution}
      \DTLifanybibfieldexists {Address,Month,Year}
       { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
  \DTLifbibfieldexists {Address}
    {
      \DTLstartsentencespace
      \DTLbibfield {Address}
      \DTLcheckbibfieldendsperiod {Address}
      \DTLifanybibfieldexists {Month,Year}
        { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
  \DTLformatdate
  \DTLifanybibfieldexists {Month,Year}
    { \DTLaddperiod } { }
  \DTLifbibfieldexists {Note}
    {
      \DTLstartsentencespace
      \DTLbibfield {Note}
      \DTLcheckbibfieldendsperiod {Note}
      \DTLaddperiod
    }
    { }
}
```

The format of an unpublished work.

```
\renewcommand*{\DTLformatunpublished}
 {
   \DTLifbibfieldexists {Author}
    {
      \DTLformatauthorlist
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Title}
    {
      \DTLstartsentencespace
      \DTLbibfield {Title}
      \DTLcheckbibfieldendsperiod {Title}
      \DTLaddperiod
    }
    { }
   \DTLifbibfieldexists {Note}
    {
      \DTLstartsentencespace
      \DTLbibfield {Note}
      \DTLcheckbibfieldendsperiod {Note}
      \DTLifanybibfieldexists {Month,Year}
       { \DTLaddcomma } { \DTLaddperiod }
    }
    { }
   \DTLformatdate
   \DTLifanybibfieldexists {Month,Year}
    { \DTLaddperiod } { }
 }
```

End of ‘plain’ style.

```
}
```

`\DTLformatbooktitle`

```
\newcommand*{\DTLformatbooktitle}[1]{\emph{#1}}
```

`\dtlbst@abbrv` Define ‘abbrv’ style. This is similar to ‘plain’ except that some of the values are abbreviated

```
\newcommand{\dtlbst@abbrv}{%
```

Base this style on ‘plain’:

```
\dtlbst@plain
```

Sets the author name format.

```
\renewcommand*{\DTLformatauthor}[4]{
 \DTLformatabbrvforenames { ##4 } ~
 \DTLformatvon { ##1 }
 \DTLformatsurname { ##2 }
 \DTLformatjr { ##3 }
}
```

Sets the editor name format.

```
\renewcommand*{\DTLformateditor}[4]{
  \DTLformatabbrvforenames { ##4 } ~
  \DTLformatvon { ##1 }
  \DTLformatsurname { ##2 }
  \DTLformatjr { ##3 }
}
```

Sets the monthname format.

```
\let\DTLmonthname\dtl@abbrvmonthname
```

Sets other predefined names:

```
\DTLresetpredefinedabbrv
```

End of ‘abbrv’ style.

```
}
```

`\dtlbst@alpha` Define ‘alpha’ style. This is similar to ‘plain’ except that the labels are strings rather than numerical.

```
\newcommand{\dtlbst@alpha}{%
```

Base this style on ‘plain’:

```
\dtlbst@plain
```

Set how to format the entire bibliography:

```
\RenewDocumentEnvironment { DTLthebibliography }
 { O{\boolean{true}} m }
{
  \dtl@createalphabiblabels { ##1 } { ##2 }
  \exp_args:NnV \begin { thebibliography }
    \g__databib_widest_label_tl
}
{ \end {thebibliography} }
```

Set how to start the bibliography entry:

```
\renewcommand* \DTLbibitem
 {
   \exp_last_unbraced:NNf
    \bibitem
     [
       { \tl_use:c { dtl@biblabel@ \DBIBcitekey } }
     ]
    { \DBIBcitekey }
 }
\renewcommand* \DTLmbibitem [1]
 {
   \exp_last_unbraced:NNf
    \bibitem
     [
       { \tl_use:c { dtl@biblabel@ \DBIBcitekey } }
     ]
    { ##1 @ \DBIBcitekey }
```

```
    }
```
End of ‘alpha’ style.
```
 }
```

`\@dtl@widestlabel` Version 3.0: replaced `\@dtl@widestlabel` with:
```
\tl_new:N \g__databib_widest_label_tl
```

`\@dtl@thislabel` Version 3.0: replaced `\@dtl@thislabel` with:
```
\tl_new:N \l__databib_label_tl
```

`\dtl@createalphabiblabels{`⟨*condition*⟩`}{`⟨*db name*⟩`}`

`\dtl@createalphabiblabels` Constructs the alpha style bib labels for the given database. (Labels are stored in the control sequence `\dtl@biblabel@`⟨*citekey*⟩.) This also sets the widest label.
```
\newcommand*{\dtl@createalphabiblabels}[2]{
 \dtl@message { Creating ~ bib ~ labels }
 \group_begin:
   \tl_gclear:N \g__databib_widest_label_tl
   \dim_zero:N \l__databib_widest_dim
```
Version 3.0: switched to read-only loop.
```
    \DTLforeachbibentry * [ #1 ] { #2 }
     {
       \dtl@message { \DBIBcitekey }
       \DTLifbibfieldexists {Author}
        {
          \__databib_get_alpha_label:NV
            \l__databib_label_tl \@dtl@key@Author
        }
        {
          \DTLifbibfieldexists {Editor}
           {
             \__databib_get_alpha_label:NV
               \l__databib_label_tl \@dtl@key@Editor
           }
           {
             \DTLifbibfieldexists {Key}
              {
                \__databib_get_first_three:Nn
                  \l__databib_label_tl { \@dtl@key@Key }
              }
              {
                \DTLifbibfieldexists {Organization}
                 {
                   \__databib_get_first_three:Nn
                     \l__databib_label_tl
                     { \@dtl@key@Organization }
```

```
            }
            {
              \__databib_get_first_three:Nn
                \l__databib_label_tl
                { \DBIBentrytype }
            }
         }
      }
   }
  \DTLifbibfieldexists {Year}
   { }
   {
     \DTLifbibfieldexists {CrossRef}
      {
        \DTLgetvalueforkey
         \@dtl@key@Year
         { Year } { #2 } { CiteKey }
         { \@dtl@key@CrossRef }
      }
      { }
   }
  \DTLifbibfieldexists {Year}
   {
     \__databib_get_year_suffix:V \@dtl@key@Year
     \tl_put_right:Ne \l__databib_label_tl
      { \l__databib_year_tl }
   }
   { }
```

Version 3.0: replaced \c@biblabel@⟨*label*⟩ with \g__databib_label_⟨*label*⟩_int, replaced \@dtl@bibfirst@⟨*label*⟩ with \g__databib_first_⟨*label*⟩_tl and replaced \dtl@biblabel@⟨*key*⟩ with \g__databib_key_⟨*key*⟩_tl

```
  \int_if_exist:cTF
    { g__databib_label_ \l__databib_label_tl  _int }
   {
     \int_compare:nNnT
        { \int_use:c { g__databib_label_ \l__databib_label_tl _int } }
          =
        { \c_one_int }
      {
        \tl_gset:ce
          {
            g__databib_key_
            \tl_use:c {  g__databib_first_ \l__databib_label_tl _tl }
             _tl
          }
          { \l__databib_label_tl a }
      }
     \int_gincr:c
      { g__databib_label_ \l__databib_label_tl _int }
```

```
            \tl_gset:ce
              { g__databib_key_ \DBIBcitekey _tl }
              {
                \l__databib_label_tl
                \int_to_alph:n
                  { \int_use:c { g__databib_label_ \l__databib_label_tl _int } }
              }
          }
          {
            \int_new:c { g__databib_label_ \l__databib_label_tl _int }
            \int_gset_eq:cN
               { g__databib_label_ \l__databib_label_tl _int }
              \c_one_int
            \tl_gset_eq:cN
               { g__databib_first_ \l__databib_label_tl _tl }
              \DBIBcitekey
            \tl_gset_eq:cN
               { g__databib_key_ \DBIBcitekey _tl }
              \l__databib_label_tl
          }
        \datatool_measure_width:Nn \dtl@tmplength
         { \tl_use:c { g__databib_key_ \DBIBcitekey _tl } }
        \dim_compare:nNnT
           { \dtl@tmplength } > { \l__databib_widest_dim }
         {
           \dim_set_eq:NN
             \l__databib_widest_dim \dtl@tmplength
           \tl_gset_eq:Nc \g__databib_widest_label_tl
             { g__databib_key_ \DBIBcitekey _tl }
         }
      }
   \group_end:
  }
```

`\dtl@listgetalphalabel` Determine the alpha style label from a list of authors/editors (the first argument must be a control sequence (in which the label is stored), the second argument must be the list of names.) Version 3.0: replaced `\dtl@listgetalphalabel` with:

```
\cs_new:Nn \__databib_get_alpha_label:Nn
{
 \group_begin:
   \tl_clear:N \DTLafterinitials
   \tl_clear:N \DTLbetweeninitials
   \tl_clear:N \DTLafterinitialbeforehyphen
   \tl_clear:N \DTLinitialhyphen
   \int_set:Nn \l__databib_author_count_int
    { \clist_count:n { #2 } }
   \int_compare:nNnTF
     { \l__databib_author_count_int } = { \c_one_int }
    {
      \int_compare:nNnTF
```

```
        { \tl_count:o { #2 } } = { 4 }
       {
         \exp_last_unbraced:No
           \databib_get_solo_author_initials:nnnnN
            #2 #1
       }
       {
         \exp_args:No \DTLstoreinitials { #2 } #1
       }
    }
    {
      \tl_clear:N #1
      \int_zero:N \l__datatool_count_int
      \clist_map_inline:nn { #2 }
       {
        \int_compare:nNnTF
          { \tl_count:n { ##1 } } = { 4 }
         {
           \databib_get_author_initials:nnnnN
             ##1 \l__databib_author_initials_tl
         }
         {
           \DTLstoreinitials
             { ##1 } \l__databib_author_initials_tl
         }
        \tl_put_right:NV #1  \l__databib_author_initials_tl
        \int_incr:N \l__datatool_count_int
        \int_compare:nNnT
          { \l__datatool_count_int } > { 2 }
         {
           \clist_map_break:
         }
       }
    }
  \exp_args:NNNV
    \group_end:
    \tl_set:Nn #1 #1
}
\cs_generate_variant:Nn \__databib_get_alpha_label:Nn { NV }
```

`\dtl@getauthorinitial` Get author's initial:

```
\tl_new:N \l__databib_author_initials_tl
\cs_new:Nn \databib_get_author_initials:nnnnN
{
 \tl_if_empty:nTF { #1 }
  {
    \DTLstoreinitials { #2 } #5
  }
  {
    \DTLstoreinitials { #1 ~ #2 } #5
```

```
  }
}
```

Get label for single author (last argument is control sequence in which to store the label):

```
\cs_new:Nn \databib_get_solo_author_initials:nnnnN
{
  \tl_if_empty:nTF { #1 }
   {
     \DTLstoreinitials { #2 } #5
   }
   {
     \DTLstoreinitials { #1 ~ #2 } #5
   }
  \int_compare:nNnT
     { \tl_count:N #5 } < { 2 }
   {
     \__databib_get_first_three:Nn #5 { #1 #2 }
   }
}
```

`\dtl@get@firstthree` Get first three letters from the given string. Version 3.0: replaced `\dtl@get@firstthree` with:

```
\cs_new:Nn \__databib_get_first_three:Nn
{
  \tl_clear:N #1
  \int_zero:N \l__datatool_count_int
  \exp_args:Nx \text_map_inline:nn { \text_purify:n { #2 } }
   {
     \tl_put_right:Nn #1 { ##1 }
     \int_incr:N \l__datatool_count_int
     \int_compare:nNnT
       { \l__datatool_count_int } = { 3 }
      {
        \text_map_break:
      }
   }
}
```

`\dtl@get@yearsuffix` Get year suffix. Version 3.0: replaced `\dtl@get@yearsuffix` with:

```
\cs_new:Nn \__databib_get_year_suffix:n
{
  \regex_extract_once:NnNTF
     \c_databib_year_suffix_regex
     { #1 }
     \l__datatool_tmp_seq
   {
     \seq_get_left:NN
       \l__datatool_tmp_seq
       \l__databib_year_tl
```

```
    }
    {
      \tl_clear:N \l__databib_year_tl
    }
  }
  \cs_generate_variant:Nn  \__databib_get_year_suffix:n { V }
  \regex_const:Nn \c_databib_year_suffix_regex
   { ( \d \d ) \Z }
```

`\@dtl@year` Version 3.0: replaced `\@dtl@year` with:

```
\tl_new:N \l__databib_year_tl
```

`\DTLbibliographystyle{`⟨*style*⟩`}`

`\DTLbibliographystyle` Sets the bibliography style.

```
\NewDocumentCommand \DTLbibliographystyle { m }
{
  \tl_if_exist:cTF { dtlbst@ #1 }
   {
     \tl_use:c { dtlbst@ #1 }
   }
   {
     \PackageError {databib}
      {
        Unknown ~ bibliography ~ style ~ `#1'
      }
      { }
   }
}
%
```

Set the default bibliography style:

```
\DTLbibliographystyle { \l__databib_style_tl }
```

25.8 Multiple Bibliographies

In order to have multiple bibliographies, there needs to be an aux file for each bibliography. The main bibliography is in `\jobname`.aux, but need to provide a means of creating additional aux files.

`\DTLmultibibs{`⟨*list*⟩`}`

`\DTLmultibibs` This creates an auxiliary file for each name in ⟨*list*⟩. For example, `\DTLmultibibs{foo,bar}` will create the files `foo.aux` and `bar.aux`.

```
\NewDocumentCommand \DTLmultibibs { m }
{
```

```
    \clist_map_inline:nn { #1 }
     {
       \__databib_auto_build:n { ##1 }
       \iow_new:c { g__databib_aux_ ##1 _iow }
       \iow_open:cn { g__databib_aux_ ##1 _iow } { ##1 .aux }
       \tl_new:c { b@ ##1 @ * }
     }
  }
```

Can only be used in the preamble:

```
\@onlypreamble { \DTLmultibibs }
```

`\@dtl@cite@write`

```
\newcommand{\@dtl@cite@write}[2]{%
 \if@filesw
   \tl_if_exist:cTF { g__databib_aux_ #1 _iow }
    {
      \iow_now:ce { g__databib_aux_ #1 _iow } { #2 }
    }
    {
      \PackageError {databib}
       { multibib ~ `#1' ~ not ~ defined }
       {
         You ~ need ~ to ~ define ~ `#1' ~ in ~
         \token_to_str:N \DTLmultibibs
       }
    }
 \fi
}

\ExplSyntaxOff
```

`\DTLcite`

`\DTLcite[`⟨*text*⟩`]{`⟨*mbib*⟩`}{`⟨*labels*⟩`}`

This is similar to `\cite[`⟨*text*⟩`]{`⟨*labels*⟩`}`, except 1) the cite information is written to the auxiliary file associated with the multi-bib ⟨*mbib*⟩ (which must be named in `\DTLmultibibs`) and 2) the cross referencing label is constructed from ⟨*mbib*⟩ and ⟨*label*⟩ to allow for the same citation to appear in multiple bibliographies.

```
\newcommand*{\DTLcite}{\@ifnextchar[{\@tempswatrue \dtl@citex
}{\@tempswafalse \dtl@citex[]}}
```

`\dtl@citex` Adapted from `\@citex`

```
\def\dtl@citex[#1]#2#3{%
\leavevmode\let\@citea\@empty
\@cite{\@for\@citeb:=#3\do{\@citea
  \def\@citea{,\penalty \@m \ }%
  \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
  \@dtl@cite@write{#2}{\string\citation{\@citeb}}%
```

```
    \@ifundefined{b@#2@\@citeb}{%
      \hbox{\reset@font\bfseries ?}%
      \G@refundefinedtrue
      \PackageWarning{databib}{Citation `\@citeb ' for multibib `#2'
        undefined}%
    }{%
      \@cite@ofmt{\csname b@#2@\@citeb \endcsname }%
    }%
    }}{#1}%
  }
```

\DTLnocite

\DTLnocite{⟨*mbib*⟩}{⟨*key list*⟩}

As \nocite but uses the aux file associated with ⟨*mbib*⟩ which must have been defined using \DTLmultibibs.

```
\newcommand*{\DTLnocite}[2]{%
  \@bsphack
  \ifx\@onlypreamble\document
    \@for\@citeb:=#2\do{%
      \edef\@citeb{\expandafter\@firstofone\@citeb}%
      \@dtl@cite@write{#1}{\string\citation{\@citeb}}%
      \@ifundefined{b@#1@\@citeb}{%
        \G@refundefinedtrue
        \PackageWarning{databib}{Citation `\@citeb ' undefined for
         multibib `#1'}}{}%
    }%
  \else
    \@latex@error{Cannot be used in preamble}\@eha
  \fi
  \@esphack
}

\ExplSyntaxOn
```

\DTLloadmbbl

\DTLloadmbib{⟨*mbib*⟩}{⟨*db name*⟩}{⟨*bib list*⟩}

```
\NewDocumentCommand \DTLloadmbbl { m m m }
{
 \@dtl@cite@write { #1 }
  {
    \token_to_str:N \bibstyle { databib }
    \iow_newline:
    \token_to_str:N \bibdata { #3 }
  }
  \DTLnewdb { #2 }
```

```
  \tl_set:Nx \DTLBIBdbname { \exp_args:Ne \tl_to_str:n { #2 } }
  \@input@ { #1 .bbl }
}
```

\DTLmbibliography

\DTLmbibliography[⟨*condition*⟩]{⟨*mbib name*⟩}{⟨*bib dbname*⟩}

Displays the bibliography for the database ⟨*bib dbname*⟩ which must have previously been loaded using `\DTLloadmbbl`, where ⟨*mbib name*⟩ must be listed in `\DTLmultibibs`.

```
\NewDocumentCommand \DTLmbibliography
  { O{\boolean{true}} m m }
{
  \begin {DTLthebibliography} [ #1 ] { #3 }
   \DTLforeachbibentry * [ #1 ] { #3 }
    {
      \DTLmbibitem { #2 }
      \DTLformatbibentry
      \DTLendbibitem
    }
  \end {DTLthebibliography}
}
```

25.9 Sort Support

The author and editor columns will be comma-separated lists with each item in the form {⟨*von*⟩}{⟨*surname*⟩}{⟨*jr*⟩}{⟨*forenames*⟩}. This means that if the database is sorted by author or editor, the names will be munged together. If it's not sufficient to sort by this, a command is provided for use with the encap sort option.

\DTLbibsortencap

\DTLbibsortencap{⟨*value*⟩}{⟨*col-idx*⟩}{⟨*db-name*⟩}

```
\newcommand \DTLbibsortencap [3]
{
   \bool_lazy_or:nnTF
     { \int_compare_p:nNn { #2 } = { \dtlcolumnindex { #3 } { Author } } }
     { \int_compare_p:nNn { #2 } = { \dtlcolumnindex { #3 } { Editor } } }
   {
     \exp_not:N \DTLbibsortname
     \clist_use:nn { #1 } { \DTLbibsortnamesep \DTLbibsortname }
   }
   { \exp_not:n { #1 } }
}
```

```
\DTLbibsortname
                    \newcommand{\DTLbibsortname}[4]{
                     \tl_if_empty:nF { #1 } { #1 ~ }
                     #2
                     \tl_if_empty:nF { #3 } { , ~ #3 }
                     \datatoolpersoncomma
                     #4
                    }

\DTLbibsortnamesep
                    \newcommand{\DTLbibsortnamesep}{ \datatoolasciiend }

                    \ExplSyntaxOff
```

25.10 Localisation Support

Version 3.0: added localisation support.

```
\RequireDataBibDialect
                    \newcommand*{\RequireDataBibDialect}[1]{%
                     \TrackLangRequireDialect{databib}{#1}%
                    }

                    \datatool@load@locales{%
                      \AnyTrackedLanguages
                      {%
                        \ForEachTrackedDialect{\@dtl@thisdialect}%
                        {%
                          \RequireDataBibDialect{\@dtl@thisdialect}%
                        }%
                      }%
                      {}%
                    }
```

26 databar.sty

Declare package:

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{databar-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{databar}[2026/07/31 v3.4.4 (NLCT)]
```

Version 3.0: no longer using xkeyval.

`\ifDTLcolorbarchart` The conditional `\ifDTLcolorbarchart` is used to determine whether to use colour or grey scale. NB may be removed or deprecated.

```
\newif\ifDTLcolorbarchart
\DTLcolorbarcharttrue
```

`\ifDTLverticalbars` Define boolean keys to govern bar chart orientation.

```
\newif\ifDTLverticalbars
```

Set defaults:

```
\DTLverticalbarstrue
```

`\DTLbarXlabelalign` Alignment for lower bar labels.

```
\newcommand\DTLbarXlabelalign{%
 \ifDTLverticalbars left,rotate=-90\else right\fi
}
```

`\DTLbarXupperlabelalign` Alignment for upper bar labels.

```
\newcommand\DTLbarXupperlabelalign{%
 \ifDTLverticalbars bottom,center\else left\fi
}
```

`\DTLbarXneglabelalign` Alignment for lower label on negative bars.

```
\newcommand\DTLbarXneglabelalign{%
 \ifDTLverticalbars right,rotate=-90\else left\fi
}
```

`TLbarXnegupperlabelalign` Alignment for upper bar labels on negative bars.

```
\newcommand\DTLbarXnegupperlabelalign{%
 \ifDTLverticalbars top,center\else right\fi
}
```

`\DTLbarYticklabelalign` Alignment for y tick labels.

```
\newcommand\DTLbarYticklabelalign{%
 \ifDTLverticalbars right\else top,center\fi
}
```

The package options have been changed to use l3keys.

Define package options:

```
\ExplSyntaxOn
\keys_define:nn { datatool }
{
  color .legacy_if_set:n = DTLcolorbarchart,
  gray .legacy_if_set_inverse:n = DTLcolorbarchart,
  verticalbars .legacy_if_set:n = DTLverticalbars ,
  vertical .code:n =
   {
     \DTLverticalbarstrue
   },
```

```
  horizontal .code:n =
   {
     \DTLverticalbarsfalse
   },
}
\ExplSyntaxOff
```

Process options:

```
\IfPackageLoadedTF{datatool}
{
  \ProcessKeyOptions[datatool]
}
{
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datatool}}
  \ProcessOptions
}
\RequirePackage{datatool}
```

Remove the package option keys so they can't be used with `\DTLsetup` (otherwise they may conflict with datapie etc).

```
\ExplSyntaxOn
\keys_define:nn { datatool }
{
  color .undefine: ,
  gray .undefine: ,
  verticalbars .undefine: ,
  vertical .undefine: ,
  horizontal .undefine: ,
}
\ExplSyntaxOff
```

Require dataplot package. Note that this also loads tikz.

```
\RequirePackage{dataplot}
```

26.1 Scratch Variables

```
\ExplSyntaxOn
```

Token list to construct content:

```
\tl_new:N \l__databar_content_tl
```

`\DTLtotalbars` Used to store the total number of bars.

```
\tl_new:N \DTLtotalbars
```

`\DTLbartotalvariables` Used to store the total number of variables for multi charts.

```
\tl_new:N \DTLbartotalvariables
```

`\DTLtotalbargroups` Used to store the total number of bar groups for multi charts.

```
\tl_new:N \DTLtotalbargroups
```

\DTLbarchartwidth Used to store the total width of the x-axis. This doesn't include the y tick marks or labels.

```
\tl_new:N \DTLbarchartwidth
```

```
\dim_new:N \l__databar_miny_dim
\dim_new:N \l__databar_maxy_dim
```

\DTLbarvariable The bar chart variable.

```
\tl_new:N \DTLbarvariable
```

\dtlbar@variables Token list variables used to create the multi bar chart are now stored in a sequence. Version 3.0: changed \dtlbar@variables to:

```
\seq_new:N \l__databar_variables_seq
```

\@dtl@start Lower bar labels. Version 3.0: renamed \@dtl@start to:

```
\fp_new:N \l__databar_start_fp
```

Consider a bar as a thick line (of one bar width). This is anchored to the x axis at the line's start \DTLstartpt and extends to the line's end \DTLendpt. Half-way along this line is the mid point \DTLmidpt. The x-coordinate for all these points is the same and will be stored in:

```
\tl_new:N \l__databar_mid_tl
\dim_new:N \l__databar_mid_dim
```

The extent is the current bar's y value converted to a dimension:

```
\dim_new:N \l__databar_extent_dim
```

The y point of the starting point will be 0. The y point of the ending point will be the variable value. But we also need the starting point of the rectangle and the end point in order to draw the bar so that the line passes through the middle. The x co-ordinate of the start of the rectangle is stored in:

```
\tl_new:N \l__databar_start_tl
```

The x co-ordinate of the end of the rectangle is stored in:

```
\tl_new:N \l__databar_end_tl
```

Four points of the rectangle are obtained with those x co-ordinates for $y = 0$ and $y = v$, where v is the bar variable's value.

Offset for group label.

```
\dim_new:N \l__databar_group_label_offset_dim
```

\dtl@extent Bar extent. Version 3.0: renamed \dtl@extent to:

```
\fp_new:N \l__databar_extent_fp
```

\dtl@unit Lower bar labels. Version 3.0: renamed \dtl@unit to:

```
\fp_new:N \l__databar_unit_fp
```

Current label offset and alignment:

```
\tl_new:N \l__databar_label_current_offset_tl
\tl_new:N \l__databar_label_current_align_tl
```

\dtl@barlabel Lower bar labels. Version 3.0: renamed \dtl@barlabel to:

```
\tl_new:N \l__databar_barlabel_tl
```

Lower bar label in multi-bar chart.

```
\tl_new:N \l__databar_lowerbarlabel_tl
```

\dtl@multibarlabels Lower bar labels for multi-bar charts. Version 3.0: changed \dtl@multibarlabels to:

```
\seq_new:N \l__databar_multibarlabels_seq
```

\dtlbar@groupgap Gap between groups in multi-bar charts (This should be in x units where 1 x unit is the width of a bar.) Version 3.0: changed \dtlbar@groupgap to:

```
\tl_new:N \l__databar_groupgap_tl
\tl_set:Nn \l__databar_groupgap_tl { 1 }
```

Gap between bars.

```
\tl_new:N \l__databar_bargap_tl
\tl_set:Nn \l__databar_bargap_tl { 0 }
```

\dtl@upperbarlabel Upper bar labels. Version 3.0: changed \dtl@upperbarlabel to:

```
\tl_new:N \l__databar_upperbarlabel_tl
```

\dtl@uppermultibarlabels Upper bar labels for multi-bar charts. Version 3.0: changed \dtl@uppermultibarlabels to:

```
\seq_new:N \l__databar_uppermultibarlabels_seq
```

dtlbar@yticlist Define list of points for y tics. (Must be a comma separated list of decimal numbers.) Version 3.0: changed from \dtlbar@yticlist to a sequence.

```
\clist_new:N \l__databar_yticpoints_clist
\seq_new:N \l__databar_yticpoints_seq
```

\dtlbar@yticgap The y tick gap. Version 3.0: changed \dtlbar@yticgap to:

```
\tl_new:N \l__databar_yticgap_tl
\fp_new:N \l__databar_yticgap_fp
```

dtlbar@yticlabels Define list of points for y tick labels. Version 3.0: changed from \dtlbar@yticlabels to a sequence.

```
\seq_new:N \l__databar_yticlabels_seq
```

\dtlbar@ylabel y axis label. Version 3.0: changed \dtlbar@ylabel to:

```
\tl_new:N \l__databar_ylabel_tl
```

\DTLyAxisLabelStyle y-axis label style.

```
\newcommand{\DTLyAxisLabelStyle}{%
 bottom, center \ifDTLverticalbars , rotate = 90 \fi
}
```

```
\tl_new:N \l__databar_ylabel_offset_tl
\tl_set:Nn \l__databar_ylabel_offset_tl
 { \baselineskip }
\tl_new:N \l__databar_ylabel_pos_tl
\tl_set:Nn \l__databar_ylabel_pos_tl
 { \c__databar_ylabel_pos_centred_tl }
```

Centred along the y axis:

```
\tl_const:Nn \c__databar_ylabel_pos_centred_tl
 {
   \ifDTLverticalbars
     \exp_not:N \pgfpoint
      {
        - \exp_not:N \l__dataplot_y_tic_label_width_dim
      }
      {
        \dim_eval:n
         {
            0.5 \DTLbarchartlength + \l__databar_miny_dim
         }
      }
   \else
     \exp_not:N \pgfpoint
      {
        \dim_eval:n
         {
            0.5 \DTLbarchartlength + \l__databar_miny_dim
         }
      }
      {
        - \exp_not:N \l__dataplot_y_tic_label_width_dim
      }
   \fi
 }
```

At the minimum end of the y axis:

```
\tl_const:Nn \c__databar_ylabel_pos_min_tl
 {
   \ifDTLverticalbars
     \exp_not:N \pgfpoint
      {
        - \exp_not:N \l__dataplot_y_tic_label_width_dim
      }
      { \exp_not:N \l__databar_miny_dim }
   \else
     \exp_not:N \pgfpoint
      { \exp_not:N \l__databar_miny_dim }
      {
        - \exp_not:N \l__dataplot_y_tic_label_width_dim
      }
```

```
    \fi
  }
```

At the maximum end:

```
\tl_const:Nn \c__databar_ylabel_pos_max_tl
 {
   \ifDTLverticalbars
     \exp_not:N \pgfpoint
      {
        - \exp_not:N \l__dataplot_y_tic_label_width_dim
      }
      { \exp_not:N \l__databar_maxy_dim }
   \else
     \exp_not:N \pgfpoint
      { \exp_not:N \l__databar_maxy_dim }
      {
        - \exp_not:N \l__dataplot_y_tic_label_width_dim
      }
   \fi
 }
```

At the origin:

```
\tl_const:Nn \c__databar_ylabel_pos_zero_tl
 {
   \ifDTLverticalbars
     \exp_not:N \pgfpoint
      {
        - \exp_not:N \l__dataplot_y_tic_label_width_dim
      }
      { \exp_not:N \c_zero_dim }
   \else
     \exp_not:N \pgfpoint
      { \exp_not:N \c_zero_dim }
      {
        - \exp_not:N \l__dataplot_y_tic_label_width_dim
      }
   \fi
 }
```

`\@dtl@barcount` Version 3.0: integer register `\@dtl@barcount` removed.

`\DTLbarindex` May be used in hooks to access the bar index.

```
\newcommand{\DTLbarindex}{0}
```

`\DTLbargroupindex` May be used in hooks to access the bar group index.

```
\newcommand{\DTLbargroupindex}{0}
```

26.2 Settings

Define some variables that govern the appearance of the bar chart.

\DTLbargrouplabelalign Alignment for group label.

```
\newcommand\DTLbargrouplabelalign{\DTLbarXlabelalign}
```

\DTLbarchartlength The total height of the bar chart is given by \DTLbarchartheight

```
\newlength\DTLbarchartlength
\DTLbarchartlength=3in
```

\DTLbarwidth The width of each bar is given by \DTLbarwidth.

```
\newlength\DTLbarwidth
\DTLbarwidth=1cm
```

\DTLbarmax The maximum value of the y axis.

```
\tl_new:N \DTLbarmax
```

Set the maximum, with a check to make sure that the value isn't negative:

```
\cs_new:Nn \__databar_set_max:n
 {
   \tl_if_empty:nTF { #1 }
    {
     \tl_clear:N \DTLbarmax
    }
    {
     \fp_compare:nNnTF
      { #1 } < { \c_zero_fp }
     {
       \PackageError { databar }
        {
          max ~ must ~ be ~ zero ~ or ~ positive ~ (or ~
          empty ~ for ~ the ~ default)
        }
        { }
     }
     {
       \tl_set:Nn \DTLbarmax { #1 }
     }
   }
 }
```

\DTLnegextent The negative extent.

```
\tl_new:N \DTLnegextent
```

Set the negative extent, with a check to make sure that the value isn't positive:

```
\cs_new:Nn \__databar_set_max_depth:n
 {
   \tl_if_empty:nTF { #1 }
    {
     \tl_clear:N \DTLnegextent
    }
    {
```

```
      \fp_compare:nNnTF
       { #1 } > { \c_zero_fp }
      {
        \PackageError { databar }
         {
           maxdepth ~ must ~ be ~ zero ~ or ~ negative ~ (or ~
           empty ~ for ~ the ~ default)
         }
         { }
      }
      {
        \tl_set:Nn \DTLnegextent { #1 }
      }
    }
  }
```

`\DTLbarlabeloffset` The offset from the x axis to the bar label is given by `\DTLbarlabeloffset`.

```
\newlength\DTLbarlabeloffset
\setlength\DTLbarlabeloffset{10pt}
```

The offset from the upper edge of the bar to the upper label.

```
\tl_new:N \l__databar_upper_label_offset_tl
\tl_set:Nn \l__databar_upper_label_offset_tl
 { \DTLbarlabeloffset }
```

Label offsets depend on the label-style setting. Lower labels:

```
\tl_new:N \l__databar_label_pos_offset_tl
\tl_set:Nn \l__databar_label_pos_offset_tl
 { \dim_to_decimal:n { -\DTLbarlabeloffset } pt }
\tl_new:N \l__databar_label_neg_offset_tl
\tl_set:Nn \l__databar_label_neg_offset_tl
 { \dim_to_decimal:n { \DTLbarlabeloffset } pt }
```

Upper labels:

```
\tl_new:N \l__databar_upper_label_pos_offset_tl
\tl_set:Nn \l__databar_upper_label_pos_offset_tl
 { \dim_to_decimal:n { \l__databar_upper_label_offset_tl } pt }
\tl_new:N \l__databar_upper_label_neg_offset_tl
\tl_set:Nn \l__databar_upper_label_neg_offset_tl
 { \dim_to_decimal:n { -\l__databar_upper_label_offset_tl } pt }
```

`\DTLBarXAxisStyle` The style of the x axis is given by `\DTLBarXAxisStyle`

```
\newcommand*{\DTLBarXAxisStyle}{-}
```

`\DTLBarYAxisStyle` The style of the y axis is given by `\DTLBarYAxisStyle`.

```
\newcommand*{\DTLBarYAxisStyle}{-}
```

`\DTLBarStyle` Any additional styling for each bar.

```
\newcommand*{\DTLBarStyle}{}
```

DTLbarroundvar DTLbarroundvar is a counter governing the number of digits to round to for the label.

```
\newcounter{DTLbarroundvar}
\setcounter{DTLbarroundvar}{1}
```

Number of digits to round the y tick label:

```
\tl_new:N \l__databar_ytic_round_tl
\tl_set:Nn \l__databar_ytic_round_tl { \int_use:N \c@DTLbarroundvar }
```

\DTLbardisplayYticklabel `\DTLbardisplayYticklabel` governs how the y tick labels appear.

```
\newcommand*{\DTLbardisplayYticklabel}[1]{#1}
```

\DTLdisplaylowerbarlabel `\DTLdisplaylowerbarlabel` governs how the lower bar labels appear.

```
\newcommand*{\DTLdisplaylowerbarlabel}[1]{#1}
```

\DTLdisplaybargrouplabel `\DTLdisplaybargrouplabel` governs how the bar group labels appear.

```
\newcommand*{\DTLdisplaybargrouplabel}[1]{#1}
```

isplaylowermultibarlabel `\DTLdisplaylowermultibarlabel` governs how the lower multi bar labels appear.

```
\newcommand*{\DTLdisplaylowermultibarlabel}[1]{#1}
```

\DTLdisplayupperbarlabel `\DTLdisplayupperbarlabel` governs how the upper bar labels appear.

```
\newcommand*{\DTLdisplayupperbarlabel}[1]{#1}
```

isplayuppermultibarlabel `\DTLdisplayuppermultibarlabel` governs how the upper multi bar labels appear.

```
\newcommand*{\DTLdisplayuppermultibarlabel}[1]{#1}
```

\DTLbaratbegintikz `\DTLbaratbegintikz` specifies any commands to apply at the start of the tikzpicture environment. By default it does nothing.

```
\newcommand*{\DTLbaratbegintikz}{}
```

\DTLbaratendtikz `\DTLbaratendtikz` specifies any commands to apply at the end of the tikzpicture environment. By default it does nothing.

```
\newcommand*{\DTLbaratendtikz}{}
```

\ifDTLbarxaxis The conditional `\ifDTLbarxaxis` is used to determine whether or not to display the x axis

```
\newif\ifDTLbarxaxis
```

\ifDTLbaryaxis The conditional `\ifDTLbaryaxis` is used to determine whether or not to display the y axis.

```
\newif\ifDTLbaryaxis
```

\ifDTLbarytics The conditional `\ifDTLbarytics` to determine whether or not to display the y tick marks.

```
\newif\ifDTLbarytics
```

No-op command:

```
\cs_new:Npn \__databar_noop:N #1
{
  \PackageError { databar }
  {
    Can't ~ use ~ \token_to_str:N #1 \c_space_tl ~ outside ~
    \token_to_str:N \DTLbarchart \c_space_tl ~ or ~
    \token_to_str:N \DTLmultibarchart
  }
  { }
}
```

There are two colour lists: the default colour list and the negative bar list. If a bar value is negative, the colour will first be looked up in the negative bar list. If missing, the default colour list will be used. Bars with positive values will just use the default colour list.

```
\int_new:N \l__databar_max_colors_int
\int_new:N \l__databar_max_neg_colors_int
```

\DTLsetbarcolor

\DTLsetbarcolor{⟨*n*⟩}{⟨*color*⟩}

Assigns colour name ⟨*color*⟩ to the ⟨*n*⟩th bar.

```
\NewDocumentCommand \DTLsetbarcolor { m m }
{
 \tl_set:cn { dtlbar@col\romannumeral#1 } { #2 }
 \int_compare:nNnT { #1 } > { \l__databar_max_colors_int }
  {
   \int_set:Nn \l__databar_max_colors_int { #1 }
  }
}
```

\DTLclearbarcolors

\DTLclearbarcolors

Clears all bar colours.

```
\NewDocumentCommand \DTLclearbarcolors { }
{
  \int_step_inline:nn { \l__databar_max_colors_int }
   {
     \cs_undefine:c { dtlbar@col\romannumeral##1 }
   }
  \int_zero:N \l__databar_max_colors_int
}
```

\DTLsetnegbarcolor

\DTLsetnegbarcolor{⟨*n*⟩}{⟨*color*⟩}

Assigns colour name ⟨*color*⟩ to the ⟨*n*⟩th bar for negative bar values.

```
\NewDocumentCommand \DTLsetnegbarcolor { m m }
{
 \tl_set:cn { dtlbar@negcol\romannumeral#1 } { #2 }
 \int_compare:nNnT { #1 } > { \l__databar_max_neg_colors_int }
  {
   \int_set:Nn \l__databar_max_neg_colors_int { #1 }
  }
}
```

\DTLclearnegbarcolors

\DTLclearnegbarcolors

Clears all negative bar colours.

```
\NewDocumentCommand \DTLclearnegbarcolors { }
{
  \int_step_inline:nn { \l__databar_max_neg_colors_int }
   {
     \cs_undefine:c { dtlbar@negcol\romannumeral##1 }
   }
   \int_zero:N \l__databar_max_neg_colors_int
}
```

Provide an option to cycle round the bar colours. This is a token list variable rather than an integer to allow it to be set to `\l__databar_max_colors_int` which can then take into account any extra colours added afterwards.

```
\tl_new:N \l__databar_color_max_index_tl
\tl_set:Nn \l__databar_color_max_index_tl { \c_zero_int }
\tl_new:N \l__databar_neg_color_max_index_tl
\tl_set:Nn \l__databar_neg_color_max_index_tl
  { \l__databar_color_max_index_tl }
```

```
\cs_new:Nn \__databar_cycle_index:nn
{
  \int_eval:n
   {
     \__databar_cycle_index_adjust:on
      {
        \int_mod:nn { #1 } { \l__databar_color_max_index_tl }
      }
      {  \l__databar_color_max_index_tl }
   }
}
\cs_new:Nn \__databar_cycle_index_adjust:nn
{
  \int_if_zero:nTF { #1 } { #2 } { #1 }
}
\cs_generate_variant:Nn \__databar_cycle_index_adjust:nn { on }
```

\DTLgetbarcolor

\DTLgetbarcolor{⟨*n*⟩}

Gets the colour specification for the ⟨*n*⟩th bar.

```
\newcommand*{\DTLgetbarcolor}[1]{%
 \int_compare:nNnTF
   { \l__databar_color_max_index_tl } > { \c_zero_int }
  {
    \__databar_get_bar_col:n
     {
       \__databar_cycle_index:nn
        { #1 } { \l__databar_color_max_index_tl }
     }
  }
  {
    \__databar_get_bar_col:n { #1 }
  }
}
\cs_new:Nn \__databar_get_bar_col:n
{
  \cs_if_exist_use:cF { dtlbar@col\romannumeral#1 } { white }
}
```

\DTLgetnegbarcolor

\DTLgetnegbarcolor{⟨*n*⟩}

Gets the negative bar colour specification for the ⟨*n*⟩th bar.

```
\newcommand*{\DTLgetnegbarcolor}[1]{%
 \int_compare:nNnTF
   { \l__databar_neg_color_max_index_tl } > { \c_zero_int }
  {
    \__databar_get_neg_bar_col:n
     {
       \__databar_cycle_index:nn
        { #1 } { \l__databar_neg_color_max_index_tl }
     }
  }
  {
    \__databar_get_neg_bar_col:n { #1 }
  }
}
\cs_new:Nn \__databar_get_neg_bar_col:n
{
  \cs_if_exist_use:cF { dtlbar@negcol\romannumeral#1 }
    { \__databar_get_bar_col:n { #1 } }
}
```

\DTLdobarcolor

\DTLdobarcolor[⟨value⟩]{⟨n⟩}

Sets the colour to that for the ⟨n⟩th bar. The value determines whether to use the default colour set or the negative set.

```
\NewDocumentCommand \DTLdobarcolor { o m }
{
  \IfValueTF { #1 }
   {
     \fp_compare:nNnTF { #1 } < { \c_zero_fp }
      {
        \__databar_do_col_or_nothing:x { \DTLgetnegbarcolor { #2 } }
      }
      {
        \__databar_do_col_or_nothing:x { \DTLgetbarcolor { #2 } }
      }
   }
   {
     \__databar_do_col_or_nothing:x { \DTLgetbarcolor { #2 } }
   }
}
\cs_new:Nn \__databar_do_col_or_nothing:n
{
  \tl_if_empty:nF { #1 } { \color { #1 } }
}
\cs_generate_variant:Nn \__databar_do_col_or_nothing:n { x }
```

\DTLdocurrentbarcolor

\DTLdocurrentbarcolor sets the colour to that of the current bar. NB this only consults the default colour list unless \DTLbarvalue is defined and __databar_index_int is greater than 0. In which case, the bar value will be tested to determine if the negative bar colour list should be used.

```
\NewDocumentCommand \DTLdocurrentbarcolor { }
{
 \int_compare:nNnTF
   { \int_use:N \l__databar_index_int } > { \c_zero_int }
  {
    \tl_if_exist:N \DTLbarvalue
     {
       \exp_args:NV \tl_if_head_eq_meaning:nNTF
         \DTLbarvalue \__datatool_datum:nnnn
        {
          \fp_compare:nNnTF
             { \DTLdatumvalue \DTLbarvalue } < { \c_zero_int }
           {
             \__databar_do_col_or_nothing:x
               { \DTLgetnegbarcolor { \l__databar_index_int } }
           }
           {
             \__databar_do_col_or_nothing:x
```

```
                    { \DTLgetbarcolor { \l__databar_index_int } }
                }
              }
              {
                \fp_compare:nNnTF
                  { \DTLbarvalue } < { \c_zero_int }
                {
                  \__databar_do_col_or_nothing:x
                    { \DTLgetnegbarcolor { \l__databar_index_int } }
                }
                {
                  \__databar_do_col_or_nothing:x
                    { \DTLgetbarcolor { \l__databar_index_int } }
                }
              }
            }
            {
              \__databar_do_col_or_nothing:x
               { \DTLgetbarcolor { \l__databar_index_int } }
            }
          }
          {
            \int_compare:nNnTF
              { \l__datatool_row_idx_int } > { \c_zero_int }
            {
              \exp_args:NV \DTLdobarcolor \l__datatool_row_idx_int
            }
            {
              \int_compare:nNnTF { \dtlforeachlevel } > { 0 }
               {
                 \exp_args:Nv \DTLdobarcolor
                    { c@DTLrow\romannumeral\dtlforeachlevel }
               }
               {
                 \__databar_noop:N \DTLdocurrentbarcolor
               }
            }
          }
        }
```

\DTLbaroutlinecolor `\DTLbaroutlinecolor` specifies what colour to draw the outline.

```
\newcommand*{\DTLbaroutlinecolor}{black}
```

\DTLbaroutlinewidth `\DTLbaroutlinewidth` specifies the line width of the outline: Outline is only drawn if the linewidth is greater than 0pt.

```
\newlength\DTLbaroutlinewidth
\DTLbaroutlinewidth=0pt
```

Set the default colours. If there are more than eight bars, more colours will need to be defined.

```
\ifDTLcolorbarchart
 \DTLsetbarcolor{1}{red}
 \DTLsetbarcolor{2}{green}
 \DTLsetbarcolor{3}{blue}
 \DTLsetbarcolor{4}{yellow}
 \DTLsetbarcolor{5}{magenta}
 \DTLsetbarcolor{6}{cyan}
 \DTLsetbarcolor{7}{orange}
 \DTLsetbarcolor{8}{white}
\else
 \DTLsetbarcolor{1}{black!15}
 \DTLsetbarcolor{2}{black!25}
 \DTLsetbarcolor{3}{black!35}
 \DTLsetbarcolor{4}{black!45}
 \DTLsetbarcolor{5}{black!55}
 \DTLsetbarcolor{6}{black!65}
 \DTLsetbarcolor{7}{black!75}
 \DTLsetbarcolor{8}{black!85}
\fi
```

`\DTLeverybarhook` Code to apply at every bar after the bar has been drawn.

```
\newcommand*{\DTLeverybarhook}{}
```

`\DTLeveryprebarhook` Code to apply at every bar before the bar has been drawn.

```
\newcommand*{\DTLeveryprebarhook}{}
```

`\DTLeverybargrouphook` Hook at the end of every bar group for `\DTLmultibarchart`.

```
\newcommand*{\DTLeverybargrouphook}{}
```

`DTLbarsetupperlabelalign`

```
\NewDocumentCommand \DTLbarsetupperlabelalign { o m }
{
  \IfValueT { #1 }
   {
     \tl_set:Nn \DTLbarXnegupperlabelalign { #1 }
   }
  \tl_set:Nn \DTLbarXupperlabelalign { #2 }
}
```

Used by `upper-label-align` to allow for missing braces around mandatory part (e.g. `upper-label-align=[left]right` or `upper-label-align=right`).

```
\cs_new:Npn \__databar_set_upper_align:w #1 \q_stop
{
  \tl_if_head_eq_charcode:nNTF { #1 } [
   {
     \__databar_set_upper_align_both:w #1 \q_stop
   }
   {
     \tl_if_single:NTF { #1 }
```

```
      {
        \DTLbarsetupperlabelalign #1
      }
      {
        \DTLbarsetupperlabelalign { #1 }
      }
   }
}
\cs_new:Npn \__databar_set_upper_align_both:w [ #1 ] #2 \q_stop
{
  \tl_if_single:NTF { #2 }
   {
     \DTLbarsetupperlabelalign [ #1 ] #2
   }
   {
     \DTLbarsetupperlabelalign [ #1 ] { #2 }
   }
}
```

Initialisation code after options have been parsed:

```
\tl_new:N \l__databar_init_tl
```

Initialisation code before options have been parsed:

```
\tl_new:N \l__databar_pre_init_tl
```

```
\keys_define:nn { datatool/bar }
{
```

Variable used to create the bar chart. (Must be a control sequence.)

```
  variable .code:n =
   {
     \tl_if_single:nTF { #1 }
      {
        \tl_set:Nn \DTLbarvariable { #1 }
      }
      {
        \PackageError { databar }
        {
          Invalid ~ value ~ in ~ `variable = \tl_to_str:n { #1 }'
        }
        {
          A ~ single ~ control ~ sequence ~ expected ~ as ~ the ~
          value ~ for ~ the ~ `variable' ~ setting
        }
      }
   } ,
```

Variables used to create the multi bar chart. (Must be a comma separated list of control sequences.)

```
  variables .code:n =
   {
     \seq_set_from_clist:Nn
```

```
      \l__databar_variables_seq { #1 }
  },
```

Rounding:

```
round .int_set:N = \c@DTLbarroundvar ,
y-tick-round .code:n =
 {
   \tl_if_empty:nTF { #1 }
    {
      \tl_set:Nn \l__databar_ytic_round_tl
       { \int_use:N \c@DTLbarroundvar }
    }
    {
      \tl_set:Nn \l__databar_ytic_round_tl { #1 }
    }
 } ,
y-tick-round .default:n = { } ,
```

Synonym:

```
ytic-round .code:n =
 {
   \tl_if_empty:nTF { #1 }
    {
      \tl_set:Nn \l__databar_ytic_round_tl
       { \int_use:N \c@DTLbarroundvar }
    }
    {
      \tl_set:Nn \l__databar_ytic_round_tl { #1 }
    }
 } ,
ytic-round .default:n = { } ,
```

Bar width:

```
barwidth .dim_set:N = \DTLbarwidth,
bar-width .dim_set:N = \DTLbarwidth,
```

Lower bar labels

```
barlabel .tl_set:N = \l__databar_barlabel_tl ,
bar-label .tl_set:N = \l__databar_barlabel_tl ,
```

Lower bar labels for multi-bar charts:

```
multibarlabels .code:n =
 {
   \seq_set_from_clist:Nn
    \l__databar_multibarlabels_seq { #1 }
 },
multi-bar-labels .code:n =
 {
   \seq_set_from_clist:Nn
    \l__databar_multibarlabels_seq { #1 }
 },
```

Gap between bars:

```
bargap .tl_set:N = \l__databar_bargap_tl ,
bar-gap .tl_set:N = \l__databar_bargap_tl ,
```

Gap between groups in multi-bar charts:

```
groupgap .tl_set:N = \l__databar_groupgap_tl ,
group-gap .tl_set:N = \l__databar_groupgap_tl ,
```

Upper bar labels:

```
upperbarlabel .tl_set:N = \l__databar_upperbarlabel_tl ,
upper-bar-label .tl_set:N = \l__databar_upperbarlabel_tl ,
```

Upper bar labels for multi-bar charts:

```
uppermultibarlabels .code:n =
 {
   \seq_set_from_clist:Nn
     \l__databar_uppermultibarlabels_seq { #1 }
 },
upper-multi-bar-labels .code:n =
 {
   \seq_set_from_clist:Nn
     \l__databar_uppermultibarlabels_seq { #1 }
 },
```

Define list of points for y tics:

```
yticpoints .code:n =
 {
   \clist_set:Nn
     \l__databar_yticpoints_clist { #1 }
   \DTLbaryticstrue
   \DTLbaryaxistrue
 },
y-tick-points .code:n =
 {
   \clist_set:Nn
     \l__databar_yticpoints_clist { #1 }
   \DTLbaryticstrue
   \DTLbaryaxistrue
 },
```

Set the y tick gap:

```
yticgap .code:n =
 {
   \tl_set:Nn \l__databar_yticgap_tl { #1 }
   \DTLbaryticstrue
   \DTLbaryaxistrue
 },
y-tick-gap .code:n =
 {
   \tl_set:Nn \l__databar_yticgap_tl { #1 }
   \DTLbaryticstrue
```

```
      \DTLbaryaxistrue
    },
```

List of labels for y tics.

```
   yticlabels .code:n =
    {
      \seq_set_from_clist:Nn
        \l__databar_yticlabels_seq { #1 }
      \DTLbaryticstrue
      \DTLbaryaxistrue
    },
   y-tick-labels .code:n =
    {
      \seq_set_from_clist:Nn
        \l__databar_yticlabels_seq { #1 }
      \DTLbaryticstrue
      \DTLbaryaxistrue
    },
```

y axis label:

```
   ylabel .tl_set:N = \l__databar_ylabel_tl ,
```

y axis label position: y axis label:

```
   ylabel-position .choice: ,
   ylabel-position / center .code:n =
    {
      \tl_set:Nn \l__databar_ylabel_pos_tl
       { \c__databar_ylabel_pos_centred_tl }
    },
   ylabel-position / min .code:n =
    {
      \tl_set:Nn \l__databar_ylabel_pos_tl
       { \c__databar_ylabel_pos_min_tl }
    },
   ylabel-position / max .code:n =
    {
      \tl_set:Nn \l__databar_ylabel_pos_tl
       { \c__databar_ylabel_pos_max_tl }
    },
   ylabel-position / zero .code:n =
    {
      \tl_set:Nn \l__databar_ylabel_pos_tl
       { \c__databar_ylabel_pos_zero_tl }
    },
```

Vertical bars:

```
   vertical .code:n =
    {
      \DTLverticalbarstrue
    },
```

Horizontal:

```
  horizontal .code:n =
   {
     \DTLverticalbarsfalse
   },
```
Option that takes a value:
```
  verticalbars .legacy_if_set:n = DTLverticalbars ,
```
Label alignment:
```
  upper-label-align .code:n =
   {
     \tl_if_empty:nTF { #1 }
      {
        \tl_set:Nn \DTLbarXupperlabelalign
         {
           \ifDTLverticalbars
             bottom, center
           \else
            left
           \fi
         }
        \tl_set:Nn \DTLbarXnegupperlabelalign
         {
           \ifDTLverticalbars
            top, center
           \else
            right
           \fi
         }
      }
      {
       \__databar_set_upper_align:w #1 \q_stop
      }
   },
  upper-label-align .default:n = { } ,
  lower-label-style .choice: ,
  lower-label-style / below .code:n =
   {
     \tl_set:Nn \DTLbarXlabelalign
      {
        \ifDTLverticalbars
         left, rotate=-90
        \else
         right
        \fi
      }
     \tl_set:Nn \DTLbarXneglabelalign
      {
        \ifDTLverticalbars
         left, rotate=-90
        \else
```

```
      right
     \fi
   }
  \tl_set:Nn \l__databar_label_pos_offset_tl
    { \dim_to_decimal:n { -\DTLbarlabeloffset } pt }
  \tl_set:Nn \l__databar_label_neg_offset_tl
    { \dim_to_decimal:n { -\DTLbarlabeloffset } pt }
} ,
lower-label-style / above .code:n =
{
  \tl_set:Nn \DTLbarXlabelalign
   {
     \ifDTLverticalbars
       right, rotate=-90
     \else
       left
     \fi
   }
 \tl_set:Nn \DTLbarXneglabelalign
   {
     \ifDTLverticalbars
       right, rotate=-90
     \else
       left
     \fi
   }
  \tl_set:Nn \l__databar_label_pos_offset_tl
    { \dim_to_decimal:n { \DTLbarlabeloffset } pt }
  \tl_set:Nn \l__databar_label_neg_offset_tl
    { \dim_to_decimal:n { \DTLbarlabeloffset } pt }
} ,
lower-label-style / same .code:n =
{
 \tl_set:Nn \DTLbarXlabelalign
  {
    \ifDTLverticalbars
      right, rotate=-90
    \else
      left
    \fi
  }
 \tl_set:Nn \DTLbarXneglabelalign
  {
    \ifDTLverticalbars
      left, rotate=-90
    \else
      right
    \fi
  }
 \tl_set:Nn \l__databar_label_pos_offset_tl
```

```
      { \dim_to_decimal:n { \DTLbarlabeloffset } pt }
    \tl_set:Nn \l__databar_label_neg_offset_tl
      { \dim_to_decimal:n { -\DTLbarlabeloffset } pt }
   } ,
  lower-label-style / opposite .code:n =
   {
    \tl_set:Nn \DTLbarXneglabelalign
     {
       \ifDTLverticalbars
         right, rotate=-90
       \else
         left
       \fi
     }
    \tl_set:Nn \DTLbarXlabelalign
     {
       \ifDTLverticalbars
         left, rotate=-90
       \else
         right
       \fi
     }
    \tl_set:Nn \l__databar_label_pos_offset_tl
      { \dim_to_decimal:n { -\DTLbarlabeloffset } pt }
    \tl_set:Nn \l__databar_label_neg_offset_tl
      { \dim_to_decimal:n { \DTLbarlabeloffset } pt }
   } ,
```

Maximum value of the y axis:

```
  max .code:n =
   {
     \__databar_set_max:n { #1 }
   } ,
  max .default:n = { } ,
```

The total length of the bar chart

```
  length .dim_set:N = \DTLbarchartlength,
```

The maximum depth (negative extent):

```
  maxdepth .code:n =
   {
     \__databar_set_max_depth:n { #1 }
   },
  maxdepth .default:n = { } ,
  max-depth .code:n =
   {
     \__databar_set_max_depth:n { #1 }
   },
  max-depth .default:n = { } ,
```

Determine which axes should be shown:

```
axes .choice: ,
axes .default:n = { both } ,
axes / both .code:n =
 {
   \DTLbarxaxistrue
   \DTLbaryaxistrue
   \DTLbaryticstrue
 },
axes / x .code:n =
 {
   \DTLbarxaxistrue
   \DTLbaryaxisfalse
   \DTLbaryticsfalse
 },
axes / y .code:n =
 {
   \DTLbarxaxisfalse
   \DTLbaryaxistrue
   \DTLbaryticstrue
 },
axes / none .code:n =
 {
   \DTLbarxaxisfalse
   \DTLbaryaxisfalse
   \DTLbaryticsfalse
 },
yaxis .legacy_if_set:n = DTLbaryaxis,
y-axis .legacy_if_set:n = DTLbaryaxis,
y-ticks .choice:   ,
y-ticks / true .code:n =
 {
   \DTLbaryaxistrue
   \DTLbaryticstrue
 },
y-ticks / false .code:n =
 {
   \DTLbaryticsfalse
 },
y-ticks .default:n = true,
```

Synonym:

```
ytics .choice:   ,
ytics / true .code:n =
 {
   \DTLbaryaxistrue
   \DTLbaryticstrue
 },
ytics / false .code:n =
 {
   \DTLbaryticsfalse
```

```
 },
ytics .default:n = true,
```

Axis style:

```
x-axis-style .tl_set:N = \DTLBarXAxisStyle ,
y-axis-style .tl_set:N = \DTLBarYAxisStyle ,
```

Bar colours:

```
color-style .choice: ,
color-style / cycle .code:n =
 {
   \tl_set:Nn \l__databar_color_max_index_tl
    { \l__databar_max_colors_int }
 },
color-style / single .code:n =
 {
   \tl_set:Nn \l__databar_color_max_index_tl
    { \c_one_int }
 },
color-style / default .code:n =
 {
   \tl_set:Nn \l__databar_color_max_index_tl
    { \c_zero_int }
 },
bar-colors .code:n =
 {
   \seq_set_from_clist:Nn \l__datatool_tmp_seq { #1 }
   \seq_map_indexed_function:NN
     \l__datatool_tmp_seq
     \DTLsetbarcolor
 },
bar-default-colors .code:n =
 {
   \DTLcolorbarcharttrue
   \DTLsetbarcolor{1}{red}
   \DTLsetbarcolor{2}{green}
   \DTLsetbarcolor{3}{blue}
   \DTLsetbarcolor{4}{yellow}
   \DTLsetbarcolor{5}{magenta}
   \DTLsetbarcolor{6}{cyan}
   \DTLsetbarcolor{7}{orange}
   \DTLsetbarcolor{8}{white}
 },
bar-default-colors .value_forbidden:n = true,
bar-default-gray .code:n =
 {
   \DTLcolorbarchartfalse
   \DTLsetbarcolor{1}{black!15}
   \DTLsetbarcolor{2}{black!25}
   \DTLsetbarcolor{3}{black!35}
   \DTLsetbarcolor{4}{black!45}
```

```
      \DTLsetbarcolor{5}{black!55}
      \DTLsetbarcolor{6}{black!65}
      \DTLsetbarcolor{7}{black!75}
      \DTLsetbarcolor{8}{black!85}
    },
  bar-default-gray .value_forbidden:n = true,
```

Colours for bars with negative values.

```
  negative-bar-colors .code:n =
    {
      \seq_set_from_clist:Nn \l__datatool_tmp_seq { #1 }
      \seq_map_indexed_function:NN
        \l__datatool_tmp_seq
        \DTLsetnegbarcolor
    },
  negative-color-style .choice: ,
  negative-color-style / cycle .code:n =
    {
      \tl_set:Nn \l__databar_neg_color_max_index_tl
       { \l__databar_max_neg_colors_int }
    },
  negative-color-style / single .code:n =
    {
      \tl_set:Nn \l__databar_neg_color_max_index_tl
       { \c_one_int }
    },
```

Default negative-color-style matches color-style.

```
  negative-color-style / default .code:n =
    {
      \tl_set:Nn \l__databar_neg_color_max_index_tl
       { \l__databar_color_max_index_tl }
    },
```

Bar outline colour:

```
  outline-color .tl_set:N = \DTLbaroutlinecolor,
  outline-width .dim_set:N = \DTLbaroutlinewidth,
```

Distance from the bar to the bar label:

```
  label-offset .dim_set:N = \DTLbarlabeloffset ,
```

Distance from the outer edge of the bar to the upper bar label:

```
  upper-label-offset .tl_set:N = \l__databar_upper_label_offset_tl ,
```

group label alignment:

```
  group-label-align .tl_set:N = \DTLbargrouplabelalign ,
```

y tick label alignment:

```
  y-tick-label-align .tl_set:N = \DTLbarYticklabelalign ,
  y-tic-label-align .tl_set:N = \DTLbarYticklabelalign ,
  ytic-label-align .tl_set:N = \DTLbarYticklabelalign ,
```

Initialisation code:

```
   init .tl_set:N = \l__databar_init_tl ,
   pre-init .tl_set:N = \l__databar_pre_init_tl ,
   pre-init .groups:n = { pre-parse },
```

Filter:

```
   include-if .cs_set:Np = \__databar_filter:T #1,
   include-if-fn .code:n =
    {
      \cs_set_eq:NN \__databar_filter:T #1
    },
```

Deprecated experimental setting. TODO remove

```
   condition .code:n =
    {
      \PackageWarning{databar}{Deprecated ~ option ~ `condition'. ~
       Use ~ `include-if' ~ instead}
      \cs_set:Npn = \__databar_filter:T ##1 { #1 }
    }
}
```

Allow these keys to be set in `\DTLsetup{bar={...}}`

```
\keys_define:nn { datatool }
{
  bar .code:n = { \keys_set:nn { datatool/bar } { #1 } }
}
```

Provide a filter function:

```
\cs_new:Npn \__databar_filter:T #1 { #1 }
```

`\@dtl@bar` Current filtered row number. Version 3.0: replaced `\@dtl@bar` with:

```
\int_new:N \l__databar_index_int

\cs_new:Npn \__databar_filtered_map:nn #1 #2
{
  \int_zero:N \l__databar_index_int
  \DTLmapdata
  {
    \DTLmapgetvalues { #1 }
    \__databar_filter:T
     {
       \int_incr:N \l__databar_index_int
       #2
     }
  }
}
```

Boolean to check the current chart type.

```
\bool_new:N \l_databar_multi_bars_bool
\bool_set_false:N \l_databar_multi_bars_bool
```

\DTLbarchart

```
\DTLbarchart[⟨conditions⟩]{⟨option list⟩}{⟨db name⟩}
{⟨assign list⟩}
```

Make a bar chart from data given in data base ⟨*db name*⟩, where ⟨*assign list*⟩ is a comma-separated list of ⟨*cmd*⟩=⟨*key*⟩ pairs. ⟨*option list*⟩ must include `variable=`⟨*cmd*⟩, where ⟨*cmd*⟩ is included in ⟨*assign list*⟩. The optional argument ⟨*conditions*⟩ is the same as that for `\DTLforeach`.

```
\NewDocumentCommand \DTLbarchart { o m m m }
{
  \group_begin:
   \bool_set_false:N \l_databar_multi_bars_bool
   \tl_set:Nn \DTLbargroupindex { 0 }
   \IfValueT { #1 }
   {
     \cs_set:Npn \__databar_filter:T ##1
     {
       \ifthenelse { #1 } { ##1 } { }
     }
   }
   \tl_if_blank:nF { #3 }
    {
      \DTLsetup { default-name = { #3 } }
    }
   \keys_set_groups:nnn { datatool/bar } { pre-parse } { #2 }
   \l__databar_pre_init_tl
   \keys_set:nn { datatool/bar } { #2 }
   \l__databar_init_tl
   \cs_set_eq:NN \__databar_do:n \use:n
   \tl_if_empty:NT \DTLbarvariable
   {
     \cs_set_eq:NN \__databar_do:n \use_none:n
     \seq_if_empty:NTF \l__databar_variables_seq
     {
       \PackageError { databar }
       {
         \token_to_str:N \DTLbarchart \c_space_tl ~ missing ~ variable
       }
       {
         You ~ need ~ to ~ set ~ the ~ `variable' ~
         key ~ to ~ the ~ placeholder ~ command
       }
     }
     {
       \PackageError { databar }
       {
         \token_to_str:N \DTLbarchart \c_space_tl ~ missing ~ variable
       }
       {
```

```
          Remember ~ that ~ you ~ need ~ to ~ use ~ `variable' ~
          not ~ `variables' ~ with ~ \token_to_str:N \DTLbarchart
        }
      }
    }
    \tl_if_empty:NF \l__databar_yticgap_tl
     {
       \fp_set:Nn \l__databar_yticgap_fp
         { \l__databar_yticgap_tl }
       \fp_compare:nNnT { \l__databar_yticgap_fp } < { \c_zero_fp }
         {
           \PackageError { databar }
           {
             y-tick ~ gap ~ `\tl_to_str:o { \l__databar_yticgap_tl } ' ~
             is ~ negative
           }
           {
             The ~ y ~ tick ~ gap ~ specification ~ evaluates ~ to ~
             a ~ negative ~ number ~ ( \fp_to_decimal:N \l__databar_yticgap_fp) ~
             so ~ the ~ yticgap ~ setting ~ will ~ be ~ ignored
           }
           \tl_clear:N \l__databar_yticgap_tl
           \fp_zero:N \l__databar_yticgap_fp
         }
     }
    \__databar_do:n
    {
      \__databar_do_chart:n { #4 }
    }
   \group_end:
}

\cs_new:Nn \__databar_do_chart:n
{
```

Clear the underlying placeholder command used to reference the variable.

```
  \exp_args:NV \tl_clear:N \DTLbarvariable
  \tl_clear:N \DTLtotalbars
  \__datatool_calc_bar_lengths:n { #1 }
```

Calculate the total number of bars if not already found.

```
  \tl_if_empty:NT \DTLtotalbars
   {
     \__databar_filtered_map:nn { #1 } { }
     \tl_set:NV \DTLtotalbars \l__databar_index_int
   }
```

A bar width is one unit so the chart width is the number of bars.

```
  \tl_set_eq:NN \DTLbarchartwidth \DTLtotalbars
```

Add on the inter bar gaps:

```
  \tl_if_eq:NnF \l__databar_bargap_tl { 0 }
   {
     \tl_set:Nx \DTLbarchartwidth
      {
        \fp_eval:n
        {
          \DTLbarchartwidth
            + \l__databar_bargap_tl
             * (\DTLtotalbars - \c_one_fp )
        }
      }
   }
  \__databar_calc_scale:
  \__databar_construct_ytics:
```

Draw the chart.

```
  \__databar_chart:n { #1 }
 }
```

Action ‘bar chart’:

```
\cs_new:cn { __datatool_action_ bar ~ chart : }
{
 \group_begin:
   \tl_if_empty:NF \l__datatool_action_name_tl
    {
      \tl_set_eq:NN
        \l__datatool_default_dbname_tl
        \l__datatool_action_name_tl
    }
   \bool_set_false:N \l_databar_multi_bars_bool
   \cs_set_eq:NN \__databar_do:n \use:n
   \tl_set:Nn \DTLbargroupindex { 0 }
```

If the ‘key’ or ‘column’ has been set, then that provides the variable, unless it’s overridden by the variable setting in the options.

```
    \__datatool_optional_key_xor_column_get_key:nTF
     {
```

No key or column. The variable will need to be provided in the options list instead (or may have already been set with `\DTLsetup`).

```
     }
     {
```

Key or column provided.

```
       \tl_set:Nn \DTLbarvariable
         { \l__databar_action_variable_tl }
       \clist_put_right:Nx \l__datatool_action_assign_clist
        {
          \exp_not:N \l__databar_action_variable_tl =
           \l__datatool_action_key_tl
        }
    }
```

```
    {
```
Invalid syntax.
```
      \cs_set_eq:NN \__databar_do:n \cs_none:n
    }
  \__databar_do:n
   {
```
Parse options if set.
```
    \clist_if_empty:NTF \l__datatool_action_options_clist
     {
       \l__databar_pre_init_tl
     }
     {
       \keys_set_groups:nnV { datatool/bar } { pre-parse }
         \l__datatool_action_options_clist
       \l__databar_pre_init_tl
       \keys_set:nV { datatool/bar }
         \l__datatool_action_options_clist
     }
    \l__databar_init_tl
```
Check variable has been set.
```
    \tl_if_empty:NT \DTLbarvariable
    {
      \cs_set_eq:NN \__databar_do:n \use_none:n
      \seq_if_empty:NTF \l__databar_variables_seq
      {
       \__datatool_action_error:nnn { databar }
        {
          \token_to_str:N \DTLbarchart \c_space_tl ~ missing ~ variable
        }
        {
          You ~ need ~ to ~ either ~ use ~ `key' ~ or ~ `column' ~
          to ~ identify ~ the ~ variable ~ column ~ or ~ set ~
          the ~ `variable' ~ key ~ to ~ the ~ placeholder ~
          command ~ within ~ `options'
        }
      }
      {
       \__datatool_action_error:nnn { databar }
        {
          missing ~ variable ~ (either ~ use ~ `key' ~ or ~ `column' ~
          to ~ identify ~ the ~ variable ~ column ~ or ~ set ~
          the ~ `variable' ~ key ~ to ~ the ~ placeholder ~
          command ~ within ~ `options')
        }
        {
          Remember ~ that ~ you ~ need ~ to ~ use ~ `variable' ~
          not ~ `variables' ~ for ~ a ~ single ~ variable ~ bar ~ chart
        }
```

```
      }
    }
   \tl_if_empty:NF \l__databar_yticgap_tl
    {
      \fp_set:Nn \l__databar_yticgap_fp
       { \l__databar_yticgap_tl }
      \fp_compare:nNnT { \l__databar_yticgap_fp } < { \c_zero_fp }
       {
        \__datatool_action_error:nnn { databar }
         {
           y-tick ~ gap ~ `\tl_to_str:o { \l__databar_yticgap_tl } ' ~
           is ~ negative
         }
         {
           The ~ y ~ tick ~ gap ~ specification ~ evaluates ~ to ~
           a ~ negative ~ number ~ ( \fp_to_decimal:N \l__databar_yticgap_fp) ~
           so ~ the ~ yticgap ~ setting ~ will ~ be ~ ignored
         }
         \tl_clear:N \l__databar_yticgap_tl
         \fp_zero:N \l__databar_yticgap_fp
       }
    }
   \__databar_do:n
    {
      \exp_args:NV \__databar_do_chart:n
       \l__datatool_action_assign_clist
    }
  }
 \group_end:
}
```

Scratch placeholder for action if key or column given instead of variable:

```
\tl_new:N \l__databar_action_variable_tl

\cs_new:Nn \__databar_single_update:
{
  \exp_args:NNV \tl_set_eq:NN
   \l__dataplot_y_tl \DTLbarvariable
  \__databar_update_bounds:
}
\cs_new:Nn \__databar_do_update:
{
  \__databar_single_update:
}
\cs_new:Nn \__databar_multi_variables_update:
{
  \seq_map_inline:Nn \l__databar_variables_seq
   {
     \tl_set_eq:NN \l__dataplot_y_tl ##1
      \__databar_update_bounds:
   }
```

```
}
\cs_new:Nn \__databar_multi_columns_update:
{
  \seq_map_inline:Nn \l__datatool_action_columns_seq
   {
     \exp_args:NNno
       \dtl@getentryfromrow
       \l__dataplot_y_tl
       { ##1 }
       \l__datatool_map_data_row_tl
     \__databar_update_bounds:
   }
}
```

Compute the bar heights unless `\DTLbarmax` has been set and negative extent unless `\DTLnegextent` has been set. This needs to take into account number formatting.

```
\cs_new:Nn \__datatool_calc_bar_lengths:n
{
  \bool_lazy_or:nnTF
     { \tl_if_empty_p:N \DTLbarmax }
     { \tl_if_empty_p:N \DTLnegextent }
   {
     \fp_set_eq:NN \l__dataplot_max_y_fp \c_minus_inf_fp
     \fp_zero:N \l__dataplot_min_y_fp
     \__databar_filtered_map:nn { #1 }
      {
        \__databar_do_update:
      }
     \tl_set:NV \DTLtotalbars \l__databar_index_int
     \tl_if_empty:NTF \DTLbarmax
      {
        \fp_compare:nNnTF
           { \l__dataplot_max_y_fp } > { \c_minus_inf_fp }
         {
           \tl_set:Nx \DTLbarmax
             { \fp_to_decimal:N \l__dataplot_max_y_fp }
         }
         {
           \PackageError { databar }
            {
               Can't ~ determine ~ maximum ~ extent. ~
               Check ~ data ~ and ~ filtering
            }
            {
              Either ~ there's ~ no ~ numeric ~ data ~ provided ~ in ~
              the ~ column ~ referenced ~ by ~ \tl_to_str:V \DTLbarvariable
              \c_space_tl ~ or ~ all ~ rows ~ have ~ been ~ filtered
            }
           \tl_set:Nn \DTLbarmax { 1 }
```

```
          }
        \tl_if_empty:NF \l__databar_yticgap_tl
          {
            \fp_set_eq:NN \l__dataplot_y_fp \l__databar_yticgap_fp
            \fp_while_do:nNnn { \l__dataplot_y_fp } < { \l__dataplot_max_y_fp }
              {
                \fp_add:Nn \l__dataplot_y_fp { \l__databar_yticgap_fp }
              }
            \tl_set:Nx \DTLbarmax { \fp_to_decimal:N \l__dataplot_y_fp }
          }
      }
      {
        \fp_set:Nn \l__dataplot_max_y_fp { \DTLbarmax }
      }
    \tl_if_empty:NTF \DTLnegextent
      {
        \fp_compare:nNnTF
          { \l__dataplot_min_y_fp } < { \c_zero_fp }
          {
            \tl_set:Nx \DTLnegextent
              { \fp_to_decimal:N \l__dataplot_min_y_fp }
            \tl_if_empty:NF \l__databar_yticgap_tl
              {
                \fp_set:Nn \l__dataplot_y_fp { - \l__databar_yticgap_fp }
                \fp_while_do:nNnn { \l__dataplot_y_fp } > { \l__dataplot_min_y_fp }
                  {
                    \fp_sub:Nn \l__dataplot_y_fp { \l__databar_yticgap_fp }
                  }
                \tl_set:Nx \DTLnegextent { \fp_to_decimal:N \l__dataplot_y_fp }
              }
          }
          {
            \tl_set:Nn \DTLnegextent { 0 }
          }
      }
      {
        \fp_set:Nn \l__dataplot_min_y_fp { \DTLnegextent }
      }
  }
  {
    \tl_if_empty:NF \DTLbarmax
      {
        \fp_set:Nn \l__dataplot_max_y_fp { \DTLbarmax }
      }
    \tl_if_empty:NF \DTLnegextent
      {
        \fp_set:Nn \l__dataplot_min_y_fp { \DTLnegextent }
      }
  }
}
```

Update max and min values. The current value from the database column should be in `\l__dataplot_y_tl`.

```
\cs_new:Nn \__databar_update_bounds:
{
  \tl_if_empty:NF \l__dataplot_y_tl
   {
     \datatool_set_fp:Nn \l__dataplot_y_fp
       { \l__dataplot_y_tl }
     \tl_if_empty:NT \DTLbarmax
      {
       \fp_compare:nNnT
         { \l__dataplot_y_fp }
           >
         { \l__dataplot_max_y_fp }
        {
         \fp_set_eq:NN
            \l__dataplot_max_y_fp
            \l__dataplot_y_fp
        }
      }
     \tl_if_empty:NT \DTLnegextent
      {
       \fp_compare:nNnT
         { \l__dataplot_y_fp }
          <
         { \l__dataplot_min_y_fp }
        {
         \fp_set_eq:NN
            \l__dataplot_min_y_fp
            \l__dataplot_y_fp
        }
      }
   }
}
```

Calculate the scaling factor.

```
\cs_new:Nn \__databar_calc_scale:
{
  \fp_set:Nn \l__databar_extent_fp
   { \DTLbarmax - \DTLnegextent }
  \fp_set:Nn \l__databar_unit_fp
   {
     \dim_to_decimal_in_sp:n { \DTLbarchartlength }
       / \l__databar_extent_fp
   }
  \dim_set:Nn \l__databar_miny_dim
   {
     \fp_eval:n
      {
        \DTLnegextent * \l__databar_unit_fp
```

```
      }
      sp
    }
  \dim_set:Nn \l__databar_maxy_dim
    {
      \fp_eval:n
       {
         \DTLbarmax * \l__databar_unit_fp
       }
       sp
    }
}
```

Construct y tick list if required.

```
\cs_new:Nn \__databar_construct_ytics:
{
  \dim_zero:N \l__dataplot_y_tic_label_width_dim
  \ifDTLbarytics
    \clist_if_empty:NTF \l__databar_yticpoints_clist
     {
       \tl_if_empty:NTF \l__databar_yticgap_tl
        {
          \fp_set:Nn \l__databar_min_gap_fp
           {
             \dim_to_decimal_in_sp:n { \DTLmintickgap }
              / \l__databar_unit_fp
           }
          \__dataplot_construct_tick_list:NNNN
           \l__dataplot_min_y_fp
           \l__dataplot_max_y_fp
           \l__databar_min_gap_fp
           \l__databar_yticpoints_seq
        }
        {
          \__dataplot_construct_tick_list_with_gap_excl:NNNN
           \l__dataplot_min_y_fp
           \l__dataplot_max_y_fp
           \l__databar_yticpoints_seq
           \l__databar_yticgap_fp
        }
     }
     {
       \__dataplot_to_fp_seq:NNnn
        \l__databar_yticpoints_seq
        \l__databar_y_tic_clist
        \l__dataplot_min_y_fp
        \l__dataplot_max_y_fp
     }
    \seq_if_empty:NT \l__databar_yticlabels_seq
     {
```

```
        \seq_map_inline:Nn \l__databar_yticpoints_seq
         {
           \tl_set:Nn \l__dataplot_y_fp { ##1 }
           \__dataplot_get_default_tic_label:Nx
             \l__dataplot_y_fp { \l__databar_ytic_round_tl }
           \seq_put_right:Nx \l__databar_yticlabels_seq
             {
               \exp_not:V \l__dataplot_tic_label_tl
             }
         }
      }
  \fi
}
```

Do the bar chart

```
\cs_new:Nn \__databar_chart:n
{
  \begin{tikzpicture}
```

Set unit vectors

```
  \ifDTLverticalbars
    \pgfsetyvec
     {
       \pgfpoint
        { 0pt } { \fp_to_int:N \l__databar_unit_fp ~ sp }
     }
    \pgfsetxvec
     {
       \pgfpoint { \DTLbarwidth } { 0pt }
     }
  \else
    \pgfsetxvec
     {
       \pgfpoint
        { \fp_to_int:N \l__databar_unit_fp ~ sp } { 0pt }
     }
    \pgfsetyvec
     {
       \pgfpoint { 0pt } { \DTLbarwidth }
     }
  \fi
```

Begin hook

```
  \DTLbaratbegintikz
```

Initialise the start point. This is the y co-ordinate for the first corner of the bar along the y axis. The end point is the start point plus one unit (which will be the same as the bar index).

```
  \fp_zero:N \l__databar_start_fp
```

Iterate through data

```
  \__databar_filtered_map:nn { #1 }
   {
```
The bar index is the same as the value of `\l__databar_index_int`:
```
    \tl_set:Nx \DTLbarindex { \int_use:N \l__databar_index_int }
```
Draw the current bar.
```
    \__databar_draw_bar:
   }
```
Clear content token list variable.
```
  \tl_clear:N \l__dataplot_content_tl
```
Add axes drawing code to content:
```
  \__databar_draw_axes:
```
Add y tick marks if required
```
  \seq_map_indexed_function:NN
    \l__databar_yticpoints_seq
    \__databar_draw_y_tics_fn:nn
```
Do any pending content and clear token list variable.
```
  \l__dataplot_content_tl
  \tl_clear:N \l__dataplot_content_tl
```
Add the y label if required
```
  \__databar_plot_ylabel:
```
Do any pending content and clear token list variable.
```
  \l__dataplot_content_tl
  \tl_clear:N \l__dataplot_content_tl
```
End hook
```
  \DTLbaratendtikz
  \end{tikzpicture}
}
```
Draw bar:
```
\cs_new:Nn \__databar_draw_bar:
{
```
`\DTLbarvariable` should be defined to the actual placeholder command.
```
  \__databar_draw_bar:NVxx \DTLbarvariable
    \l__databar_index_int
    {
      \tl_if_empty:NF \l__databar_barlabel_tl
       {
         \exp_not:N \DTLdisplaylowerbarlabel
          { \exp_not:V \l__databar_barlabel_tl }
       }
    }
    {
      \tl_if_empty:NF \l__databar_upperbarlabel_tl
       {
```

```
          \exp_not:N \DTLdisplayupperbarlabel
           { \exp_not:V \l__databar_upperbarlabel_tl }
        }
     }
```

Update start.

```
   \fp_add:Nn \l__databar_start_fp { \c_one_fp }
 }
```

`\__databar_draw_bar:Nnnn`⟨*tl-var*⟩`{`⟨*bar index*⟩`}{`⟨*lower label*⟩`}{`⟨*upper label*⟩`}` Draw bar for the given variable. The starting point `\l__databar_start_int` should already be set. For multi bar charts, the bar index is the index within the group (that is, it's reset to 1 at the start of each group).

```
 \cs_new:Nn \__databar_draw_bar:Nnnn
 {
```

Add the inter bar gap if this isn't the first bar.

```
   \tl_if_eq:NnF \l__databar_bargap_tl { 0 }
    {
      \int_compare:nNnT
        { #2 } > { \c_one_int }
       {
         \fp_add:Nn \l__databar_start_fp
          { \l__databar_bargap_tl }
       }
    }
   \tl_set:Nx \l__databar_start_tl
     { \fp_to_decimal:N \l__databar_start_fp }
```

Content token list variable is reset for each bar to allow the hook to access the current bar settings.

```
   \tl_clear:N \l__dataplot_content_tl
```

Get the variable. The original variable data is stored in `\l__dataplot_y_tl` but this will be a formatted number that needs converting to a decimal. This is stored twice: as a floating point variable `\l__dataplot_y_fp` to avoid repeated parsing where calculations are required, and as a token list variable `\l__dataplot_decimal_y_tl` where the value needs expanding for the pgf parser.

```
   \tl_set_eq:NN \l__dataplot_y_tl #1
   \tl_if_empty:NTF \l__dataplot_y_tl
    {
```

Missing data for this row so assume 0.

```
      \fp_zero:N \l__dataplot_y_fp
      \tl_set:Nn \l__dataplot_decimal_y_tl { 0 }
      \tl_set:Nn \DTLbarvalue
        { \__datatool_datum:nnnn { 0 } { 0 } { } { \c_datatool_decimal_int } }
    }
    {
      \datatool_set_fp:NV \l__dataplot_y_fp
         \l__dataplot_y_tl
```

```
      \tl_set:Nx \l__dataplot_decimal_y_tl
        { \fp_to_tl:N \l__dataplot_y_fp }
```

Set `\DTLbarvalue` to the rounded formatted value for use in the labels or hook.

```
      \tl_set:Nx \l__datatool_result_tl
       {
         \fp_to_decimal:n
          { round ( \l__dataplot_y_fp, \c@DTLbarroundvar) }
       }
      \datatool_pad_trailing_zeros:Nn
       \l__datatool_result_tl \c@DTLbarroundvar
      \exp_args:NV \DTLdecimaltolocale
        \l__datatool_result_tl \l__datatool_tmpa_tl
      \tl_set:Nx \DTLbarvalue
        {
          \exp_not:N \__datatool_datum:nnnn
          { \exp_not:V \l__datatool_tmpa_tl }
          { \exp_not:V \l__datatool_result_tl }
          { }
          { \exp_not:N \c_datatool_decimal_int }
        }
    }
```

Mid point of bar (along the y axis).

```
  \tl_set:Nx \l__databar_mid_tl
    { \fp_eval:n { \l__databar_start_fp + 0.5 } }
  \dim_set_eq:NN \l__databar_mid_dim \DTLbarwidth
  \dim_set:Nn \l__databar_mid_dim
    {
      \l__databar_mid_tl \DTLbarwidth
    }
  \dim_set:Nn \l__databar_extent_dim
    {
      \fp_eval:n
      {
        \l__dataplot_y_fp * \l__databar_unit_fp
      }
      sp
    }
```

End point.

```
  \tl_set:Nx \l__databar_end_tl
    { \fp_eval:n { \l__databar_start_fp + \c_one_fp } }
```

Set placeholder commands `\DTLstartpt`, `\DTLmidpt` and `\DTLendpt`. These are the start, middle and end of the bar along the bar's mid axis.

```
  \ifDTLverticalbars
   \tl_set:Nx \DTLstartpt
     {
       \exp_not:N \pgfpointxy
         { \l__databar_mid_tl } { 0 }
```

```
    }
  \tl_set:Nx \DTLmidpt
    {
      \exp_not:N \pgfpointxy
        { \l__databar_mid_tl }
        { \fp_to_decimal:n { 0.5 \l__dataplot_y_fp } }
    }
  \tl_set:Nx \DTLendpt
    {
      \exp_not:N \pgfpointxy
        { \l__databar_mid_tl }
        { \l__dataplot_decimal_y_tl }
    }
\else
   \tl_set:Nx \DTLstartpt
     {
       \exp_not:N \pgfpointxy
       { 0 } { \l__databar_mid_tl }
     }
  \tl_set:Nx \DTLmidpt
    {
      \exp_not:N \pgfpointxy
        { \fp_to_decimal:n { 0.5 \l__dataplot_y_fp } }
        { \l__databar_mid_tl }
    }
  \tl_set:Nx \DTLendpt
    {
      \exp_not:N \pgfpointxy
        { \l__dataplot_decimal_y_tl }
        { \l__databar_mid_tl }
    }
\fi
```

Draw bar.

```
\tl_put_right:Nn \l__dataplot_content_tl
   {
     \begin{scope}
   }
```

Every pre bar hook

```
\tl_put_right:Nn \l__dataplot_content_tl
  { \DTLeveryprebarhook }
\fp_compare:nNnTF { \l__dataplot_y_fp } < { \c_zero_fp }
  {
    \tl_set:Nx \l__datatool_tmpa_tl
     {
       \DTLgetnegbarcolor { #2 }
     }
  }
  {
    \tl_set:Nx \l__datatool_tmpa_tl
```

```
    {
      \DTLgetbarcolor { #2 }
    }
  }
\tl_put_right:Nn \l__dataplot_content_tl
 {
   \path ~ [
 }
\tl_if_empty:NF \l__datatool_tmpa_tl
  {
    \tl_put_right:Nx \l__dataplot_content_tl
    {
       fill = \l__datatool_tmpa_tl ,
    }
  }
\bool_lazy_and:nnT
 {
   \tl_if_empty_p:N \DTLbaroutlinecolor
 }
 {
   \dim_compare_p:nNn
     { \DTLbaroutlinewidth } > { \c_zero_dim }
 }
 {
   \tl_put_right:Nx \l__dataplot_content_tl
   {
      draw = \DTLbaroutlinecolor ,
      line ~ width = \exp_not:N \DTLbaroutlinewidth ,
   }
 }
\tl_put_right:Nx \l__dataplot_content_tl
 {
   \DTLBarStyle ]
 }
\ifDTLverticalbars
  \tl_put_right:Nx \l__dataplot_content_tl
   {
     ( \l__databar_start_tl , ~ 0) ~
      -- ~
     (
       \l__databar_start_tl , ~
       \l__dataplot_decimal_y_tl
     ) ~
      -- ~
     (
       \l__databar_end_tl, ~
       \l__dataplot_decimal_y_tl
     ) ~
     -- ~
     ( \l__databar_end_tl , ~ 0)
```

```
    }
  \else
    \tl_put_right:Nx
      \l__dataplot_content_tl
      {
        ( 0 , ~ \l__databar_start_tl ) ~
         -- ~
        (
          \l__dataplot_decimal_y_tl , ~
          \l__databar_start_tl
        ) ~
         -- ~
        (
          \l__dataplot_decimal_y_tl , ~
          \l__databar_end_tl
        ) ~
        -- ~
        ( 0 , ~ \l__databar_end_tl )
      }
  \fi
  \tl_put_right:Nn \l__dataplot_content_tl
   {
     -- ~ cycle; ~
     \end{scope}
   }
```

Draw lower bar label.

```
   \fp_compare:nNnTF { \l__dataplot_y_fp } < { \c_zero_fp }
    {
      \tl_set_eq:NN \l__databar_label_current_offset_tl
        \l__databar_label_neg_offset_tl
      \tl_set:Nx \l__databar_label_current_align_tl
        { \DTLbarXneglabelalign }
    }
    {
      \tl_set_eq:NN \l__databar_label_current_offset_tl
        \l__databar_label_pos_offset_tl
      \tl_set:Nx \l__databar_label_current_align_tl
        { \DTLbarXlabelalign }
    }
    \tl_if_empty:nF { #3 }
    {
     \tl_put_right:Nn \l__dataplot_content_tl
      {
        \pgftext [ at =
      }
     \ifDTLverticalbars
      \tl_put_right:Nx \l__dataplot_content_tl
       {
         {
```

```
            \exp_not:N \pgfpoint
              { \l__databar_mid_dim }
              { \l__databar_label_current_offset_tl }
          }
        }
      \else
       \tl_put_right:Nx \l__dataplot_content_tl
        {
          {
             \exp_not:N \pgfpoint
               { \l__databar_label_current_offset_tl }
               { \l__databar_mid_dim }
          }
        }
      \fi
      \tl_put_right:Nx \l__dataplot_content_tl
       {
          , \exp_not:V \l__databar_label_current_align_tl
        ]
       }
      \tl_put_right:Nn \l__dataplot_content_tl
       {
         { #3 }
       }
      \bool_lazy_and:nnT
        { \bool_if_p:N \l_databar_multi_bars_bool }
        { ! \tl_if_empty_p:N \l__databar_barlabel_tl }
       {
         \tl_put_right:Nn \l__dataplot_content_tl
          { \__databar_update_group_offset: }
       }
    }
```

Draw upper bar label

```
  \tl_if_empty:nF { #4 }
   {
    \tl_put_right:Nn \l__dataplot_content_tl
     {
       \pgftext [ at =
     }
    \ifDTLverticalbars
```

Vertical bars.

```
      \fp_compare:nNnTF
         { \l__dataplot_y_fp } < { \c_zero_fp }
       {
         \tl_put_right:Nx \l__dataplot_content_tl
          {
            {
               \exp_not:N \pgfpoint
                { \l__databar_mid_dim }
```

```
                {
                  \dim_eval:n
                   {
                     \l__databar_extent_dim
                     + \l__databar_upper_label_neg_offset_tl
                   }
                }
              }
              , ~ \DTLbarXnegupperlabelalign
            }
          }
          {
            \tl_put_right:Nx \l__dataplot_content_tl
             {
               {
                  \exp_not:N \pgfpoint
                   { \l__databar_mid_dim }
                   {
                     \dim_eval:n
                      {
                        \l__databar_extent_dim
                        + \l__databar_upper_label_pos_offset_tl
                      }
                   }
              }
              , ~ \DTLbarXupperlabelalign
            }
          }
      \else
```

Horizontal bars.

```
        \fp_compare:nNnTF
           { \l__dataplot_y_fp } < { \c_zero_fp }
         {
           \tl_put_right:Nx \l__dataplot_content_tl
            {
              {
                 \exp_not:N \pgfpoint
                  {
                    \dim_eval:n
                     {
                       \l__databar_extent_dim
                       + \l__databar_upper_label_neg_offset_tl
                     }
                  }
                  { \l__databar_mid_dim }
              }
              , ~ \DTLbarXnegupperlabelalign
            }
         }
```

```
        {
          \tl_put_right:Nx \l__dataplot_content_tl
           {
             {
                \exp_not:N \pgfpoint
                 {
                   \dim_eval:n
                    {
                      \l__databar_extent_dim
                      + \l__databar_upper_label_pos_offset_tl
                    }
                 }
                 { \l__databar_mid_dim }
             }
             , ~ \DTLbarXupperlabelalign
           }
        }
      \fi
      \tl_put_right:Nn \l__dataplot_content_tl
       {
         ]
         { #4 }
       }
     }
```

Every bar hook

```
   \tl_put_right:Nn \l__dataplot_content_tl
     { \DTLeverybarhook }
```

Do content and clear token list variable.

```
   \l__dataplot_content_tl
   \tl_clear:N \l__dataplot_content_tl
 }
 \cs_generate_variant:Nn \__databar_draw_bar:Nnnn
  { NVxx , NnVV }
```

Draw axes if applicable.

```
 \cs_new:Nn \__databar_draw_axes:
 {
```

Draw x axis

```
   \ifDTLbarxaxis
     \tl_put_right:Nx \l__dataplot_content_tl
      {
        \exp_not:N \draw [ \DTLBarXAxisStyle ] ~
        (0,0) ~ -- ~
      }
     \ifDTLverticalbars
       \tl_put_right:Nn \l__dataplot_content_tl
        {
          ( \DTLbarchartwidth, 0 );
```

```
      }
    \else
      \tl_put_right:Nn \l__dataplot_content_tl
      {
        ( 0, \DTLbarchartwidth );
      }
    \fi
  \fi
```

Draw y axis

```
  \ifDTLbaryaxis
    \tl_put_right:Nx \l__dataplot_content_tl
    {
      \exp_not:N \draw [ \DTLBarYAxisStyle ] ~
    }
    \ifDTLverticalbars
      \tl_put_right:Nn \l__dataplot_content_tl
      {
        ( 0, \DTLnegextent ) ~ -- ~ ( 0, \DTLbarmax );
      }
    \else
      \tl_put_right:Nn \l__dataplot_content_tl
      {
        ( \DTLnegextent, 0) ~ -- ~ ( \DTLbarmax, 0 );
      }
    \fi
  \fi
}
\cs_new:Nn \__databar_draw_y_tics_fn:nn
{
  \tl_set:Nn \l__dataplot_y_fp { #2 }
  \tl_set:Nx \l__dataplot_decimal_y_tl
    { \fp_to_decimal:N \l__dataplot_y_fp }
  \dim_set:Nn \l__databar_extent_dim
    {
      \fp_eval:n
      {
        \l__dataplot_y_fp * \l__databar_unit_fp
      }
      sp
    }
  \ifDTLverticalbars
    \tl_put_right:Nx \l__dataplot_content_tl
    {
      \exp_not:N \pgfpathmoveto
        {
          \exp_not:N \pgfpointxy
            { 0 } { \l__dataplot_decimal_y_tl }
        }
      \exp_not:N \pgfpathlineto
```

```
    {
      \exp_not:N \pgfpoint
       { \dim_to_decimal:n { - \DTLticklength } pt }
       { \l__databar_extent_dim }
    }
  }
\else
 \tl_put_right:Nx \l__dataplot_content_tl
  {
    \exp_not:N \pgfpathmoveto
     {
       \exp_not:N \pgfpointxy
         { \l__dataplot_decimal_y_tl } { 0 }
     }
    \exp_not:N \pgfpathlineto
     {
       \exp_not:N \pgfpoint
        { \l__databar_extent_dim }
        { \dim_to_decimal:n { - \DTLticklength } pt }
     }
  }
\fi
\tl_put_right:Nn \l__dataplot_content_tl
 {
   \pgfusepath{stroke}
 }
\int_compare:nNnTF
  { #1 } > { \seq_count:N \l__databar_yticlabels_seq }
 {
   \__dataplot_get_default_tic_label:Nx
    \l__dataplot_y_fp { \l__databar_ytic_round_tl }
 }
 {
   \tl_set:Nx \l__dataplot_tic_label_tl
    { \seq_item:Nn \l__databar_yticlabels_seq { #1 } }
 }
\tl_put_right:Nx \l__dataplot_content_tl
 {
   \exp_not:N \pgftext [ \DTLbarYticklabelalign , at=
 }
\ifDTLverticalbars
  \tl_put_right:Nx \l__dataplot_content_tl
   {
     {
       \exp_not:N \pgfpoint
         { \dim_to_decimal:n { -\DTLticklabeloffset } pt }
         { \l__databar_extent_dim }
     }
   }
\else
```

```
    \tl_put_right:Nx \l__dataplot_content_tl
     {
       {
         \exp_not:N \pgfpoint
           { \l__databar_extent_dim }
           { \dim_to_decimal:n { -\DTLticklabeloffset } pt }
       }
     }
  \fi
  \tl_put_right:Nx \l__dataplot_content_tl
   {
     ]
     {
       \exp_not:N \DTLbardisplayYticklabel
         { \exp_not:V \l__dataplot_tic_label_tl }
     }
   }
  \tl_if_empty:NF \l__databar_ylabel_tl
   {
     \tl_put_right:Nn \l__dataplot_content_tl
      { \__databar_update_ylabel_offset: }
   }
  \l__dataplot_content_tl
  \tl_clear:N \l__dataplot_content_tl
}
\cs_new:Nn \__databar_update_ylabel_offset:
{
  \ifDTLverticalbars
   \dim_set:Nn
    \l__dataplot_y_tic_label_width_dim
    {
      \dim_max:nn
       { \l__dataplot_y_tic_label_width_dim }
       { \dim_abs:n { \pgf@pathminx} }
    }
  \else
   \dim_set:Nn
    \l__dataplot_y_tic_label_width_dim
    {
      \dim_max:nn
       { \l__dataplot_y_tic_label_width_dim }
       { \dim_abs:n { \pgf@pathminy} }
    }
  \fi
}
\cs_new:Nn \__databar_plot_ylabel:
{
  \tl_if_empty:NF \l__databar_ylabel_tl
  {
```

```
    \dim_add:Nn \l__dataplot_y_tic_label_width_dim
      { \l__databar_ylabel_offset_tl }
    \tl_put_right:Nx \l__dataplot_content_tl
     {
       \exp_not:N \pgftext
        [
          at = { \l__databar_ylabel_pos_tl } ,
          \DTLyAxisLabelStyle
        ]
        { \exp_not:V \l__databar_ylabel_tl }
     }
  }
}
```

\DTLmultibarchart

\DTLmultibarchart[⟨*conditions*⟩]{⟨*option list*⟩}{⟨*db name*⟩}{⟨*assign list*⟩}

Make a multi-bar chart from data given in data base ⟨*db name*⟩, where ⟨*assign list*⟩ is a comma-separated list of ⟨*cmd*⟩=⟨*key*⟩ pairs. ⟨*option list*⟩ must include the `variables` key which must be a comma separated list of commands, where each command is included in ⟨*assign list*⟩. The optional argument ⟨*conditions*⟩ is the same as that for `\DTLforeach`.

```
\NewDocumentCommand \DTLmultibarchart { o m m m }
{
  \group_begin:
   \bool_set_true:N \l_databar_multi_bars_bool
   \IfValueT { #1 }
   {
     \cs_set:Npn \__databar_filter:T ##1
     {
       \ifthenelse { #1 } { ##1 } { }
     }
   }
   \tl_if_blank:nF { #3 }
    {
      \DTLsetup { default-name = { #3 } }
    }
   \keys_set_groups:nnn { datatool/bar } { pre-parse } { #2 }
   \l__databar_pre_init_tl
   \keys_set:nn { datatool/bar } { #2 }
   \l__databar_init_tl
   \cs_set_eq:NN \__databar_do:n \use:n
   \seq_if_empty:NT \l__databar_variables_seq
   {
     \cs_set_eq:NN \__databar_do:n \use_none:n
     \tl_if_empty:NTF \DTLbarvariable
     {
       \PackageError { databar }
```

```
    {
      \token_to_str:N \DTLmultibarchart \c_space_tl ~ missing ~ variables
    }
    {
      You ~ need ~ to ~ set ~ the ~ `variables' ~
      key ~ to ~ the ~ list ~ of ~ placeholder ~ commands
    }
  }
  {
    \PackageError { databar }
    {
      \token_to_str:N \DTLmultibarchart \c_space_tl ~ missing ~
      variables ~ (`variable' ~ setting ~ not ~ applicable ~
      for ~ multibar ~ charts)
    }
    {
      Remember ~ that ~ you ~ need ~ to ~ use ~ `variables' ~
      not ~ `variable' ~ with ~ \token_to_str:N \DTLmultibarchart
    }
  }
}
\tl_if_empty:NF \l__databar_yticgap_tl
 {
   \fp_set:Nn \l__databar_yticgap_fp
     { \l__databar_yticgap_tl }
   \fp_compare:nNnT { \l__databar_yticgap_fp } < { \c_zero_fp }
     {
       \PackageError { databar }
       {
         y-tick ~ gap ~ `\tl_to_str:o { \l__databar_yticgap_tl } ' ~
         is ~ negative
       }
       {
         The ~ y ~ tick ~ gap ~ specification ~ evaluates ~ to ~
         a ~ negative ~ number ~ ( \fp_to_decimal:N \l__databar_yticgap_fp) ~
         so ~ the ~ yticgap ~ setting ~ will ~ be ~ ignored
       }
       \tl_clear:N \l__databar_yticgap_tl
       \fp_zero:N \l__databar_yticgap_fp
     }
 }
\__databar_do:n
{
  \cs_set_eq:NN
    \__databar_do_update:
    \__databar_multi_variables_update:
  \cs_set_eq:NN
    \__databar_draw_bars:
    \__databar_variables_draw_bars:
  \tl_set:Nx \DTLbartotalvariables
```

```
      { \seq_count:N \l__databar_variables_seq }
     \__databar_do_multi_chart:n { #4 }
   }
  \group_end:
}
```

Action ‘multibar chart’:

```
\cs_new:cn { __datatool_action_ multibar ~ chart : }
{
 \group_begin:
  \tl_if_empty:NF \l__datatool_action_name_tl
   {
     \tl_set_eq:NN
       \l__datatool_default_dbname_tl
       \l__datatool_action_name_tl
   }
  \bool_set_true:N \l_databar_multi_bars_bool
  \cs_set_eq:NN \__databar_do:n \use:n
  \__datatool_optional_columns:
  \clist_if_empty:NTF \l__datatool_action_options_clist
   {
     \l__databar_pre_init_tl
   }
   {
     \keys_set_groups:nnV { datatool/bar } { pre-parse }
       \l__datatool_action_options_clist
     \l__databar_pre_init_tl
     \keys_set:nV { datatool/bar }
       \l__datatool_action_options_clist
   }
  \l__databar_init_tl
  \seq_if_empty:NTF \l__datatool_action_columns_seq
   {
     \seq_if_empty:NTF \l__databar_variables_seq
      {
        \__datatool_action_error:nn
         {
           missing ~ variables
         }
         {
           You ~ need ~ to ~ either ~ set ~ the ~ `keys' ~
           or ~ `columns' ~ action ~ settings ~ or ~ set ~
           `variables' ~ within ~ the ~ `options' ~
           action ~ setting ~ in ~
           \token_to_str:N \DTLaction [...]{ \l__datatool_action_tl }
         }
        \cs_set_eq:NN \__databar_do:n \cs_none:n
      }
      {
        \tl_set:Nx \DTLbartotalvariables
```

```
          { \seq_count:N \l__databar_variables_seq }
        \cs_set_eq:NN
          \__databar_do_update:
          \__databar_multi_variables_update:
        \cs_set_eq:NN
          \__databar_draw_bars:
          \__databar_variables_draw_bars:
      }
    }
    {
      \tl_set:Nx \DTLbartotalvariables
       { \seq_count:N \l__datatool_action_columns_seq }
      \cs_set_eq:NN
        \__databar_do_update:
        \__databar_multi_columns_update:
      \cs_set_eq:NN
        \__databar_draw_bars:
        \__databar_columns_draw_bars:
    }
   \__databar_do:n
   {
     \exp_args:NV
       \__databar_do_multi_chart:n
       \l__datatool_action_assign_clist
   }
 \group_end:
}

\cs_new:Nn \__databar_do_multi_chart:n
{
 \tl_clear:N \DTLtotalbars
 \__datatool_calc_bar_lengths:n { #1 }
```

Calculate the total number of bar groups if not already found.

```
 \tl_if_empty:NT \DTLtotalbars
  {
    \__databar_filtered_map:nn { #1 } { }
    \tl_set:NV \DTLtotalbars \l__databar_index_int
  }
 \tl_set_eq:NN \DTLtotalbargroups \DTLtotalbars
```

Calculate the overall total number of bars.

```
 \tl_set:Nx \DTLtotalbars
   { \int_eval:n { \DTLtotalbargroups * \DTLbartotalvariables } }
```

Calculate the bar group width, which needs to take into account the bar gap, but first set it to just the total number of bars in a group.

```
 \tl_set:Nx \DTLbargroupwidth
  { \DTLbartotalvariables }
```

Add on the inter bar gaps to the group width.

```
 \tl_if_eq:NnF \l__databar_bargap_tl { 0 }
```

```
  {
    \tl_set:Nx \DTLbargroupwidth
     {
       \fp_eval:n
        {
          \DTLbargroupwidth
            + \l__databar_bargap_tl
               * ( \DTLbargroupwidth - \c_one_fp )
        }
     }
   }
```

Calculate the bar chart width (x-axis length).

```
  \tl_set:Nx \DTLbarchartwidth
   {
     \fp_eval:n
      {
        \DTLtotalbargroups * \DTLbargroupwidth
        + \l__databar_groupgap_tl
          * ( \DTLtotalbargroups - \c_one_fp )
      }
   }
  \__databar_calc_scale:
  \__databar_construct_ytics:
  \__databar_multi_chart:n { #1 }
 }
```

Code that loops through each variable:

```
 \cs_new:Nn \__databar_draw_bars:
 {
   \__databar_variables_draw_bars:
 }
```

Draw bars if variables option used:

```
 \cs_new:Nn \__databar_variables_draw_bars:
 {
   \seq_map_indexed_function:NN
     \l__databar_variables_seq
     \__databar_draw_bar:nn
 }
```

Draw bars if columns or keys action setting used:

```
 \cs_new:Nn \__databar_columns_draw_bars:
 {
   \seq_map_indexed_inline:Nn
    \l__datatool_action_columns_seq
    {
      \exp_args:NNno
        \dtl@getentryfromrow
        \l__databar_action_variable_tl
        { ##2 }
```

```
        \l__datatool_map_data_row_tl
       \__databar_draw_bar:nn
        { ##1 }
        \l__databar_action_variable_tl
    }
}
\cs_new:Nn \__databar_multi_chart:n
{
  \begin{tikzpicture}
```

Set unit vectors

```
  \ifDTLverticalbars
    \pgfsetyvec
     {
       \pgfpoint
        { 0pt } { \fp_to_int:N \l__databar_unit_fp ~ sp }
     }
    \pgfsetxvec
     {
       \pgfpoint { \DTLbarwidth } { 0pt }
     }
  \else
    \pgfsetxvec
     {
       \pgfpoint
        { \fp_to_int:N \l__databar_unit_fp ~ sp } { 0pt }
     }
    \pgfsetyvec
     {
       \pgfpoint { 0pt } { \DTLbarwidth }
     }
  \fi
```

Begin hook

```
  \DTLbaratbegintikz
```

Initialise:

```
  \fp_zero:N \l__databar_start_fp
```

Iterate through data

```
  \__databar_filtered_map:nn { #1 }
   {
```

Initialise bar group label offset.

```
      \dim_zero:N \l__databar_group_label_offset_dim
```

Draw each bar in the group.

```
      \__databar_draw_bars:
```

Do the bar group label, if set.

```
      \tl_if_empty:NF \l__databar_barlabel_tl
       {
```

```
    \dim_add:Nn
      \l__databar_group_label_offset_dim
      { \DTLbarlabeloffset }
    \tl_put_right:Nn \l__dataplot_content_tl
      { \pgftext [ at = \pgfpoint }
    \ifDTLverticalbars
     \tl_put_right:Nx \l__dataplot_content_tl
      {
        {
          \fp_eval:n
           {
             \l__databar_start_fp
                - 0.5 * \DTLbargroupwidth
           }
          \exp_not:N \DTLbarwidth
        }
        {
          - \exp_not:N \l__databar_group_label_offset_dim
        }
      }
    \else
     \tl_put_right:Nx \l__dataplot_content_tl
      {
        {
          - \exp_not:N \l__databar_group_label_offset_dim
        }
        {
          \fp_eval:n
           {
             \l__databar_start_fp
                - 0.5 * \DTLbargroupwidth
           }
          \exp_not:N \DTLbarwidth
        }
      }
    \fi
    \tl_put_right:Nx \l__dataplot_content_tl
      {
        , ~ \DTLbargrouplabelalign ] ~
         {
           \exp_not:N \DTLdisplaybargrouplabel
            { \exp_not:V \l__databar_barlabel_tl }
         }
      }
    \l__dataplot_content_tl
    \tl_clear:N \l__dataplot_content_tl
  }
```

Bar group hook

```
  \DTLeverybargrouphook
```

```
      \fp_add:Nn \l__databar_start_fp { \l__databar_groupgap_tl }
    }
```
Add axes drawing code to content:
```
  \__databar_draw_axes:
```
Add y tick marks if required
```
  \seq_map_indexed_function:NN
    \l__databar_yticpoints_seq
    \__databar_draw_y_tics_fn:nn
```
Do any pending content and clear token list variable.
```
  \l__dataplot_content_tl
  \tl_clear:N \l__dataplot_content_tl
```
Add the y label if required
```
  \__databar_plot_ylabel:
```
Do any pending content and clear token list variable.
```
  \l__dataplot_content_tl
  \tl_clear:N \l__dataplot_content_tl
```
End hook
```
  \DTLbaratendtikz
  \end{tikzpicture}
}
```
Draw all bars in the current group.
```
\cs_new:Nn \__databar_draw_bar:nn
{
```
The bar group index is the same as the value of `\l__databar_index_int`:
```
  \tl_set:Nx \DTLbargroupindex
    { \int_use:N \l__databar_index_int }
  \tl_set:Nn \DTLbarindex { #1 }
```
Set the lower label
```
  \tl_set:Nx \l__databar_lowerbarlabel_tl
   { \seq_item:Nn \l__databar_multibarlabels_seq { #1 } }
  \tl_if_empty:NF \l__databar_lowerbarlabel_tl
   {
     \tl_set:Nx \l__databar_lowerbarlabel_tl
      {
        \exp_not:N \DTLdisplaylowermultibarlabel
         { \exp_not:V \l__databar_lowerbarlabel_tl }
      }
   }
```
Set the upper label
```
  \tl_set:Nx \l__databar_upperbarlabel_tl
   { \seq_item:Nn \l__databar_uppermultibarlabels_seq { #1 } }
  \tl_if_empty:NF \l__databar_upperbarlabel_tl
   {
     \tl_set:Nx \l__databar_upperbarlabel_tl
```

```
      {
        \exp_not:N \DTLdisplayuppermultibarlabel
         { \exp_not:V \l__databar_upperbarlabel_tl }
      }
   }
  \__databar_draw_bar:NnVV #2 { #1 }
    \l__databar_lowerbarlabel_tl
    \l__databar_upperbarlabel_tl
  \fp_add:Nn \l__databar_start_fp { \c_one_fp }
}
```

Update group offset.

```
\cs_new:Nn \__databar_update_group_offset:
{
  \ifDTLverticalbars
    \dim_compare:nNnT
      { \pgf@pathminy } < { \c_zero_dim }
     {
       \dim_set:Nn
         \l__databar_group_label_offset_dim
        {
          \dim_max:nn
           { \l__databar_group_label_offset_dim }
           { \dim_abs:n { \pgf@pathminy} }
        }
     }
  \else
    \dim_compare:nNnT
      { \pgf@pathminx } < { \c_zero_dim }
     {
       \dim_set:Nn
         \l__databar_group_label_offset_dim
        {
          \dim_max:nn
           { \l__databar_group_label_offset_dim }
           { \dim_abs:n { \pgf@pathminx} }
        }
     }
  \fi
}
\ExplSyntaxOff
```

27 datapie.sty

Declare package:

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{datapie-2019-09-27.sty}
```

```
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{datapie}[2026/07/31 v3.4.4 (NLCT)]
```

Version 3.0: no longer using xkeyval.

`\ifDTLcolorpiechart` The conditional `\ifDTLcolorpiechart` is to determine whether to use colour or grey scale. NB may be removed or deprecated.

```
\newif\ifDTLcolorpiechart
\DTLcolorpiecharttrue
```

`\ifDTLrotateinner` Define boolean keys to govern label rotations.

```
\newif\ifDTLrotateinner
\DTLrotateinnerfalse
```

`\ifDTLrotateouter`

```
\newif\ifDTLrotateouter
\DTLrotateouterfalse
```

The package options have been changed to use l3keys.

Define package options:

```
\ExplSyntaxOn
\keys_define:nn { datatool }
{
  color .legacy_if_set:n = DTLcolorpiechart,
  gray .legacy_if_set_inverse:n = DTLcolorpiechart,
  rotateinner .legacy_if_set:n = DTLrotateinner,
  norotateinner .legacy_if_set_inverse:n = DTLrotateinner,
  rotateouter .legacy_if_set:n = DTLrotateouter,
  norotateouter .legacy_if_set_inverse:n = DTLrotateouter,
}
\ExplSyntaxOff
```

Process options:

```
\IfPackageLoadedTF{datatool}
{
  \ProcessKeyOptions[datatool]
}
{
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datatool}}
  \ProcessOptions
}
\RequirePackage{datatool}
```

Remove the package option keys so they can't be used with `\DTLsetup` (otherwise they may conflict with databar etc).

```
\ExplSyntaxOn
\keys_define:nn { datatool }
{
  color .undefine: ,
```

```
  gray .undefine: ,
  rotateinner .undefine: ,
  norotateinner .undefine: ,
  rotateouter .undefine: ,
  norotateouter .undefine: ,
}
\ExplSyntaxOff
```

Required packages:

```
\RequirePackage{tikz}
```

Define some variables that govern the appearance of the pie chart.

```
\ExplSyntaxOn
```

The pie chart variable.

```
\tl_new:N \DTLpievariable
```

`\DTLradius` The radius of the pie chart is given by `\DTLradius`.

```
\newlength\DTLradius
\DTLradius=2cm
```

`\DTLinnerratio` The inner label offset ratio is given by `\DTLinnerratio`

```
\newcommand*{\DTLinnerratio}{0.5}
```

`\DTLouterratio` The outer label offset ratio is given by `\DTLouterratio`.

```
\newcommand*{\DTLouterratio}{1.25}
```

`\DTLcutawayratio` The cutaway offset ratio is given by `\DTLcutawayratio`.

```
\newcommand*\DTLcutawayratio{0.2}
```

`\DTLstartangle` The angle of the first segment is given by `\DTLstartangle`.

```
\newcommand*{\DTLstartangle}{0}
```

`\dtl@inneroffset` Version 3.0: replaced `\dtl@inneroffset` with:

```
\dim_new:N \l__datapie_inner_offset_dim
\dim_set:Nn \l__datapie_inner_offset_dim
 { \DTLinnerratio \DTLradius }
```

`\dtl@outeroffset` Version 3.0: replaced `\dtl@outeroffset` with:

```
\dim_new:N \l__datapie_outer_offset_dim
\dim_set:Nn \l__datapie_outer_offset_dim
 { \DTLouterratio \DTLradius }
```

`\dtl@cutawayoffset` Version 3.0: replaced `\dtl@cutawayoffset` with:

```
\dim_new:N \l__datapie_cutaway_offset_dim
\dim_set:Nn \l__datapie_cutaway_offset_dim
 { \DTLcutawayratio \DTLradius }
```

\dtl@piecutaways A comma separated list of segments that need to be cut away from the pie chart. This now uses a clist variable.

```
\clist_new:N \l__datapie_cutaways_clist
```

\dtl@innerlabel `\dtl@innerlabel` specifies the label to appear inside the segment. By default this is the variable used to create the pie chart.

Version 3.0: renamed `\l__datatool_pie_inner_label_tl`

```
\tl_new:N \l__datapie_inner_label_tl
\tl_set:Nn \l__datapie_inner_label_tl { \DTLpievariable }
```

\dtl@outerlabel Version 3.0: renamed `\l__datatool_pie_outer_label_tl`

```
\tl_new:N \l__datapie_outer_label_tl
```

DTLpieroundvar DTLpieroundvar is a counter governing the number of digits to round to (for display).

```
\newcounter{DTLpieroundvar}
\setcounter{DTLpieroundvar}{1}
```

\DTLdisplayinnerlabel

```
\DTLdisplayinnerlabel{⟨label⟩}
```

This is used to format the inner label. This just does the label by default.

```
\newcommand*{\DTLdisplayinnerlabel}[1]{#1}
```

\DTLdisplayouterlabel

```
\DTLdisplayouterlabel{⟨label⟩}
```

This is used to format the outer label. This just does the label by default.

```
\newcommand*{\DTLdisplayouterlabel}[1]{#1}
```

No-op command:

```
\cs_new:Nn \__datapie_noop:N
{
  \PackageError { datapie }
  {
    Can't ~ use ~ \token_to_str:N #1 \c_space_tl ~ outside ~
    \token_to_str:N \DTLpiechart
  }
  {
    Certain ~ datapie.sty ~ commands ~ may ~ only ~ be ~ used ~
    within ~ pie ~ chart ~ hooks ~ or ~ option ~ values
  }
}
```

\DTLpiepercent `\DTLpiepercent` returns the percentage value of the current segment.

```
\newcommand*{\DTLpiepercent}{%
  \__datapie_noop:N \DTLpiepercent
}
```

```
\cs_new:Nn \__datapie_percent:
{
  \use:c
   {
     dtl@piepercent@
     \romannumeral \l__datapie_segment_int
   }
}
```

\DTLpieatsegment

```
\DTLpieatsegment{⟨segment index⟩}{⟨total⟩}{⟨start angle⟩}{⟨mid angle⟩}{⟨end angle⟩}{⟨shift angle⟩}{⟨shift offset⟩}{⟨inner offset⟩}{⟨outer offset⟩}
```

Hook at each segment.

```
\newcommand*{\DTLpieatsegment}[9]{}
```

\DTLpieatbegintikz `\DTLpieatbegintikz` specifies any commands to apply at the start of the tikzpicture environment. By default it does nothing.

```
\newcommand*{\DTLpieatbegintikz}{}
```

\DTLpieatendtikz `\DTLpieatendtikz` specifies any commands to apply at the end of the tikzpicture environment. By default it does nothing.

```
\newcommand*{\DTLpieatendtikz}{}
```

\DTLsetpiesegmentcolor

```
\DTLsetpiesegmentcolor{⟨n⟩}{⟨color⟩}
```

Assign colour name ⟨*color*⟩ to the ⟨*n*⟩th segment.

```
\newcommand*{\DTLsetpiesegmentcolor}[2]{%
 \tl_set:cn { dtlpie@segcol\romannumeral#1 } { #2 }
}
```

\DTLgetpiesegmentcolor

```
\DTLgetpiesegmentcolor{⟨n⟩}
```

Get the colour specification for segment ⟨*n*⟩

```
\newcommand*{\DTLgetpiesegmentcolor}[1]{%
 \cs_if_exist_use:cF { dtlpie@segcol\romannumeral#1 } { white }
}
```

\DTLdopiesegmentcolor

```
\DTLdopiesegmentcolor{⟨n⟩}
```

Set the colour to that for segment ⟨*n*⟩

```
\newcommand*{\DTLdopiesegmentcolor}[1]{
```

```
    \cs_if_exist:cTF { dtlpie@segcol\romannumeral#1 }
    {
      \exp_args:Nc \color { dtlpie@segcol\romannumeral#1 }
    }
    {
      \PackageWarning{datapie}{No ~ colour ~ assigned ~ to ~ segment ~ \number#1}
    }
  }
```

docurrentpiesegmentcolor `\DTLdocurrentpiesegmentcolor` sets the colour to that of the current segment. Allow it to also be used within `\DTLmapdata` or `\DTLforeach`.

```
\NewDocumentCommand \DTLdocurrentpiesegmentcolor { }
{
  \int_compare:nNnTF
    { \l__datapie_segment_int } > { \c_zero_int }
  {
    \exp_args:NV \DTLdopiesegmentcolor \l__datapie_segment_int
  }
  {
    \int_compare:nNnTF
      { \l__datatool_row_idx_int } > { \c_zero_int }
    {
      \exp_args:NV \DTLdopiesegmentcolor \l__datatool_row_idx_int
    }
    {
      \int_compare:nNnTF
        { \dtlforeachlevel } > { \c_zero_int }
      {
        \exp_args:Nv \DTLdopiesegmentcolor { c@DTLrow\romannumeral\dtlforeachlevel }
      }
      {
        \__datapie_noop:N \DTLdocurrentpiesegmentcolor
      }
    }
  }
}
```

\DTLpieoutlinecolor `\DTLpieoutlinecolor` specifies what colour to draw the outline.

```
\newcommand*{\DTLpieoutlinecolor}{black}
```

\DTLpieoutlinewidth `\DTLpieoutlinewidth` specifies the line width of the outline: Outline is only drawn if the linewidth is greater than 0pt.

```
\newlength\DTLpieoutlinewidth
\DTLpieoutlinewidth=0pt
```

Set the default colours. If there are more than eight segments, more colours will need to be defined.

```
\ifDTLcolorpiechart
 \DTLsetpiesegmentcolor{1}{red}
```

```
  \DTLsetpiesegmentcolor{2}{green}
  \DTLsetpiesegmentcolor{3}{blue}
  \DTLsetpiesegmentcolor{4}{yellow}
  \DTLsetpiesegmentcolor{5}{magenta}
  \DTLsetpiesegmentcolor{6}{cyan}
  \DTLsetpiesegmentcolor{7}{orange}
  \DTLsetpiesegmentcolor{8}{white}
 \else
  \DTLsetpiesegmentcolor{1}{black!15}
  \DTLsetpiesegmentcolor{2}{black!25}
  \DTLsetpiesegmentcolor{3}{black!35}
  \DTLsetpiesegmentcolor{4}{black!45}
  \DTLsetpiesegmentcolor{5}{black!55}
  \DTLsetpiesegmentcolor{6}{black!65}
  \DTLsetpiesegmentcolor{7}{black!75}
  \DTLsetpiesegmentcolor{8}{black!85}
 \fi
```

Provide key=value for `pie` option in `\DTLsetup`

```
\keys_define:nn { datatool/pie }
{
  rotateinner .legacy_if_set:n = DTLrotateinner,
  rotate-inner .legacy_if_set:n = DTLrotateinner,
  rotateouter .legacy_if_set:n = DTLrotateouter,
  rotate-outer .legacy_if_set:n = DTLrotateouter,
```

Inner ratio:

```
  innerratio .code:n =
   {
     \tl_set:Nn \DTLinnerratio { #1 }
     \dim_set:Nn \l__datapie_inner_offset_dim
       { \DTLinnerratio \DTLradius }
   },
  inner-ratio .code:n =
   {
     \tl_set:Nn \DTLinnerratio { #1 }
     \dim_set:Nn \l__datapie_inner_offset_dim
       { \DTLinnerratio \DTLradius }
   },
```

Outer ratio:

```
  outerratio .code:n =
   {
     \tl_set:Nn \DTLouterratio { #1 }
     \dim_set:Nn \l__datapie_outer_offset_dim
       { \DTLouterratio \DTLradius }
   },
  outer-ratio .code:n =
   {
     \tl_set:Nn \DTLouterratio { #1 }
     \dim_set:Nn \l__datapie_outer_offset_dim
```

```
      { \DTLouterratio \DTLradius }
  },
```

Cutaway ratio:

```
cutawayratio .code:n =
 {
   \tl_set:Nn \DTLcutawayratio { #1 }
   \dim_set:Nn \l__datapie_cutaway_offset_dim
     { \DTLcutawayratio \DTLradius }
 },
cutaway-ratio .code:n =
 {
   \tl_set:Nn \DTLcutawayratio { #1 }
   \dim_set:Nn \l__datapie_cutaway_offset_dim
     { \DTLcutawayratio \DTLradius }
 },
```

Sets the inner offset as an absolute value (not dependent on the radius):

```
inneroffset .dim_set:N = \l__datapie_inner_offset_dim ,
inner-offset .dim_set:N = \l__datapie_inner_offset_dim ,
```

Sets the outer offset as an absolute value (not dependent on the radius):

```
outeroffset .dim_set:N = \l__datapie_outer_offset_dim ,
outer-offset .dim_set:N = \l__datapie_outer_offset_dim ,
```

Sets the cutaway offset as an absolute value (not dependent on the radius):

```
cutawayoffset .dim_set:N = \l__datapie_cutaway_offset_dim ,
cutaway-offset .dim_set:N = \l__datapie_cutaway_offset_dim ,
```

Starting angle:

```
start-angle .tl_set:N = \DTLstartangle,
start .tl_set:N = \DTLstartangle,
```

Set the radius of the pie chart.

```
radius .code:n =
 {
   \dim_set:Nn \DTLradius { #1 }
   \dim_set:Nn \l__datapie_inner_offset_dim
     { \DTLinnerratio \DTLradius }
   \dim_set:Nn \l__datapie_outer_offset_dim
     { \DTLouterratio \DTLradius }
   \dim_set:Nn \l__datapie_cutaway_offset_dim
     { \DTLcutawayratio \DTLradius }
 },
```

Set the radius of the pie chart without adjusting the offsets.

```
radius* .dim_set:N = \DTLradius ,
```

Rounding:

```
round .int_set:N = \c@DTLpieroundvar,
segment-colors .code:n =
 {
   \seq_set_from_clist:Nn \l__datatool_tmp_seq { #1 }
```

```
    \seq_map_indexed_function:NN
      \l__datatool_tmp_seq
      \DTLsetpiesegmentcolor
  },
 segment-default-colors .code:n =
  {
    \DTLcolorpiecharttrue
    \DTLsetpiesegmentcolor{1}{red}
    \DTLsetpiesegmentcolor{2}{green}
    \DTLsetpiesegmentcolor{3}{blue}
    \DTLsetpiesegmentcolor{4}{yellow}
    \DTLsetpiesegmentcolor{5}{magenta}
    \DTLsetpiesegmentcolor{6}{cyan}
    \DTLsetpiesegmentcolor{7}{orange}
    \DTLsetpiesegmentcolor{8}{white}
  },
 segment-default-colors .value_forbidden:n = true,
 segment-default-gray .code:n =
  {
    \DTLcolorpiechartfalse
    \DTLsetpiesegmentcolor{1}{black!15}
    \DTLsetpiesegmentcolor{2}{black!25}
    \DTLsetpiesegmentcolor{3}{black!35}
    \DTLsetpiesegmentcolor{4}{black!45}
    \DTLsetpiesegmentcolor{5}{black!55}
    \DTLsetpiesegmentcolor{6}{black!65}
    \DTLsetpiesegmentcolor{7}{black!75}
    \DTLsetpiesegmentcolor{8}{black!85}
  },
 segment-default-gray .value_forbidden:n = true,
 outline-color .tl_set:N = \DTLpieoutlinecolor,
 outline-width .dim_set:N = \DTLpieoutlinewidth,
```

List of cut away segments.

```
 cutaway .clist_set:N = \l__datapie_cutaways_clist,
```

Variable used to create the pie chart. (Must be a control sequence.)

```
 variable .tl_set:N = \DTLpievariable,
```

Inner label:

```
 innerlabel .tl_set:N = \l__datapie_inner_label_tl,
 inner-label .tl_set:N = \l__datapie_inner_label_tl,
```

Outer label:

```
 outerlabel .tl_set:N = \l__datapie_outer_label_tl,
 outer-label .tl_set:N = \l__datapie_outer_label_tl,
```

Filter:

```
 include-if .cs_set:Np = \__datapie_filter:T #1,
 include-if-fn .code:n =
  {
    \cs_set_eq:NN \__datapie_filter:T #1
```

```
    },
```

Deprecated experimental setting. TODO remove

```
   condition .code:n =
    {
      \PackageWarning{datapie}{Deprecated ~ option ~ `condition'. ~
       Use ~ `include-if' ~ instead}
      \cs_set:Nn = \__datapie_filter:T { #1 }
    }
 }
```

Allow these keys to be set in `\DTLsetup{pie={...}}`

```
\keys_define:nn { datatool }
{
  pie .code:n = { \keys_set:nn { datatool/pie } { #1 } }
}
```

Provide a filter function:

```
\cs_new:Nn \__datapie_filter:T { #1 }
```

```
\cs_new:Nn \__datapie_filtered_map:nn
{
  \DTLmapdata
  {
    \DTLmapgetvalues { #1 }
    \__datapie_filter:T { #2 }
  }
}
```

Floating point variables.

```
\fp_new:N \l__datapie_x_fp
\fp_new:N \l__datapie_total_fp
\fp_new:N \l__datapie_start_fp
\fp_new:N \l__datapie_end_fp
\fp_new:N \l__datapie_extent_fp
\fp_new:N \l__datapie_angle_fp
\fp_new:N \l__datapie_mid_fp
```

Constants:

```
\fp_const:Nn \c_datapie_whole_circle_fp { 360 }
\fp_const:Nn \c_datapie_half_circle_fp { 180 }
\fp_const:Nn \c_datapie_minus_half_circle_fp { -180 }
\fp_const:Nn \c_datapie_quarter_circle_fp { 90 }
\fp_const:Nn \c_datapie_three_quarter_circle_fp { 270 }
\fp_const:Nn \c_datapie_minus_quarter_circle_fp { -90 }
```

Define integer variable to keep track of segments

`\@dtl@seg` Version 3.0 replaced `\@dtl@seg` with:

```
\int_new:N \l__datapie_segment_int
```

Token lists containing values to write to content token list.

`\@dtl@start` Version 3.0: replaced `\@dtl@start` with:
```
\tl_new:N \l__datapie_start_tl
```

`\dtl@midangle` Version 3.0: replaced `\dtl@midangle` with:
```
\tl_new:N \l__datapie_mid_tl
```

`\dtl@endangle` Version 3.0: replaced `\dtl@endangle` with:
```
\tl_new:N \l__datapie_end_tl
```

`\dtl@extent` Version 3.0: replaced `\dtl@extent` with:
```
\tl_new:N \l__datapie_extent_tl
```

`\dtl@angle` Version 3.0: replaced `\dtl@angle` with:
```
\tl_new:N \l__datapie_angle_tl
```

`\dtl@cutlen` Version 3.0: replaced `\dtl@cutlen` with:
```
\tl_new:N \l__datapie_cut_length_tl
```

`\dtl@innernodeopt` Version 3.0: replaced `\dtl@innernodeopt` with:
```
\tl_new:N \l__datapie_inner_node_opt_tl
```

`\dtl@outernodeopt` Version 3.0: replaced `\dtl@outernodeopt` with:
```
\tl_new:N \l__datapie_outer_node_opt_tl
```

Calculate the total (filtered column sum).
```
\cs_new:Nn \__datapie_compute_total:n
 {
   \fp_zero:N \l__datapie_total_fp
   \__datapie_filtered_map:nn { #1 }
    {
      \datatool_set_fp:NV \l__datapie_x_fp \DTLpievariable
      \fp_add:Nn \l__datapie_total_fp { \l__datapie_x_fp }
    }
 }
```
Calculate the angles for each segment.
```
\cs_new:Nn \__datapie_compute_angles:n
 {
   \datatool_set_fp:NV \l__datapie_start_fp \DTLstartangle
   \tl_set:Nn \l__datapie_start_tl
     { \fp_to_decimal:N \l__datapie_start_fp }
   \int_zero:N \l__datapie_segment_int
   \__datapie_filtered_map:nn { #1 }
    {
      \int_incr:N \l__datapie_segment_int
      \datatool_set_fp:NV
        \l__datapie_x_fp \DTLpievariable
      \__datapie_compute_angles:nN
        { \l__datapie_segment_int }
```

```
        \l__datapie_x_fp
      \fp_set:Nn \l__datapie_x_fp
       {
         round
         (
           ( 100 * \l__datapie_x_fp ) / \l__datapie_total_fp,
           \c@DTLpieroundvar
         )
       }
      \tl_set:cx
       {
         dtl@piepercent@
         \romannumeral \l__datapie_segment_int
       }
       { \fp_to_decimal:N \l__datapie_x_fp }
     \datatool_pad_trailing_zeros:cn
       {
         dtl@piepercent@
         \romannumeral \l__datapie_segment_int
       }
       { \c@DTLpieroundvar }
   }
 }
```

Compute angles for segment. Syntax: {⟨*segment index*⟩}⟨*x-var*⟩

```
\cs_new:Nn \__datapie_compute_angles:nN
 {
   \fp_compare:nNnTF
     { \l__datapie_start_fp } > { \c_datapie_half_circle_fp }
    {
```

if startangle > 180

```
      \fp_sub:Nn \l__datapie_start_fp
        { \c_datapie_whole_circle_fp }
    }
    {
      \fp_compare:nNnT
        { \l__datapie_start_fp } < { \c_datapie_minus_half_circle_fp }
      {
```

if startangle < -180

```
        \fp_add:Nn \l__datapie_start_fp
          { \c_datapie_whole_circle_fp }
      }
    }
   \tl_set:cx
     { dtl@sang@ \romannumeral #1 }
     { \fp_to_decimal:N \l__datapie_start_fp }
   \fp_set:Nn \l__datapie_tmpa_fp
     { ( #2 * \c_datapie_whole_circle_fp ) / \l__datapie_total_fp }
   \tl_set:cx
```

```
      { dtl@angle@ \romannumeral #1 }
      { \fp_to_decimal:N \l__datapie_tmpa_fp }
    \fp_add:Nn \l__datapie_start_fp
      { \l__datapie_tmpa_fp }
    \tl_set:cx { dtl@cut@angle@ \romannumeral #1 } { 0 }
    \tl_set:cx { dtl@cut@len@ \romannumeral #1 } { 0cm }
    \tl_set:Nx \l__datapie_start_tl
      { \fp_to_decimal:N \l__datapie_start_fp }
  }
```

The new version of tikz doesn't seem to like command names in certain contexts, so use a token list variable to build up the content of the tikzpicture.

```
\tl_new:N \l__datapie_content_tl
\tl_new:N \l__datapie_hook_tl
```

This has only partially been converted to LaTeX3:

```
\cs_new:Nn \__datapie_draw:n
{
```

Set the starting angle

```
  \tl_set:Nx \l__datapie_start_tl { \DTLstartangle }
  \int_zero:N \l__datapie_segment_int
  \int_zero:N \l__datatool_row_idx_int
  \begin{tikzpicture}
  \DTLpieatbegintikz
  \__datapie_filtered_map:nn { #1 }
   {
```

Clear the content token list used for this segment.

```
      \tl_clear:N \l__datapie_content_tl
      \int_incr:N \l__datapie_segment_int
```

Set the start angle.

```
      \tl_set:Nx \l__datapie_start_tl
       {
         \tl_use:c
           {
             dtl@sang@ \romannumeral \l__datapie_segment_int
           }
       }
      \fp_set:Nn \l__datapie_start_fp
        { \l__datapie_start_tl }
```

Set the extent

```
      \tl_set:Nx \l__datapie_extent_tl
        {
          \tl_use:c
            {
              dtl@angle@
              \romannumeral \l__datapie_segment_int
            }
        }
```

```
\fp_set:Nn \l__datapie_extent_fp
  { \l__datapie_extent_tl }
```

Compute the end angle

```
\fp_set:Nn \l__datapie_end_fp
 { \l__datapie_start_fp + \l__datapie_extent_fp }
\tl_set:Nx \l__datapie_end_tl
  { \fp_to_decimal:N \l__datapie_end_fp }
```

Compute the shift.

```
\tl_set:Nx \l__datapie_angle_tl
  {
    \tl_use:c
     {
       dtl@cut@angle@
       \romannumeral \l__datapie_segment_int
     }
  }
\fp_set:Nn \l__datapie_angle_fp
  { \l__datapie_angle_tl }
\fp_compare:nNnT
  { \l__datapie_angle_fp } > { \c_datapie_half_circle_fp }
 {
   \fp_sub:Nn \l__datapie_angle_fp
     { \c_datapie_whole_circle_fp }
   \tl_set:Nx \l__datapie_angle_tl
     { \fp_to_decimal:N \l__datapie_angle_fp }
 }
\tl_set:Nx \l__datapie_cut_length_tl
 {
   \tl_use:c
    {
      dtl@cut@len@
      \romannumeral \l__datapie_segment_int
    }
 }
\tl_put_right:Nn \l__datapie_content_tl
  { \begin { scope} }
\tl_set:Nx \l__datapie_hook_tl
  { { \l__datapie_angle_tl } { \l__datapie_cut_length_tl } }
\tl_put_right:Nx \l__datapie_content_tl
  {
    [ shift =
      {
        (
          \l__datapie_angle_tl \c_colon_str
          \l__datapie_cut_length_tl
        )
      } ]
  }
```

Compute the mid way angle.

```
\fp_set:Nn \l__datapie_mid_fp
  { 0.5 * \l__datapie_extent_fp + \l__datapie_start_fp }
\tl_set:Nx \l__datapie_mid_tl
  { \fp_to_decimal:N \l__datapie_mid_fp }
```

Draw the segment.

```
\tl_put_right:Nn \l__datapie_content_tl { \fill }
\tl_put_right:Nx \l__datapie_content_tl
{
  [color= \DTLgetpiesegmentcolor \l__datapie_segment_int ]
   (0,0) ~ -- ~
    ( \l__datapie_start_tl \c_colon_str \DTLradius) ~
   arc
    (
      \l__datapie_start_tl \c_colon_str
      \l__datapie_end_tl \c_colon_str
      \DTLradius
    ) ~
   -- ~ cycle;
}
```

Draw the outline if required:

```
\dim_compare:nNnT
  { \DTLpieoutlinewidth } > { \c_zero_dim }
 {
   \tl_put_right:Nn \l__datapie_content_tl { \draw }
   \tl_put_right:Nx \l__datapie_content_tl
   {
     [color=\DTLpieoutlinecolor,
         line ~ width=\DTLpieoutlinewidth]
     (0,0) ~ -- ~
       (
         \l__datapie_start_tl \c_colon_str \DTLradius
       ) ~
     arc
      (
        \l__datapie_start_tl \c_colon_str
        \l__datapie_end_tl \c_colon_str
        \DTLradius
      ) ~
     -- ~ cycle;
   }
 }
```

Continue constructing segment hook:

```
\tl_put_left:Nx \l__datapie_hook_tl
 {
    \exp_not:N \DTLpieatsegment
    { \int_use:N \l__datapie_segment_int }
    { \fp_to_decimal:N \l__datapie_total_fp }
```

```
      { \l__datapie_start_tl }
      { \l__datapie_mid_tl }
      { \l__datapie_end_tl }
  }
```

Determine whether to rotate inner labels

```
  \legacy_if:nTF { DTLrotateinner }
  {
```

If the mid way angle is between 90 and 270, the text will look upside-down, so adjust accordingly.

```
    \fp_compare:nTF
      {
        \c_datapie_quarter_circle_fp
        < \l__datapie_mid_fp
        < \c_datapie_three_quarter_circle_fp
      }
     {
       \cs_set_eq:NN \__datapie_next:nn \use_i:nn
     }
     {
      \fp_compare:nNnTF
        { \l__datapie_mid_fp }
         < { \c_datapie_minus_quarter_circle_fp }
       {
         \cs_set_eq:NN \__datapie_next:nn \use_i:nn
       }
       {
         \cs_set_eq:NN \__datapie_next:nn \use_ii:nn
       }
     }
    \__datapie_next:nn
     {
      \tl_set:Nx \l__datapie_inner_node_opt_tl
        {
          anchor = east ,
          rotate =
           \fp_eval:n
             { \l__datapie_mid_fp - \c_datapie_half_circle_fp }
        }
     }
     {
       \tl_set:Nx \l__datapie_inner_node_opt_tl
        {
          anchor = west ,
          rotate = \l__datapie_mid_tl
        }
     }
```

Don’t rotate inner labels

```
  }
```

```
    {
      \tl_set:Nn \l__datapie_inner_node_opt_tl
        { anchor = center }
    }
```
Determine whether to rotate outer labels
```
  \legacy_if:nTF { DTLrotateouter }
   {
```
If the mid way angle is between 90 and 270, the text will look upside-down, so adjust accordingly.
```
    \fp_compare:nTF
     {
       \c_datapie_quarter_circle_fp
        < \l__datapie_mid_fp
        < \c_datapie_three_quarter_circle_fp
     }
    {
       \cs_set_eq:NN \__datapie_next:nn \use_i:nn
    }
    {
      \fp_compare:nNnTF
       { \l__datapie_mid_fp } < { \c_datapie_minus_quarter_circle_fp }
       {
         \cs_set_eq:NN \__datapie_next:nn \use_i:nn
       }
       {
         \cs_set_eq:NN \__datapie_next:nn \use_ii:nn
       }
    }
    \__datapie_next:nn
    {
       \tl_set:Nx \l__datapie_outer_node_opt_tl
        {
          anchor = east ,
          rotate =
           \fp_eval:n
            { \l__datapie_mid_fp - \c_datapie_half_circle_fp }
        }
    }
    {
      \tl_set:Nx \l__datapie_outer_node_opt_tl
       {
         anchor = west ,
         rotate = \l__datapie_mid_tl
       }
    }
  }
  {
```
Don’t rotate outer labels If ($\theta > -45$ and $\theta < 45$) or $\theta = 45$ or $\theta > 315$

```
      \fp_compare:nNnTF
        { \l__datapie_mid_fp } > { 315 }
      {
        \cs_set_eq:NN \__datapie_next:nn \use_i:nn
      }
      {
        \fp_compare:nTF
          { -45 < \l__datapie_mid_fp <= 45 }
         {
           \cs_set_eq:NN \__datapie_next:nn \use_i:nn
         }
         {
           \cs_set_eq:NN \__datapie_next:nn \use_ii:nn
         }
      }
     \__datapie_next:nn
      {
```

East quadrant

```
        \tl_set:Nn \l__datapie_outer_node_opt_tl
         { anchor = west }
      }
      {
        \fp_compare:nTF
          { 45 < \l__datapie_mid_fp <= 135 }
         {
           \cs_set_eq:NN \__datapie_next:nn \use_i:nn
         }
         {
           \cs_set_eq:NN \__datapie_next:nn \use_ii:nn
         }
        \__datapie_next:nn
         {
```

North quadrant

```
          \tl_set:Nn \l__datapie_outer_node_opt_tl
            { anchor = south }
         }
         {
```

If ($\theta > 135$ and $\theta < 225$) or $\theta = 225$ or $\theta = -135$ or $\theta < -135$

```
          \fp_compare:nTF
            { 135 <= \l__datapie_mid_fp <= 225 }
           {
             \cs_set_eq:NN \__datapie_next:nn \use_i:nn
           }
           {
              \fp_compare:nNnTF
               { \l__datapie_mid_fp } > { -135 }
                {
                  \cs_set_eq:NN \__datapie_next:nn \use_ii:nn
```

```
                }
                {
                  \cs_set_eq:NN \__datapie_next:nn \use_i:nn
                }
              }
            }
          \__datapie_next:nn
          {
```

West quadrant

```
            \tl_set:Nn \l__datapie_outer_node_opt_tl
             { anchor = east }
          }
          {
            \tl_set:Nn \l__datapie_outer_node_opt_tl
             { anchor = north }
          }
      }
    }
```

Inner label drawing code:

```
\tl_put_right:Nx \l__datapie_content_tl
{
  \exp_not:N \draw
  (
    \l__datapie_mid_tl
    \c_colon_str
    \dim_use:N \l__datapie_inner_offset_dim
  ) ~
  node [ \l__datapie_inner_node_opt_tl ] ~
  {
    \exp_not:N \DTLdisplayinnerlabel
     { \exp_not:V \l__datapie_inner_label_tl }
  };
}
```

Outer label drawing code:

```
\tl_put_right:Nx \l__datapie_content_tl
{
  \exp_not:N \draw
  (
    \l__datapie_mid_tl \c_colon_str
    \dim_use:N \l__datapie_outer_offset_dim
  ) ~
  node [ \l__datapie_outer_node_opt_tl ] ~
  {
    \exp_not:N \DTLdisplayouterlabel
     { \exp_not:V \l__datapie_outer_label_tl }
  };
}
\tl_put_right:Nn \l__datapie_hook_tl
```

```
      {
        { \l__datapie_inner_offset_dim }
        { \l__datapie_outer_offset_dim }
      }
    \tl_put_right:Nn \l__datapie_content_tl
      {
        \l__datapie_hook_tl
        \end { scope }
      }
    \l__datapie_content_tl
  }
  \int_zero:N \l__datapie_segment_int
  \int_zero:N \l__datatool_row_idx_int
  \DTLpieatendtikz
  \end{tikzpicture}
}
```

\DTLpiechart

\DTLpiechart[⟨*conditions*⟩]{⟨*option list*⟩}{⟨*db name*⟩}{⟨*assign list*⟩}

Make a pie chart from data given in data base ⟨*db name*⟩, where ⟨*assign list*⟩ is a comma-separated list of ⟨*cmd*⟩=⟨*key*⟩ pairs. ⟨*option list*⟩ must include `variable=`⟨*cmd*⟩, where ⟨*cmd*⟩ is included in ⟨*assign list*⟩. The optional argument ⟨*conditions*⟩ is the same as that for `\DTLforeach`.

Version 3.0 now uses `\DTLmapdata`. The condition can now be passed as an argument in ⟨*option list*⟩ but the optional argument is retained for backward compatibility. The condition in the option list will override the optional argument.

```
\NewDocumentCommand \DTLpiechart { o m m m }
{
  \group_begin:
   \IfValueT { #1 }
   {
     \cs_set:Nn \__datapie_filter:T
     {
       \ifthenelse { #1 } { ##1 } { }
     }
   }
   \tl_if_blank:nF { #3 }
    {
      \DTLsetup { default-name = { #3 } }
    }
   \keys_set:nn { datatool/pie } { #2 }
   \tl_if_empty:NTF \DTLpievariable
   {
     \PackageError { datapie }
     {
       \token_to_str:N \DTLpiechart \c_space_tl ~ missing ~ variable
```

```
      }
      {
        You ~ need ~ to ~ set ~ the ~ `variable' ~ key ~ in ~ the ~
        first ~ mandatory ~ argument ~ of ~
        \token_to_str:N \DTLpiechart \c_space_tl ~ to ~ the ~
        applicable ~ placeholder ~ command ~ listed ~ in ~ the ~
        final ~ argument
      }
    }
    {
      \__datapie_do_chart:n { #4 }
    }
  \group_end:
}
```

Do the chart. The argument is the assignment list.

```
\cs_new:Nn \__datapie_do_chart:n
{
```

Enable commands that may only be used within `\DTLpiechart`:

```
  \cs_set_eq:NN \DTLpiepercent \__datapie_percent:
```

Compute the total. This needs to take into account number formatting.

```
  \__datapie_compute_total:n { #1 }
  \__datapie_compute_angles:n { #1 }
```

Compute the offsets for each cut away segment

```
  \clist_map_inline:Nn \l__datapie_cutaways_clist
   {
     \__datapie_set_offset:nw ##1 - \q_stop
   }
  \__datapie_draw:n { #1 }
}
```

Scratch placeholder for action if key or column given instead of variable:

```
\tl_new:N \l__datapie_action_variable_tl
```

Action ‘pie chart’:

```
\cs_new:cn { __datatool_action_ pie ~ chart : }
{
 \group_begin:
   \tl_if_empty:NF \l__datatool_action_name_tl
    {
      \tl_set_eq:NN
        \l__datatool_default_dbname_tl
        \l__datatool_action_name_tl
    }
   \cs_set_eq:NN \__datapie_do:n \use:n
```

If the ‘key’ or ‘column’ has been set, then that provides the variable, unless it’s overridden by the variable setting in the options.

```
    \__datatool_optional_key_xor_column_get_key:nTF
     {
```

No key or column. The variable will need to be provided in the options list instead (or may have already been set with \DTLsetup).

```
    }
    {
```

Key or column provided.

```
      \tl_set:Nn  \DTLpievariable
        { \l__datapie_action_variable_tl }
      \clist_put_right:Nx \l__datatool_action_assign_clist
       {
         \exp_not:N \l__datapie_action_variable_tl =
          \l__datatool_action_key_tl
       }
    }
    {
```

Invalid syntax.

```
      \cs_set_eq:NN \__datapie_do:n \cs_none:n
    }
   \__datapie_do:n
    {
```

Parse options if set.

```
     \clist_if_empty:NF \l__datatool_action_options_clist
      {
        \keys_set:nV { datatool/pie }
          \l__datatool_action_options_clist
      }
```

Check variable has been set.

```
     \tl_if_empty:NT \DTLpievariable
      {
        \cs_set_eq:NN \__datapie_do:n \use_none:n
        \__datatool_action_error:nnn { datapie }
         {
           \token_to_str:N \DTLpiechart \c_space_tl ~ missing ~ variable
         }
         {
           You ~ need ~ to ~ either ~ use ~ `key' ~ or ~ `column' ~
           to ~ identify ~ the ~ variable ~ column ~ or ~ set ~
           the ~ `variable' ~ key ~ to ~ the ~ placeholder ~
           command ~ within ~ `options'
         }
       }
      \__datapie_do:n
       {
         \exp_args:NV \__datapie_do_chart:n
          \l__datatool_action_assign_clist
       }
     }
\group_end:
```

```
}
```

Set the offset angles.

`\@dtl@set@off` Version 3.0: replaced `\@dtl@set@off` with:

```
\cs_new:Npn \__datapie_set_offset:nw
 #1 - #2 \q_stop
{
  \tl_if_empty:nTF { #2 }
  {
    \__datapie_set_offset:n { #1 }
  }
  {
    \__datapie_set_offset_range:nw #1 - #2 \q_stop
  }
}
```

Set offset for individual segment:

`\@@dtl@set@off`

```
\cs_new:Nn \__datapie_set_offset:n
{
  \tl_set:cx { dtl@cut@angle@ \romannumeral #1 }
   {
     \fp_eval:n
      {
        0.5 * \tl_use:c { dtl@angle@ \romannumeral #1 }
        + \tl_use:c { dtl@sang@ \romannumeral #1 }
      }
   }
  \tl_set:cx
    { dtl@cut@len@ \romannumeral #1 }
    { \dim_use:N \l__datapie_cutaway_offset_dim }
}
```

`\@@dtl@setoffr` Set offset for a range of segments. Version 3.0: replaced `\@@dtl@setoffr` with:

```
\cs_new:Npn \__datapie_set_offset_range:nw
 #1 - #2 - \q_stop
{
 \int_compare:nNnTF { #1 } > { #2 }
  {
   \PackageError {datapie}
   {
     Segment ~ ranges ~ must ~ go ~ in ~ ascending ~ order
   }
   {
     Try ~ #2 - #1 ~ instead of ~ #1 - #2
   }
  }
  {
    \fp_zero:N \l__datapie_angle_fp
```

```
    \int_step_inline:nnn { #1 } { #2 }
     {
       \fp_add:Nn \l__datapie_angle_fp
        {
          \tl_use:c { dtl@angle@ \romannumeral ##1 }
        }
     }
    \tl_set:Nx \l__datapie_angle_tl
     {
       \fp_eval:n
        {
          0.5 * \l__datapie_angle_fp
          + \tl_use:c { dtl@sang@\romannumeral#1 }
        }
     }
    \int_step_inline:nnn { #1 } { #2 }
     {
       \tl_set_eq:cN { dtl@cut@angle@\romannumeral ##1 }
        \l__datapie_angle_tl
       \tl_set:cx
        { dtl@cut@len@\romannumeral ##1 }
         { \dim_use:N \l__datapie_cutaway_offset_dim }
     }
  }
}

\ExplSyntaxOff
```

28 dataplot.sty

Declare package:

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{dataplot-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{dataplot}[2026/07/31 v3.4.4 (NLCT)]
```

Version 3.0: no longer using xkeyval.

```
\IfPackageLoadedTF{datatool}
{
  \DeclareOption*{\expandafter\DTLsetup\expandafter{\CurrentOption}}
}
{
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datatool}}
}
\ProcessOptions
```

Required packages

```
\RequirePackage{datatool}
\RequirePackage{tikz}
```

Load TikZ plot libraries

```
\usetikzlibrary{plotmarks}
\usetikzlibrary{plothandlers}
```

Load calc library

```
\usetikzlibrary{calc}
```

```
\ExplSyntaxOn
```

28.1 Scratch variables.

List of x keys:

```
\clist_new:N \l__dataplot_x_keys_clist
\seq_new:N \l__dataplot_x_keys_seq
\int_new:N \l_dataplot_x_key_int
```

List of y keys:

```
\seq_new:N \l__dataplot_y_keys_seq
\int_new:N \l_dataplot_y_key_int
```

Any extra assignments that need to be made in the loop. For example, to assist filtering.

```
\tl_new:N \l__dataplot_extra_assign_tl
```

`\dtl@xkey` The database key for the x value. Version 3.0 replaced `\dtl@xkey` with:

```
\tl_new:N \l__dataplot_x_key_tl
```

`\dtl@ykey` The database key for the y value. Version 3.0 replaced `\dtl@ykey` with:

```
\tl_new:N \l__dataplot_y_key_tl
```

Placeholders for x and y obtained from database:

`\dtl@x` Version 3.0 replaced `\dtl@x` with:

```
\tl_new:N \l__dataplot_x_tl
```

`\dtl@decx` Version 3.0 replaced `\dtl@decx` with:

```
\tl_new:N \l__dataplot_decimal_x_tl
```

`\dtl@y` Version 3.0 replaced `\dtl@y` with:

```
\tl_new:N \l__dataplot_y_tl
```

`\dtl@decy` Version 3.0 replaced `\dtl@decy` with:

```
\tl_new:N \l__dataplot_decimal_y_tl
```

Floating point values:

```
\fp_new:N \l__dataplot_x_fp
\fp_new:N \l__dataplot_y_fp
```

Plot bounds.

```
\fp_new:N \l__dataplot_min_x_fp
\fp_new:N \l__dataplot_min_y_fp
\fp_new:N \l__dataplot_max_x_fp
\fp_new:N \l__dataplot_max_y_fp
```

Support for extending the axes beyond the plot bounds:

```
\fp_new:N \l__dataplot_extend_min_x_axis_fp
\fp_new:N \l__dataplot_extend_max_x_axis_fp
\fp_new:N \l__dataplot_extend_min_y_axis_fp
\fp_new:N \l__dataplot_extend_max_y_axis_fp
```

Extended plot bounds.

```
\fp_new:N \l__dataplot_extended_min_x_fp
\fp_new:N \l__dataplot_extended_min_y_fp
\fp_new:N \l__dataplot_extended_max_x_fp
\fp_new:N \l__dataplot_extended_max_y_fp
```

Tick gaps:

```
\fp_const:Nn \c_dataplot_smallest_gap_fp { 0.000001 }
```

`\@dtl@neggap` Version 3.0 replaced `\@dtl@neggap` with:

```
\fp_new:N \l__dataplot_neg_gap_fp
```

`\@dtl@posgap` Version 3.0 replaced `\@dtl@posgap` with:

```
\fp_new:N \l__dataplot_pos_gap_fp
```

`\@dtl@gap` Version 3.0 replaced `\@dtl@gap` with:

```
\fp_new:N \l__dataplot_gap_fp

\fp_new:N \l__dataplot_min_minor_gap_fp
```

Scale factors and offsets.

`\dtl@scale@x` Version 3.0 replaced `\dtl@scale@x` with:

```
\fp_new:N \l__dataplot_scale_x_fp
\fp_set_eq:NN \l__dataplot_scale_x_fp \c_one_fp
```

`\dtl@scale@y` Version 3.0 replaced `\dtl@scale@y` with:

```
\fp_new:N \l__dataplot_scale_y_fp
\fp_set_eq:NN \l__dataplot_scale_y_fp \c_one_fp
```

Inverse scale factors:

```
\fp_new:N \l__dataplot_inv_scale_x_fp
\fp_set_eq:NN \l__dataplot_inv_scale_x_fp \c_one_fp
\fp_new:N \l__dataplot_inv_scale_y_fp
\fp_set_eq:NN \l__dataplot_inv_scale_y_fp \c_one_fp
```

`\dtl@offset@x` Version 3.0 replaced `\dtl@offset@x` with:

```
\fp_new:N \l__dataplot_offset_x_fp
```

`\dtl@offset@y` Version 3.0 replaced `\dtl@offset@y` with:
```
\fp_new:N \l__dataplot_offset_y_fp
```

`\dtl@dx` Version 3.0 replaced `\dtl@dx` with:
```
\fp_new:N \l__dataplot_x_extent_fp
```

`\dtl@dy` Version 3.0 replaced `\dtl@dy` with:
```
\fp_new:N \l__dataplot_y_extent_fp
```

`\dtl@ticklength` Tick length in terms of canvas co-ordinates. Version 3.0 replaced `\dtl@ticklength` with:
```
\fp_new:N \l__dataplot_tick_length_fp
```

`\@dtl@width` Version 3.0 replaced `\@dtl@width` with:
```
\fp_new:N \l__dataplot_width_fp
\fp_new:N \l__dataplot_height_fp
```

`\dtl@xticlabelheight` The height of the x tick labels. Version 3.0: replaced `\dtl@xticlabelheight` with:
```
\dim_new:N \l__dataplot_x_tic_label_height_dim
```

`\dtl@yticlabelwidth` Dimension used to store the width of the y tick labels. Version 3.0 replaced `\dtl@yticlabelwidth` with:
```
\dim_new:N \l__dataplot_y_tic_label_width_dim
```

`\dtl@ticlabeloffset` The offset of the label. Version 3.0: replaced `\dtl@ticlabeloffset` with:
```
\fp_new:N \l__dataplot_tic_label_offset_fp
```

Tick label:
```
\tl_new:N \l__dataplot_tic_label_tl
```

List of database names:
```
\seq_new:N \l__dataplot_dbnames_seq
```

List of plot marks (obtained from `\DTLplotmarks`):
```
\seq_new:N \l__dataplot_mark_seq
```

`\dtl@mark` Current mark. Version 3.0: replaced `\dtl@mark` with:
```
\tl_new:N \l__dataplot_current_mark_tl
```

Boolean to determine whether to cycle marks for each database or for each (x, y) pair.
```
\bool_new:N \l__dataplot_mark_group_bool
\bool_set_false:N \l__dataplot_mark_group_bool
```

List of plot mark colours (obtained from `\DTLplotmarkcolors`):
```
\seq_new:N \l__dataplot_mark_colors_seq
```

`\dtl@thisplotmarkcolor` Current mark colour. Version 3.0: replaced `\dtl@thisplotmarkcolor` with:
```
\tl_new:N \l__dataplot_current_mark_color_tl
```

Boolean to determine whether to cycle mark colour for each database or for each (x, y) pair.

```
\bool_new:N \l__dataplot_mark_color_group_bool
\bool_set_false:N \l__dataplot_mark_color_group_bool
```

Current mark style and colour combination:

```
\tl_new:N \l__dataplot_current_mark_style_tl
```

List of line styles (obtained from `\DTLplotlines`):

```
\seq_new:N \l__dataplot_line_seq
```

`\dtl@linestyle` Current line. Version 3.0: replaced `\dtl@linestyle` with:

```
\tl_new:N \l__dataplot_current_line_tl
```

Boolean to determine whether to cycle line style for each database or for each (x, y) pair.

```
\bool_new:N \l__dataplot_line_group_bool
\bool_set_false:N \l__dataplot_line_group_bool
```

List of line colours (obtained from `\DTLplotlinecolors`):

```
\seq_new:N \l__dataplot_line_colors_seq
```

`\dtl@thisplotlinecolor` Current line colour. Version 3.0: replace `\dtl@thisplotlinecolor` with:

```
\tl_new:N \l__dataplot_current_line_color_tl
```

Boolean to determine whether to cycle line colour for each database or for each (x, y) pair.

```
\bool_new:N \l__dataplot_line_color_group_bool
\bool_set_false:N \l__dataplot_line_color_group_bool
```

Current line style and colour combination:

```
\tl_new:N \l__dataplot_current_line_style_tl
```

Boolean to determine whether to reset or cycle if group setting not on.

```
\bool_new:N \l__dataplot_no_group_mark_reset_bool
\bool_set_false:N \l__dataplot_no_group_mark_reset_bool
\bool_new:N \l__dataplot_no_group_mark_color_reset_bool
\bool_set_false:N \l__dataplot_no_group_mark_color_reset_bool
\bool_new:N \l__dataplot_no_group_line_reset_bool
\bool_set_false:N \l__dataplot_no_group_line_reset_bool
\bool_new:N \l__dataplot_no_group_line_color_reset_bool
\bool_set_false:N \l__dataplot_no_group_line_color_reset_bool
```

`\dtl@xminorticlist` List of minor x ticks. Version 3.0: replaced `\dtl@xminorticlist` with:

```
\seq_new:N \l__dataplot_x_minor_tic_seq
```

`\dtl@yminorticlist` List of minor y ticks. Version 3.0: replaced `\dtl@yminorticlist` with:

```
\seq_new:N \l__dataplot_y_minor_tic_seq
```

`\dtl@stream` Plot stream token list. Version 3.0: replaced `\dtl@stream` with:

```
\tl_new:N \l_dataplot_stream_tl
```

`\dtl@legend` Plot legend token list. This is a public temporary variable as it needs to be referenced in `\DTLaddtoplotlegend`, which may be redefined by the user. Version 3.0: replaced `\dtl@legend` with:

```
\tl_new:N \l_dataplot_legend_tl
```

`\dtl@bounds` The bounds of the graph may be given as a four-element list ⟨*min x*⟩, ⟨*min y*⟩, ⟨*max x*⟩, ⟨*max y*⟩. Version 3.0 replaced `\dtl@bounds` comma-separated list with the sequence variable:

```
\seq_new:N \l__dataplot_bounds_seq
```

Token list to contain the plot code:

```
\tl_new:N \l__dataplot_content_tl
```

28.2 Settings

`\DTLplotmarks` `\DTLplotmarks` contains a list of plot marks used by `\DTLplot`. This will be converted to the sequence `\l__dataplot_marks_seq` in `\DTLplot`.

```
\newcommand*{\DTLplotmarks}{%
  \pgfuseplotmark{o},%
  \pgfuseplotmark{x},%
  \pgfuseplotmark{+},%
  \pgfuseplotmark{square},%
  \pgfuseplotmark{triangle},%
  \pgfuseplotmark{diamond},%
  \pgfuseplotmark{pentagon},%
  \pgfuseplotmark{asterisk},%
  \pgfuseplotmark{star}%
}
```

`\DTLplotmarkcolors` `\DTLplotmarkcolors` contains a list of the plot mark colours. This will be converted to the sequence `\l__dataplot_mark_colors_seq` in `\DTLplot`.

```
\newcommand*{\DTLplotmarkcolors}{%
  red,%
  green,%
  blue,%
  yellow,%
  magenta,%
  cyan,%
  orange,%
  black,%
  gray}
```

`\DTLplotlines` `\DTLplotlines` contains a list of dash patterns used by `\DLTplot`. This will be converted to the sequence `\l__dataplot_line_seq` in `\DTLplot`.

```
\newcommand*{\DTLplotlines}{%
  \pgfsetdash{}{0pt},% solid line
  \pgfsetdash{{10pt}{5pt}}{0pt},%
  \pgfsetdash{{5pt}{5pt}}{0pt},%
```

```
    \pgfsetdash{{1pt}{5pt}}{0pt},%
    \pgfsetdash{{5pt}{5pt}{1pt}{5pt}}{0pt},%
    \pgfsetdash{{1pt}{3pt}}{0pt}%
  }
```

`\DTLplotlinecolors` `\DTLplotlinecolors` contains a list of the plot line colours. This will be converted to the sequence `\l__dataplot_line_colors_seq` in `\DTLplot`.

```
\newcommand*{\DTLplotlinecolors}{%
  red,%
  green,%
  blue,%
  yellow,%
  magenta,%
  cyan,%
  orange,%
  black,%
  gray}
```

`\DTLplotwidth` The default total plot width is stored in the length `\dtlplotwidth`

```
\newlength\DTLplotwidth
\setlength\DTLplotwidth{4in}
```

`\DTLplotheight` The default total plot height is stored in the length `\dtlplotheight`

```
\newlength\DTLplotheight
\setlength\DTLplotheight{4in}
```

`\DTLticklength` The length of the tick marks is given by `\DTLticklength`

```
\newlength\DTLticklength
\setlength\DTLticklength{5pt}
```

`\DTLminorticklength` The length of the minor tick marks is given by `\DTLminorticklength`.

```
\newlength\DTLminorticklength
\setlength\DTLminorticklength{2pt}
```

`\DTLticklabeloffset` The offset from the axis to the tick label is given by `\DTLticklabeloffset`.

```
\newlength\DTLticklabeloffset
\setlength\DTLticklabeloffset{8pt}
```

`\DTLmintickgap` `\DTLmintickgap` stores the suggested minimum distance between tick marks where the gap is not specified.

```
\newlength\DTLmintickgap
\setlength\DTLmintickgap{20pt}
```

`\DTLminminorticksgap` The suggested minimum distance between minor tick marks where the gap is not specified is given by `\DTLminminorticksgap`.

```
\newlength\DTLminminorticksgap
\setlength\DTLminminorticksgap{5pt}
```

`DTLplotroundXvar` Round x tick labels to the number of digits given by the counter DTLplotroundXvar.

```
\newcounter{DTLplotroundXvar}
\setcounter{DTLplotroundXvar}{2}
```

`DTLplotroundYvar` Round y tick labels to the number of digits given by the counter DTLplotroundYvar.

```
\newcounter{DTLplotroundYvar}
\setcounter{DTLplotroundYvar}{2}
```

`\ifDTLxaxis` The conditional `\ifDTLxaxis` is used to determine whether or not to display the x axis.

```
\newif\ifDTLxaxis
\DTLxaxistrue
```

`\DTLXAxisStyle` The style of the x axis is given by `\DTLXAxisStyle`. This is just a solid line by default.

```
\newcommand*{\DTLXAxisStyle}{-}
```

`\ifDTLyaxis` The conditional `\ifDTLyaxis` is used to determine whether or not to display the y axis

```
\newif\ifDTLyaxis
\DTLyaxistrue
```

`\DTLYAxisStyle` The style of the y axis is given by `\DTLYAxisStyle`. This is just a solid line by default.

```
\newcommand*{\DTLYAxisStyle}{-}
```

`\DTLmajorgridstyle` The style of the major grid lines is given by `\DTLmajorgridstyle`.

```
\newcommand*{\DTLmajorgridstyle}{color=gray,-}
```

`\DTLminorgridstyle` The style of the minor grid lines is given by `\DTLminorgridstyle`.

```
\newcommand*{\DTLminorgridstyle}{color=lightgray,very ~ thin}
```

`\ifDTLxticsin` The conditional `\ifDTLxticsin` is used to determine whether the x tics should point in or out.

```
\newif\ifDTLxticsin
\DTLxticsintrue
```

`\ifDTLyticsin` The conditional `\ifDTLyticsin` is used to determine whether the y tics should point in or out.

```
\newif\ifDTLyticsin
\DTLyticsintrue
```

`\DTLlegendxoffset` The gap between the border of plot and legend is given by the lengths `\DTLlegendxoffset` and `\DTLlegendyoffset`

```
\newlength\DTLlegendxoffset
\setlength\DTLlegendxoffset{10pt}
```

\DTLlegendyoffset

```
\newlength\DTLlegendyoffset
\setlength\DTLlegendyoffset{10pt}
```

\DTLformatlegend

```
\DTLformatlegend{⟨legend⟩}
```

This formats the legend.

```
\newcommand*{\DTLformatlegend}[1]{%
\setlength{\fboxrule}{1.1pt}%
\fcolorbox{black}{white}{#1}}
```

\DTLcustomlegend

```
\DTLcustomlegend{⟨legend⟩}
```

Used with the custom legend setting. This should be redefined to place the legend at the desired position.

```
\newcommand{\DTLcustomlegend}[1]{
  \node { \DTLformatlegend { #1 } } ;
}
```

Adds the begin tabular code for the legend:

```
\cs_new:Nn \dataplot_legend_add_begin:
{
  \tl_put_left:Nn \l_dataplot_legend_tl
   { \begin { tabular } { c l } }
}
```

Adds the end tabular code for the legend:

```
\cs_new:Nn \dataplot_legend_add_end:
{
  \tl_put_right:Nn \l_dataplot_legend_tl
   { \end { tabular } }
}
```

\ifDTLshowmarkers The conditional \ifDTLshowmarkers is used to specify whether or not to use markers.

```
\newif\ifDTLshowmarkers
\DTLshowmarkerstrue
```

\ifDTLshowlines The conditional \ifDTLshowlines is used to specify whether or not to use lines.

```
\newif\ifDTLshowlines
\DTLshowlinesfalse
```

\ifDTLbox Enclose plot in a box

```
\newif\ifDTLbox
\DTLboxfalse
```

`\ifDTLxticstrue` Condition to determine whether to show the x tick marks

```
\newif\ifDTLxtics
\DTLxticstrue
```

`\ifDTLyticstrue` Condition to determine whether to show the y tick marks

```
\newif\ifDTLytics
\DTLyticstrue
```

`\ifDTLxminortics` Condition to determine whether to show the x minor tick marks

```
\newif\ifDTLxminortics
\DTLxminorticsfalse
```

`\ifDTLyminortics` Condition to determine whether to show the y minor tick marks

```
\newif\ifDTLyminortics
\DTLyminorticsfalse
```

`\ifDTLgrid` Determine whether to draw the grid

```
\newif\ifDTLgrid
```

`\DTLplotdisplayticklabel`

```
\DTLplotdisplayticklabel{⟨text⟩}
```

Formatting used for default x tick labels.

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

`DTLplotdisplayXticklabel`

```
\DTLplotdisplayXticklabel{⟨text⟩}
```

Formatting used for default x tick labels.

```
\newcommand{\DTLplotdisplayXticklabel}[1]{\DTLplotdisplayticklabel{#1}}
```

x tick label node style:

```
\tl_new:N \l__dataplot_x_tick_label_style_tl
```

`DTLplotdisplayYticklabel`

```
\DTLplotdisplayYticklabel{⟨text⟩}
```

Formatting used for default y tick labels.

```
\newcommand{\DTLplotdisplayYticklabel}[1]{\DTLplotdisplayticklabel{#1}}
```

y tick label node style:

```
\tl_new:N \l__dataplot_y_tick_label_style_tl
\tl_set:Nn \l__dataplot_y_tick_label_style_tl { anchor=east }
```

`\dtl@minx` The lower x bound. Version 3.0: replaced `\dtl@minx` with:

```
\tl_new:N \l__dataplot_min_x_tl
```

`\dtl@maxx` The upper x bound: Version 3.0: replaced `\dtl@maxx` with:

```
\tl_new:N \l__dataplot_max_x_tl
```

`\dtl@miny` The lower y bound: Version 3.0: replaced `\dtl@miny` with:

```
\tl_new:N \l__dataplot_min_y_tl
```

`\dtl@maxy` The upper y bound: Version 3.0: replaced `\dtl@maxy` with:

```
\tl_new:N \l__dataplot_max_y_tl
```

`\dtl@xticlist` The list of floating point values for x ticks. Version 3.0: replaced `\dtl@xticlist` with:

```
\clist_new:N \l__dataplot_x_tic_clist
\seq_new:N \l__dataplot_x_tic_seq
```

`\dtl@yticlist` The list of decimal values for y ticks. Version 3.0: replaced `\dtl@yticlist` with:

```
\clist_new:N \l__dataplot_y_tic_clist
\seq_new:N \l__dataplot_y_tic_seq
```

`\dtl@xticgap` The gap between x tick marks (`xticpoints` overrides `xticgap`) Version 3.0: replaced `\dtl@xticgap` with:

```
\tl_new:N \l__dataplot_xtic_gap_tl
```

`\dtl@yticgap` The gap between y tick marks (`yticpoints` overrides `yticgap`) Version 3.0: replaced `\dtl@yticgap` with:

```
\tl_new:N \l__dataplot_ytic_gap_tl
```

`\dtl@xticlabels` A comma separated list of labels for x ticks. Version 3.0 replaced `\dtl@xticlabels` with a sequence variable:

```
\seq_new:N \l__dataplot_xtic_labels_seq
```

`\dtl@yticlabels` A comma separated list of labels for y ticks. Version 3.0 replaced `\dtl@yticlabels` with a sequence variable:

```
\seq_new:N \l__dataplot_ytic_labels_seq
```

`\dtl@xlabel` x axis label Version 3.0 replaced `\dtl@xlabel` with:

```
\tl_new:N \l__dataplot_xlabel_tl
```

`\dtl@ylabel` y axis label Version 3.0 replaced `\dtl@ylabel` with:

```
\tl_new:N \l__dataplot_ylabel_tl
```

Labels at either end of axes and their style:

```
\tl_new:N \l__dataplot_x_min_label_tl
\tl_new:N \l__dataplot_x_min_label_style_tl
\tl_set:Nn \l__dataplot_x_min_label_style_tl { left }
\tl_new:N \l__dataplot_x_max_label_tl
\tl_new:N \l__dataplot_x_max_label_style_tl
\tl_set:Nn \l__dataplot_x_max_label_style_tl { right }
\tl_new:N \l__dataplot_y_min_label_tl
```

```
\tl_new:N \l__dataplot_y_min_label_style_tl
\tl_set:Nn \l__dataplot_y_min_label_style_tl { below }
\tl_new:N \l__dataplot_y_max_label_tl
\tl_new:N \l__dataplot_y_max_label_style_tl
\tl_set:Nn \l__dataplot_y_max_label_style_tl { above }
```

`\dtl@legendlabels` Legend labels (comma separated list). If empty, the database name is used. Version 3.0 replaced `\dtl@legendlabels` with the sequence:

```
\seq_new:N \l__dataplot_legend_labels_seq
```

`\dtl@legendsetting` Integer variable corresponding to the legend setting: `none` (0, don't show it), `north` (1), `northeast` (2), `east` (3), `southeast` (4), `south` (5), `southwest` (6), `west` (7), `northwest` (8) or `custom` (9). The custom setting will use `\DTLcustomlegend` to position the legend which should be redefined as applicable. Version 3.0 replaced `\dtl@legendsetting` with:

```
\int_new:N \l__dataplot_legend_setting_int
```

Provide a filter function:

```
\cs_new:Nn \__dataplot_filter:T { #1 }
```

Convert a list of numbers to a sequence containing floating point markup (so that the numbers don't have to keep being reparsed). Only include numbers in the given range (inclusive) Syntax: ⟨*seq*⟩`{`⟨*clist*⟩`}{`⟨*min*⟩`}{`⟨*max*⟩`}`

```
\cs_new:Nn \__dataplot_to_fp_seq:NNnn
{
  \seq_clear:N #1
  \clist_map_inline:Nn { #2 }
  {
    \datatool_set_fp:Nn \l__datatool_tmpa_fp { ##1 }
    \fp_compare:nT
      { #3 <= \l__datatool_tmpa_fp <= #4 }
     {
       \seq_put_right:NV #1 \l__datatool_tmpa_fp
     }
  }
}
\cs_new:Nn \__dataplot_filtered_map:nnn
{
  \int_zero:N \l_dataplot_x_key_int
  \int_zero:N \l_dataplot_y_key_int
  \seq_set_from_clist:NN \l__dataplot_x_keys_seq
    \l__dataplot_x_keys_clist
```

Iterate over the (x, y) coordinate pairs:

```
  \seq_map_inline:Nn \l__dataplot_y_keys_seq
  {
    \int_incr:N \l_dataplot_y_key_int
```

The y key is the sequence iteration value:

```
    \tl_set:Nn \l__dataplot_y_key_tl { ##1 }
```

Get the x key:

```
\seq_if_empty:NT \l__dataplot_x_keys_seq
{
  \int_zero:N \l_dataplot_x_key_int
  \seq_set_from_clist:NN \l__dataplot_x_keys_seq
    \l__dataplot_x_keys_clist
}
\seq_pop_left:NN \l__dataplot_x_keys_seq
  \l__dataplot_x_key_tl
\int_incr:N \l_dataplot_x_key_int
```

Both the x and y key must exist for the current database for this plot stream. If either doesn't exist, skip this iteration. Check the y key exists for the current database:

```
\datatool_if_has_key:nnTF
  { \l__datatool_default_dbname_tl }
  { \l__dataplot_y_key_tl }
 {
```

Check the x key exists for the current database:

```
\datatool_if_has_key:nnTF
   { \l__datatool_default_dbname_tl }
   { \l__dataplot_x_key_tl }
  {
```

Pre map code:

```
#1
```

Iterate over all rows in this database:

```
\DTLmapdata
{
  \tl_if_empty:NF \l__dataplot_extra_assign_tl
  {
    \exp_args:NV \DTLmapgetvalues
     \l__dataplot_extra_assign_tl
  }
```

Get the x coordinate:

```
\DTLmapget
 {
   return = \l__dataplot_x_tl ,
   key = \l__dataplot_x_key_tl
 }
```

Get the y coordinate:

```
\DTLmapget
 {
   return = \l__dataplot_y_tl ,
   key = \l__dataplot_y_key_tl
 }
\__dataplot_filter:T { #2 }
}
```

Post map code:

```
          #3
        }
        {
```

The y key isn't defined for the current database.

```
          \dataplot_skipping_key_msg:nnnn
            { \seq_count:N \l__dataplot_dbnames_seq }
            { x } { \l__dataplot_x_key_tl }
            { \l__datatool_default_dbname_tl }
        }
     }
     {
```

The y key isn't defined for the current database.

```
       \dataplot_skipping_key_msg:nnnn
         { \seq_count:N \l__dataplot_dbnames_seq }
         { y } { \l__dataplot_y_key_tl }
         { \l__datatool_default_dbname_tl }
     }
   }
 }
 \cs_new:Nn \__dataplot_filtered_map:n
 {
   \__dataplot_filtered_map:nnn { } { #1 } { }
 }
```

Syntax: `{`⟨*num databases*⟩`}{`⟨*x/y*⟩`}{`⟨*key*⟩`}{`⟨*db-name*⟩`}` Warn or verbose message if skipping key:

```
 \cs_new:Nn \dataplot_skipping_key_msg:nnnn
 {
   \int_compare:nNnTF { #1 } = { \c_one_int }
    {
       \PackageWarning { dataplot }
       {
         \token_to_str:N \DTLplot : ~ skipping ~ #2 ~ key ~
         ` #3 ' ~ for ~ database ~ `\l__datatool_default_dbname_tl' ~
         (column ~ not ~ defined)
       }
    }
    {
      \dtl@message
       {
         \token_to_str:N \DTLplot : ~ skipping ~ #2 ~ key ~
         ` #3 ' ~ for ~ database ~ `\l__datatool_default_dbname_tl' ~
         (column ~ not ~ defined)
       }
    }
 }
```

If the side axes boolean is on, the axes will always be draw on the left and bottom. If

the setting is off and there are negative values, the axes will pass through 0 otherwise they will be drawn on the side.

```
\bool_new:N \l__dataplot_side_axes_bool
\bool_set_false:N \l__dataplot_side_axes_bool
```

Booleans set after computing the bounds. True if x axis includes 0. That is, the y axis should intersect at x=0:

```
\bool_new:N \l__dataplot_zero_x_axis_bool
```

True if y axis includes 0. That is, the x axis should intersect at y=0:

```
\bool_new:N \l__dataplot_zero_y_axis_bool
```

Omit tick label at $x = 0$ or $y = 0$ if true:

```
\bool_new:N \l__dataplot_omit_zero_x_label_bool
\bool_new:N \l__dataplot_omit_zero_y_label_bool
```

Setting to determine whether or not to omit zero labels: 1 (auto), 2 (omit both), 3 (omit x), 4 (omit y), 5 (don't omit)

```
\int_new:N \l__dataplot_omit_zero_label_action_int
```

Determines whether to show ticks on the box if box=true. Values: 1=match axes, 2=in, 3=out, 0=none. Any other value is treated as 0.

```
\int_new:N \l__dataplot_box_ticks_int
\int_set_eq:NN \l__dataplot_box_ticks_int \c_one_int
```

```
\keys_define:nn { datatool/plot }
{
```

Plot settings. The database key for the x value is given by the `x` setting:

```
  x .clist_set:N = \l__dataplot_x_keys_clist ,
```

The database key for the y value is given by the `y` setting:

```
  y .code:n  =
    {
      \seq_set_from_clist:Nn \l__dataplot_y_keys_seq { #1 }
    },
```

Any extra assignments that need to be made in `\DTLmapdata`:

```
  extra-assign .tl_set:N = \l__dataplot_extra_assign_tl ,
```

The list of plot mark colours is given by the `markcolors` setting. (This should be a comma separated list of colour names.)

```
  markcolors .clist_set:N = \DTLplotmarkcolors,
  mark-colors .clist_set:N = \DTLplotmarkcolors,
```

The list of plot line colours is given by the `linecolors` setting. (This should be a comma separated list of colour names.)

```
  linecolors .clist_set:N = \DTLplotlinecolors,
  line-colors .clist_set:N = \DTLplotlinecolors,
```

The list of plot mark and line colours is given by the `colors` setting. (This should be a comma separated list of colour names.)

```
  colors .code:n =
```

```
 {
   \clist_set:Nn \DTLplotmarkcolors { #1 }
   \clist_set:Nn \DTLplotlinecolors { #1 }
 },
```

The list of plot marks is given by the `marks` setting. (This should be a comma separated list of code that generates pgf plot marks.)

```
marks .clist_set:N = \DTLplotmarks ,
```

The list of plot line styles is given by the `lines` setting. (This should be a comma separated list of code that sets the line style.) An empty set will create solid lines.

```
lines .clist_set:N = \DTLplotlines ,
```

Determine whether or not styles not enabled with group-styles should continue cycling or reset at the start of each database.

```
style-resets .multichoice: ,
style-resets .default:n = { all },
style-resets / mark-style .code:n =
 {
   \bool_set_true:N \l__dataplot_no_group_mark_reset_bool
 } ,
style-resets / mark-color .code:n =
 {
   \bool_set_true:N \l__dataplot_no_group_mark_color_reset_bool
 } ,
style-resets / line-style .code:n =
 {
   \bool_set_true:N \l__dataplot_no_group_line_reset_bool
 } ,
style-resets / line-color .code:n =
 {
   \bool_set_true:N \l__dataplot_no_group_line_color_reset_bool
 } ,
style-resets / all .code:n =
 {
   \bool_set_true:N \l__dataplot_no_group_mark_reset_bool
   \bool_set_true:N \l__dataplot_no_group_mark_color_reset_bool
   \bool_set_true:N \l__dataplot_no_group_line_reset_bool
   \bool_set_true:N \l__dataplot_no_group_line_color_reset_bool
 } ,
style-resets / none .code:n =
 {
   \bool_set_false:N \l__dataplot_no_group_mark_reset_bool
   \bool_set_false:N \l__dataplot_no_group_mark_color_reset_bool
   \bool_set_false:N \l__dataplot_no_group_line_reset_bool
   \bool_set_false:N \l__dataplot_no_group_line_color_reset_bool
 } ,
```

Determine whether to cycle mark and line styles for each database or for each (x, y) pair.

```
group-styles .multichoice:,
```

```
    group-styles .default:n = { all },
    group-styles / mark-style .code:n =
     {
       \bool_set_true:N \l__dataplot_mark_group_bool
     } ,
    group-styles / mark-color .code:n =
     {
       \bool_set_true:N \l__dataplot_mark_color_group_bool
     } ,
    group-styles / line-style .code:n =
     {
       \bool_set_true:N \l__dataplot_line_group_bool
     } ,
    group-styles / line-color .code:n =
     {
       \bool_set_true:N \l__dataplot_line_color_group_bool
     } ,
    group-styles / all .code:n =
     {
       \bool_set_true:N \l__dataplot_mark_group_bool
       \bool_set_true:N \l__dataplot_mark_color_group_bool
       \bool_set_true:N \l__dataplot_line_group_bool
       \bool_set_true:N \l__dataplot_line_color_group_bool
     } ,
    group-styles / none .code:n =
     {
       \bool_set_false:N \l__dataplot_mark_group_bool
       \bool_set_false:N \l__dataplot_mark_color_group_bool
       \bool_set_false:N \l__dataplot_line_group_bool
       \bool_set_false:N \l__dataplot_line_color_group_bool
     } ,
```

The total width of the plot is given by the `width` setting.

```
    width .dim_set:N = \DTLplotwidth ,
```

The total height of the plot is given by the `height` setting.

```
    height .dim_set:N =\DTLplotheight ,
```

Tick label offset.

```
    tick-label-offset .dim_set:N = \DTLticklabeloffset ,
```

Determine whether to show lines, markers or both

```
    style .choice:,
    style / both .code:n =
     {
       \DTLshowlinestrue
       \DTLshowmarkerstrue
     },
    style / lines .code:n =
     {
       \DTLshowlinestrue
```

```
    \DTLshowmarkersfalse
  },
 style / markers .code:n =
  {
    \DTLshowmarkerstrue
    \DTLshowlinesfalse
  },
```

Determine whether or not to display the axes

```
 axes .choice:,
 axes / both .code:n =
  {
    \DTLxaxistrue
    \DTLxticstrue
    \DTLyaxistrue
    \DTLyticstrue
  },
 axes / x .code:n =
  {
    \DTLxaxistrue
    \DTLxticstrue
    \DTLyaxisfalse
    \DTLyticsfalse
  },
 axes / y .code:n =
  {
    \DTLxaxisfalse
    \DTLxticsfalse
    \DTLyaxistrue
    \DTLyticstrue
  },
 axes / none .code:n =
  {
   \DTLxaxisfalse
   \DTLxticsfalse
   \DTLyaxisfalse
   \DTLyticsfalse
  },
```

Axis drawing style:

```
 x-axis-style .tl_set:N = \DTLXAxisStyle ,
 y-axis-style .tl_set:N = \DTLYAxisStyle ,
 axis-style .code:n =
  {
    \tl_set:Nn \DTLXAxisStyle { #1 }
    \tl_set:Nn \DTLYAxisStyle { #1 }
  },
```

Major and minor grid drawing style:

```
 major-grid-style .tl_set:N = \DTLmajorgridstyle ,
 minor-grid-style .tl_set:N = \DTLminorgridstyle ,
```

Indicate whether or not the plot should be enclosed in a box:

```
box .legacy_if_set:n = DTLbox ,
```

Indicate if the box should also have tick marks:

```
box-ticks .choices:nn=
  { none, match-axes, in , out }
 {
   \int_set:Nn \l__dataplot_box_ticks_int
    { \l_keys_choice_int - \c_one_int }
 } ,
box-ticks .default:n = { match-axes },
box-ticks .initial:n = { match-axes },
```

Rounding for default tick labels:

```
round-x .int_set:N = \c@DTLplotroundXvar ,
round-y .int_set:N = \c@DTLplotroundYvar ,
round .code:n =
 {
   \int_set:Nn \c@DTLplotroundXvar { #1 }
   \int_set:Nn \c@DTLplotroundYvar { #1 }
 },
```

Indicates whether to show the x tick marks:

```
xtics .legacy_if_set:n = DTLxtics ,
x-ticks .legacy_if_set:n = DTLxtics ,
```

Indicates whether to show the y tick marks:

```
ytics .legacy_if_set:n = DTLytics ,
y-ticks .legacy_if_set:n = DTLytics ,
```

Indicates whether to show the x and y tick marks:

```
tics .code:n =
 {
   \setbool{DTLxtics}{#1}
   \setbool{DTLytics}{#1}
 },
ticks .code:n =
 {
   \setbool{DTLxtics}{#1}
   \setbool{DTLytics}{#1}
 },
```

Indicates whether to show the x minor tick marks:

```
xminortics .choice: ,
xminortics / true .code:n =
  {
    \DTLxminorticstrue
    \DTLxticstrue
  } ,
xminortics / false .code:n =
  {
    \DTLxminorticsfalse
```

```
  } ,
xminortics .default:n = true,
```

Synonym:

```
x-minor-ticks .choice: ,
x-minor-ticks / true .code:n =
  {
    \DTLxminorticstrue
    \DTLxticstrue
  } ,
x-minor-ticks / false .code:n =
  {
    \DTLxminorticsfalse
  } ,
x-minor-ticks .default:n = true,
```

Indicates whether to show the y minor tick marks:

```
yminortics .choice:,
yminortics / true .code:n =
  {
    \DTLyminorticstrue
    \DTLyticstrue
  } ,
yminortics / false .code:n =
  {
    \DTLyminorticsfalse
  } ,
yminortics .default:n = true,
```

Synonym:

```
y-minor-ticks .choice:,
y-minor-ticks / true .code:n =
  {
    \DTLyminorticstrue
    \DTLyticstrue
  } ,
y-minor-ticks / false .code:n =
  {
    \DTLyminorticsfalse
  } ,
y-minor-ticks .default:n = true,
```

Indicates whether to show the x and y minor tick marks:

```
minor-ticks .choice: ,
minor-ticks / true .code:n =
  {
    \DTLxminorticstrue
    \DTLxticstrue
    \DTLyminorticstrue
    \DTLyticstrue
  } ,
```

```
    minor-ticks / false .code:n =
      {
        \DTLxminorticsfalse
        \DTLyminorticsfalse
      } ,
    minor-ticks .default:n = true,
```

Synonym:

```
    minortics .choice: ,
    minortics / true .code:n =
      {
        \DTLxminorticstrue
        \DTLxticstrue
        \DTLyminorticstrue
        \DTLyticstrue
      } ,
    minortics / false .code:n =
      {
        \DTLxminorticsfalse
        \DTLyminorticsfalse
      } ,
    minortics .default:n = true,
```

Determine whether to draw the grid:

```
    grid .legacy_if_set:n = DTLgrid ,
```

Determine whether the x tick marks should point in or out:

```
    xticdir .choice: ,
    xticdir / in .code:n =
      {
        \DTLxticsintrue
      },
    xticdir / out .code:n =
      {
        \DTLxticsinfalse
      },
```

Synonym:

```
    x-tick-dir .choice: ,
    x-tick-dir / in .code:n =
      {
        \DTLxticsintrue
      },
    x-tick-dir / out .code:n =
      {
        \DTLxticsinfalse
      },
```

Determine whether the y tick marks should point in or out:

```
    yticdir .choice: ,
    yticdir / in .code:n =
      {
```

```
      \DTLyticsintrue
    },
  yticdir / out .code:n =
    {
      \DTLyticsinfalse
    },
```

Synonym:

```
  y-tick-dir .choice: ,
  y-tick-dir / in .code:n =
    {
      \DTLyticsintrue
    },
  y-tick-dir / out .code:n =
    {
      \DTLyticsinfalse
    },
```

Determine whether the x and y tick marks should point in or out:

```
  ticdir .choice: ,
  ticdir / in .code:n =
    {
      \DTLxticsintrue
      \DTLyticsintrue
    },
  ticdir / out .code:n =
    {
      \DTLxticsinfalse
      \DTLyticsinfalse
    },
```

Synonym:

```
  tick-dir .choice: ,
  tick-dir / in .code:n =
    {
      \DTLxticsintrue
      \DTLyticsintrue
    },
  tick-dir / out .code:n =
    {
      \DTLxticsinfalse
      \DTLyticsinfalse
    },
```

Set the bounds of the graph (value must be in the form $\langle min\ x \rangle$, $\langle min\ y \rangle$, $\langle max\ x \rangle$, $\langle max\ y \rangle$. Version 3.0: bounds no longer overrides minx, miny, maxx and maxy settings. Set to empty to cancel.

```
  bounds .code:n =
   {
     \seq_set_from_clist:Nn \l__dataplot_bounds_seq { #1 }
     \seq_if_empty:NF \l__dataplot_bounds_seq
```

```
      {
        \int_compare:nNnTF
          { \seq_count:N \l__dataplot_bounds_seq } = { 4 }
        {
          \tl_set:Nx \l__dataplot_min_x_tl
            { \seq_item:Nn \l__dataplot_bounds_seq { 1 } }
          \tl_set:Nx \l__dataplot_min_y_tl
            { \seq_item:Nn \l__dataplot_bounds_seq { 2 } }
          \tl_set:Nx \l__dataplot_max_x_tl
            { \seq_item:Nn \l__dataplot_bounds_seq { 3 } }
          \tl_set:Nx \l__dataplot_max_y_tl
            { \seq_item:Nn \l__dataplot_bounds_seq { 4 } }
        }
        {
           \PackageError {dataplot}
           {
             Invalid ~ syntax ~ in ~ bounds=#1 ~
             (expected ~ empty ~ or ~ four ~ items, ~ found ~
              \seq_count:N \l__dataplot_bounds_seq)
           }
           {
             The ~ bounds ~ must ~ be ~ specified ~ as ~ minX,minY,maxX,maxY
           }
        }
      }
    },
```

Set only the lower x bound

```
    minx .tl_set:N = \l__dataplot_min_x_tl ,
    min-x .tl_set:N = \l__dataplot_min_x_tl ,
```

Set only the upper x bound

```
    maxx .tl_set:N = \l__dataplot_max_x_tl ,
    max-x .tl_set:N = \l__dataplot_max_x_tl ,
```

Set only the lower y bound

```
    miny .tl_set:N = \l__dataplot_min_y_tl ,
    min-y .tl_set:N = \l__dataplot_min_y_tl ,
```

Set only the upper y bound

```
    maxy .tl_set:N = \l__dataplot_max_y_tl ,
    max-y .tl_set:N = \l__dataplot_max_y_tl ,
```

Define list of points for x ticks. (Must be a comma separated list of decimal numbers.)
These need to be converted to a sequence containing the floating point representation.

```
  xticpoints .code:n =
    {
      \tl_if_empty:nTF { #1 }
       {
         \clist_clear:N \l__dataplot_x_tic_clist
       }
       {
```

```
        \clist_set:Nn \l__dataplot_x_tic_clist { #1 }
        \DTLxticstrue
        \DTLxaxistrue
      }
  },
x-tick-points .code:n =
  {
    \tl_if_empty:nTF { #1 }
      {
        \clist_clear:N \l__dataplot_x_tic_clist
      }
      {
        \clist_set:Nn \l__dataplot_x_tic_clist { #1 }
        \DTLxticstrue
        \DTLxaxistrue
      }
  },
```

Define list of points for y ticks. (Must be a comma separated list of decimal numbers.)

```
yticpoints .code:n =
  {
    \tl_if_empty:nTF { #1 }
      {
        \clist_clear:N \l__dataplot_y_tic_clist
      }
      {
        \clist_set:Nn \l__dataplot_y_tic_clist { #1 }
        \DTLyticstrue
        \DTLyaxistrue
      }
  },
y-tick-points .code:n =
  {
    \tl_if_empty:nTF { #1 }
      {
        \clist_clear:N \l__dataplot_y_tic_clist
      }
      {
        \clist_set:Nn:Nn \l__dataplot_y_tic_clist { #1 }
        \DTLyticstrue
        \DTLyaxistrue
      }
  },
```

Define the gap between x tick marks (xticpoints overrides xticgap)

```
xticgap .code:n =
 {
   \tl_set:Nn \l__dataplot_xtic_gap_tl { #1 }
   \tl_if_empty:NF \l__dataplot_xtic_gap_tl
    {
      \DTLxticstrue
```

```
      \DTLxaxistrue
    }
 },
x-tick-gap .code:n =
 {
   \tl_set:Nn \l__dataplot_xtic_gap_tl { #1 }
   \tl_if_empty:NF \l__dataplot_xtic_gap_tl
    {
     \DTLxticstrue
     \DTLxaxistrue
    }
 },
```

Define the gap between y tick marks (yticpoints overrides yticgap)

```
yticgap .code:n =
 {
   \tl_set:Nn \l__dataplot_ytic_gap_tl { #1 }
   \tl_if_empty:NF \l__dataplot_ytic_gap_tl
    {
     \DTLyticstrue
     \DTLyaxistrue
    }
 },
y-tick-gap .code:n =
 {
   \tl_set:Nn \l__dataplot_ytic_gap_tl { #1 }
   \tl_if_empty:NF \l__dataplot_ytic_gap_tl
    {
     \DTLyticstrue
     \DTLyaxistrue
    }
 },
```

Define the gap between x and y tick marks:

```
ticgap .code:n =
 {
   \tl_set:Nn \l__dataplot_xtic_gap_tl { #1 }
   \tl_set:Nn \l__dataplot_ytic_gap_tl { #1 }
   \tl_if_empty:NF \l__dataplot_xtic_gap_tl
    {
     \DTLxticstrue
     \DTLxaxistrue
     \DTLyticstrue
     \DTLyaxistrue
    }
 },
tick-gap .code:n =
 {
   \tl_set:Nn \l__dataplot_xtic_gap_tl { #1 }
   \tl_set:Nn \l__dataplot_ytic_gap_tl { #1 }
   \tl_if_empty:NF \l__dataplot_xtic_gap_tl
```

```
      {
       \DTLxticstrue
       \DTLxaxistrue
       \DTLyticstrue
       \DTLyaxistrue
      }
  },
```

Define comma separated list of labels for x ticks.

```
  xticlabels .code:n =
   {
     \tl_if_empty:nTF { #1 }
      {
        \seq_clear:N \l__dataplot_xtic_labels_seq
      }
      {
       \seq_set_from_clist:Nn \l__dataplot_xtic_labels_seq { #1 }
       \DTLxticstrue
       \DTLxaxistrue
      }
   },
  x-tick-labels .code:n =
   {
     \tl_if_empty:nTF { #1 }
      {
        \seq_clear:N \l__dataplot_xtic_labels_seq
      }
      {
       \seq_set_from_clist:Nn \l__dataplot_xtic_labels_seq { #1 }
       \DTLxticstrue
       \DTLxaxistrue
      }
   },
```

Define comma separated list of labels for y ticks.

```
  yticlabels .code:n =
   {
     \tl_if_empty:nTF { #1 }
      {
        \seq_clear:N \l__dataplot_ytic_labels_seq
      }
      {
       \seq_set_from_clist:Nn \l__dataplot_ytic_labels_seq { #1 }
       \DTLyticstrue
       \DTLyaxistrue
      }
   },
  y-tick-labels .code:n =
   {
     \tl_if_empty:nTF { #1 }
      {
```

```
      \seq_clear:N \l__dataplot_ytic_labels_seq
    }
    {
     \seq_set_from_clist:Nn \l__dataplot_ytic_labels_seq { #1 }
     \DTLyticstrue
     \DTLyaxistrue
    }
 },
```

Define x axis label:

```
xlabel .tl_set:N = \l__dataplot_xlabel_tl ,
x-label .tl_set:N = \l__dataplot_xlabel_tl ,
```

Define y axis label:

```
ylabel .tl_set:N = \l__dataplot_ylabel_tl ,
y-label .tl_set:N = \l__dataplot_ylabel_tl ,
```

The legend setting may be one of: `none` (don't show it), `north`, `northeast`, `east`, `southeast`, `south`, `southwest`, `west`, or `northwest`. These set the count register `\dtl@legendsetting`.

```
legend .choices:nn =
 {
    none, north, northeast, east, southeast,
    south, southwest, west, northwest, custom
 }
 {
   \int_set:Nn \l__dataplot_legend_setting_int
     { \l_keys_choice_int - 1 }
 },
 legend .default:n = { northeast },
```

Legend labels (comma separated list). If omitted, the default is used.

```
 legendlabels .code:n =
  {
    \seq_set_from_clist:Nn
       \l__dataplot_legend_labels_seq { #1 }
  } ,
 legend-labels .code:n =
  {
    \seq_set_from_clist:Nn
       \l__dataplot_legend_labels_seq { #1 }
  } ,
```

Legend offset may be a single value, in which case it applies to both x and y offsets, or two values.

```
 legend-offset .code:n =
  {
    \seq_set_from_clist:Nn
      \l__datatool_tmp_seq { #1 }
    \int_case:nnF { \seq_count:N \l__datatool_tmp_seq  }
     {
```

```
        { 1 }
          {
            \dim_set:Nn \DTLlegendxoffset
              { \seq_item:Nn \l__datatool_tmp_seq { 1 } }
            \dim_set_eq:Nn \DTLlegendyoffset \DTLlegendxoffset
          }
        { 2 }
          {
            \dim_set:Nn \DTLlegendxoffset
              { \seq_item:Nn \l__datatool_tmp_seq { 1 } }
            \dim_set:Nn \DTLlegendyoffset
              { \seq_item:Nn \l__datatool_tmp_seq { 2 } }
          }
      }
      {
        \PackageError { dataplot }
        {
           Invalid ~ legend-offset ~ value ~ `#1': one ~ or ~ two ~
           dimensions ~ required
        }
        {
          The ~ legend-offset ~ value ~ must ~ be ~ either ~ a ~
          single ~ dimension ~ or ~ two ~ comma-separated ~ dimensions
        }
      }
   },
```

Set whether or not to enforce side axes:

```
side-axes .bool_set:N = \l__dataplot_side_axes_bool ,
```

Should the zero labels be omitted:

```
omit-zero-label .choices:nn =
 { auto , both , x , y , false }
 {
   \int_set_eq:NN
     \l__dataplot_omit_zero_label_action_int
     \l_keys_choice_int
 } ,
omit-zero-label .initial:n = { auto } ,
omit-zero-label .default:n = { both } ,
```

Allow the x axis to be extended beyond the plot bounds:

```
extend-x-axis .code:n =
 {
   \int_case:nnF
     { \clist_count:n { #1 } }
    {
      { \c_one_int }
        {
          \exp_args:NNx \datatool_set_fp:Nn
            \l__dataplot_extend_min_x_axis_fp
```

```
            { \clist_item:nn { #1 } { 1 } }
          \fp_set_eq:NN
            \l__dataplot_extend_max_x_axis_fp
            \l__dataplot_extend_min_x_axis_fp
        }
      { 2 }
        {
          \exp_args:NNx \datatool_set_fp:Nn
            \l__dataplot_extend_min_x_axis_fp
            { \clist_item:nn { #1 } { 1 } }
          \exp_args:NNx \datatool_set_fp:Nn
             \l__dataplot_extend_max_x_axis_fp
            { \clist_item:nn { #1 } { 2 } }
        }
    }
    {
       \PackageError { dataplot }
       {
          Invalid ~ extend-x-axis ~ value ~ `#1': one ~ or ~ two ~
          numbers ~ required
       }
       {
         The ~ extend-x-axis ~ value ~ must ~ be ~ either ~ a ~
         single ~ number ~ or ~ two ~ comma-separated ~ numbers
       }
    }
  },
```

Allow the y axis to be extended beyond the plot bounds:

```
  extend-y-axis .code:n =
   {
     \int_case:nnF
       { \clist_count:n { #1 } }
      {
        { \c_one_int }
          {
            \exp_args:NNx \datatool_set_fp:Nn
               \l__dataplot_extend_min_y_axis_fp
              { \clist_item:nn { #1 } { 1 } }
            \fp_set_eq:NN
              \l__dataplot_extend_max_y_axis_fp
              \l__dataplot_extend_min_y_axis_fp
          }
        { 2 }
          {
            \exp_args:NNx \datatool_set_fp:Nn
               \l__dataplot_extend_min_y_axis_fp
              { \clist_item:nn { #1 } { 1 } }
            \exp_args:NNx \datatool_set_fp:Nn
              \l__dataplot_extend_max_y_axis_fp
```

```
            { \clist_item:nn { #1 } { 2 } }
         }
      }
      {
         \PackageError { dataplot }
         {
            Invalid ~ extend-y-axis ~ value ~ `#1': one ~ or ~ two ~
            numbers ~ required
         }
         {
           The ~ extend-y-axis ~ value ~ must ~ be ~ either ~ a ~
           single ~ number ~ or ~ two ~ comma-separated ~ numbers
         }
      }
   },
```

Set the extension for both x and y axes:

```
extend-axes .code:n =
 {
   \int_case:nnF
      { \clist_count:n { #1 } }
    {
      { \c_one_int }
        {
          \exp_args:NNx \datatool_set_fp:Nn
            \l__dataplot_extend_min_x_axis_fp
            { \clist_item:nn { #1 } { 1 } }
          \fp_set_eq:NN
            \l__dataplot_extend_max_x_axis_fp
            \l__dataplot_extend_min_x_axis_fp
        }
      { 2 }
        {
          \exp_args:NNx \datatool_set_fp:Nn
            \l__dataplot_extend_min_x_axis_fp
            { \clist_item:nn { #1 } { 1 } }
          \exp_args:NNx \datatool_set_fp:Nn
             \l__dataplot_extend_max_x_axis_fp
            { \clist_item:nn { #1 } { 2 } }
        }
    }
    {
       \PackageError { dataplot }
       {
          Invalid ~ extend-x-axis ~ value ~ `#1': one ~ or ~ two ~
          numbers ~ required
       }
       {
         The ~ extend-x-axis ~ value ~ must ~ be ~ either ~ a ~
         single ~ number ~ or ~ two ~ comma-separated ~ numbers
```

```
      }
    }
   \fp_set_eq:NN
     \l__dataplot_extend_min_y_axis_fp
     \l__dataplot_extend_min_x_axis_fp
   \fp_set_eq:NN
     \l__dataplot_extend_max_y_axis_fp
     \l__dataplot_extend_max_x_axis_fp
 },
```

Label minimum end of the x axis:

```
min-x-label .tl_set:N = \l__dataplot_x_min_label_tl ,
min-x-label-style .tl_set:N = \l__dataplot_x_min_label_style_tl ,
```

Label maximum end of the x axis:

```
max-x-label .tl_set:N = \l__dataplot_x_max_label_tl ,
max-x-label-style .tl_set:N = \l__dataplot_x_max_label_style_tl ,
```

Label minimum end of the y axis:

```
min-y-label .tl_set:N = \l__dataplot_y_min_label_tl ,
min-y-label-style .tl_set:N = \l__dataplot_y_min_label_style_tl ,
```

Label maximum end of the y axis:

```
max-y-label .tl_set:N = \l__dataplot_y_max_label_tl ,
max-y-label-style .tl_set:N = \l__dataplot_y_max_label_style_tl ,
```

Style for x tick labels:

```
x-tick-label-style .tl_set:N = \l__dataplot_x_tick_label_style_tl ,
x-tic-label-style .tl_set:N = \l__dataplot_x_tick_label_style_tl ,
```

Style for y tick labels:

```
y-tick-label-style .tl_set:N = \l__dataplot_y_tick_label_style_tl ,
y-tic-label-style .tl_set:N = \l__dataplot_y_tick_label_style_tl ,
```

Shortcut for both:

```
tick-label-style .code:n =
 {
   \tl_set:Nn \l__dataplot_x_tick_label_style_tl { #1 }
   \tl_set:Nn \l__dataplot_y_tick_label_style_tl { anchor=east, #1 }
 } ,
tic-label-style .code:n =
 {
   \tl_set:Nn \l__dataplot_x_tick_label_style_tl { #1 }
   \tl_set:Nn \l__dataplot_y_tick_label_style_tl { anchor=east, #1 }
 } ,
```

Filter:

```
include-if .cs_set:Np = \__dataplot_filter:T #1,
include-if-fn .code:n =
 {
   \cs_set_eq:NN \__dataplot_filter:T #1
 },
```

Deprecated experimental setting. TODO remove

```
   condition .code:n =
    {
      \PackageWarning{dataplot}{Deprecated ~ option ~ `condition'. ~
       Use ~ `include-if' ~ instead}
      \cs_set:Nn = \__dataplot_filter:T { #1 }
    }
}
```

Allow these keys to be set in `\DTLsetup{plot={...}}`

```
\keys_define:nn { datatool }
{
  plot .code:n = { \keys_set:nn { datatool/plot } { #1 } }
}
```

29 Plotting Commands and Hooks

Calculate the transformation matrix and extent based on `\DTLminX`, `\DTLminY`, `\DTLmaxX`, `\DTLmaxY`, `\DTLplotwidth` and `\DTLplotheight`. Note that this doesn't take the labels and axis extensions into account. The x-scale factor is given by:

$$s_x = \frac{W}{x_{\max} - x_{\min}}$$

where W is the plot width. The x offset is $-s_x x_{\min}$. Similarly for y.

```
\cs_new:Nn \dataplot_calc_transform:
{
  \fp_set:Nn \l__dataplot_x_extent_fp
    { \DTLmaxX - \DTLminX }
  \fp_set:Nn \l__dataplot_scale_x_fp
   {
```

The width as points:

```
     \dim_to_decimal:n \DTLplotwidth / \l__dataplot_x_extent_fp
   }
  \fp_set:Nn \l__dataplot_offset_x_fp
   {
     - \l__dataplot_scale_x_fp * \DTLminX
   }
  \fp_set:Nn \l__dataplot_y_extent_fp
   { \DTLmaxY - \DTLminY }
  \fp_set:Nn \l__dataplot_scale_y_fp
   {
```

The height as points:

```
     \dim_to_decimal:n \DTLplotheight / \l__dataplot_y_extent_fp
   }
  \fp_set:Nn \l__dataplot_offset_y_fp
   {
```

```
      - \l__dataplot_scale_y_fp * \DTLminY
    }
```
Calculate the inverse scale factors:
```
   \fp_set:Nn \l__dataplot_inv_scale_x_fp
      { 1 / \l__dataplot_scale_x_fp }
   \fp_set:Nn \l__dataplot_inv_scale_y_fp
      { 1 / \l__dataplot_scale_y_fp }
 }
```
Transform data co-ordinates using the scales and offsets calculated above.
```
 \cs_new:Nn \dataplot_transform_data_coords:NN
 {
   \fp_set:Nn \l__dataplot_x_fp
    {
      \l__dataplot_scale_x_fp * #1 + \l__dataplot_offset_x_fp
    }
   \fp_set:Nn \l__dataplot_y_fp
    {
      \l__dataplot_scale_y_fp * #2 + \l__dataplot_offset_y_fp
    }
   \tl_set:Nx #1 { \fp_to_tl:N \l__dataplot_x_fp }
   \tl_set:Nx #2 { \fp_to_tl:N \l__dataplot_y_fp }
 }
```
Apply pgf transform using the scales and offsets calculated above.
```
 \cs_new:Nn \dataplot_apply_pgftransform:
 {
   \pgftransformcm
     { \fp_use:N \l__dataplot_scale_x_fp } { 0 }
     { 0 } { \fp_use:N \l__dataplot_scale_y_fp }
     {
       \pgfpoint
        { \fp_to_decimal:N \l__dataplot_offset_x_fp pt }
        { \fp_to_decimal:N \l__dataplot_offset_y_fp pt }
     }
 }
```
Apply inverse pgf transform.
```
 \cs_new:Nn \dataplot_apply_inverse_pgftransform:
 {
   \pgftransformcm
     { \fp_use:N \l__dataplot_inv_scale_x_fp } { 0 }
     { 0 } { \fp_use:N \l__dataplot_inv_scale_y_fp }
     {
       \pgfpoint
        {
          \fp_to_decimal:n
           { - \l__dataplot_offset_x_fp * \l__dataplot_inv_scale_x_fp }
          pt
        }
```

```
        {
          \fp_to_decimal:n
           { - \l__dataplot_offset_y_fp * \l__dataplot_inv_scale_y_fp }
          pt
        }
    }
}
```

\DTLplotstream

\DTLplotstream[⟨*condition*⟩]{⟨*db name*⟩}{⟨*x key*⟩}{⟨*y key*⟩}

Add points to a stream from the database called ⟨*db name*⟩ where the x co-ordinates are given by the key ⟨*x key*⟩ and the y co-ordinates are given by the key ⟨*y key*⟩. The optional argument ⟨*condition*⟩ is the same as that for \DTLforeach

```
\NewDocumentCommand \DTLplotstream { o m m m }
{
  \DTLmapdata [ name = { #2 } ]
  {
    \DTLmapgetvalues { \l__dataplot_x_tl = #3 , \l__dataplot_y_tl= #4 }
    \IfValueTF { #1 }
    {
      \ifthenelse { #1 } { \__dataplot_stream: } { }
    }
    {
      \__dataplot_stream:
    }
  }
}

\cs_new:Nn \__dataplot_stream:
{
  \DTLconverttodecimal
    { \l__dataplot_x_tl } { \l__dataplot_decimal_x_tl }
  \DTLconverttodecimal
    { \l__dataplot_y_tl } { \l__dataplot_decimal_y_tl }
  \dataplot_transform_data_coords:NN
    \l__dataplot_decimal_x_tl
    \l__dataplot_decimal_y_tl
  \pgfplotstreampoint
   {
     \pgfpointxy
      { \l__dataplot_decimal_x_tl }
      { \l__dataplot_decimal_y_tl }
   }
}
```

\DTLplotatbegintikz

\DTLplotatbegintikz is a hook to insert stuff at the start of the tikzpicture environment (after the unit vectors have been set).

```
\newcommand*{\DTLplotatbegintikz}{}
```

\dtlplothandlermark

```
\newcommand*{\dtlplothandlermark}[1]{
  \PackageWarning { dataplot }
   {
     \string \dtlplothandlermark \space ~
     found ~ outside ~ \string \DTLplot
   }
  \pgfplothandlermark { #1 }
}
```

\@dtlplothandlermark

```
\newcommand*{\@dtlplothandlermark}[1]{
  \pgftransformreset
  \tl_set:Nn \__dataplot_post_at_begin_tl
   {
     \dataplot_apply_pgftransform:
   }
  \pgfplothandlermark { #1 }
}
\tl_new:N \__dataplot_post_at_begin_tl
```

\DTLplotatendtikz `\DTLplotatendtikz` is a hook to insert stuff at the end of the tikzpicture environment.

```
\newcommand*{\DTLplotatendtikz}{}
```

Obtain the plot bounds, either from the settings (if provided) or calculate from the data. This will set token list variables `\DTLminX`, `\DTLminY`, `\DTLmaxX` and `\DTLmaxY` as well as floating point variables: `\l__dataplot_min_x_fp`, `\l__dataplot_min_y_fp`, `\l__dataplot_max_x_fp`, `\l__dataplot_max_y_fp` to avoid repeatedly converting between the two representations. The arguments are the column keys for the x and y data.

```
\cs_new:Nn \__dataplot_update_bounds:
{
  \bool_lazy_all:nTF
   {
     { \bool_not_p:n { \tl_if_empty_p:N \l__dataplot_min_x_tl } }
     { \bool_not_p:n { \tl_if_empty_p:N \l__dataplot_min_y_tl } }
     { \bool_not_p:n { \tl_if_empty_p:N \l__dataplot_max_x_tl } }
     { \bool_not_p:n { \tl_if_empty_p:N \l__dataplot_max_y_tl } }
   }
  {
```

Store the numbers as floating point variables to save repeated parsing.

```
    \datatool_set_fp:Nn \l__dataplot_min_x_fp { \l__dataplot_min_x_tl }
    \datatool_set_fp:Nn \l__dataplot_min_y_fp { \l__dataplot_min_y_tl }
    \datatool_set_fp:Nn \l__dataplot_max_x_fp { \l__dataplot_max_x_tl }
    \datatool_set_fp:Nn \l__dataplot_max_y_fp { \l__dataplot_max_y_tl }
```

Placeholders for use in hooks.

```
    \tl_set:Nx \DTLminX { \fp_to_tl:N \l__dataplot_min_x_fp }
    \tl_set:Nx \DTLminY { \fp_to_tl:N \l__dataplot_min_y_fp }
    \tl_set:Nx \DTLmaxX { \fp_to_tl:N \l__dataplot_max_x_fp }
    \tl_set:Nx \DTLmaxY { \fp_to_tl:N \l__dataplot_max_y_fp }
  }
  {
    \tl_set_eq:NN \l__dataplot_min_x_fp \c_novalue_tl
    \tl_set_eq:NN \l__dataplot_min_y_fp \c_novalue_tl
    \tl_set_eq:NN \l__dataplot_max_x_fp \c_novalue_tl
    \tl_set_eq:NN \l__dataplot_max_y_fp \c_novalue_tl
    \seq_map_inline:Nn \l__dataplot_dbnames_seq
    {
      \DTLsetup{ default-name = { ##1 } }
      \__dataplot_filtered_map:n
      {
        \datatool_set_fp:Nn \l__dataplot_x_fp
           { \l__dataplot_x_tl }
        \datatool_set_fp:Nn \l__dataplot_y_fp
           { \l__dataplot_y_tl }
        \exp_args:NV \tl_if_novalue:nTF \l__dataplot_min_x_fp
        {
          \tl_if_empty:NT \l__dataplot_min_x_tl
          {
            \fp_set_eq:NN
               \l__dataplot_min_x_fp
               \l__dataplot_x_fp
          }
        }
        {
          \fp_set:Nn \l__dataplot_min_x_fp
            { min ( \l__dataplot_min_x_fp , \l__dataplot_x_fp ) }
        }
        \exp_args:NV \tl_if_novalue:nTF \l__dataplot_min_y_fp
        {
          \tl_if_empty:NT \l__dataplot_min_y_tl
          {
            \fp_set_eq:NN
              \l__dataplot_min_y_fp
              \l__dataplot_y_fp
          }
        }
        {
          \fp_set:Nn \l__dataplot_min_y_fp
            { min ( \l__dataplot_min_y_fp , \l__dataplot_y_fp ) }
        }
        \exp_args:NV \tl_if_novalue:nTF \l__dataplot_max_x_fp
        {
          \tl_if_empty:NT \l__dataplot_max_x_tl
```

```
        {
          \fp_set_eq:NN
            \l__dataplot_max_x_fp
            \l__dataplot_x_fp
        }
      }
      {
        \fp_set:Nn \l__dataplot_max_x_fp
          { max ( \l__dataplot_max_x_fp , \l__dataplot_x_fp ) }
      }
      \exp_args:NV \tl_if_novalue:nTF \l__dataplot_max_y_fp
      {
        \tl_if_empty:NT \l__dataplot_max_y_tl
        {
          \fp_set_eq:NN
            \l__dataplot_max_y_fp
            \l__dataplot_y_fp
        }
      }
      {
        \fp_set:Nn \l__dataplot_max_y_fp
          { max ( \l__dataplot_max_y_fp , \l__dataplot_y_fp ) }
      }
    }
  }
```

Update `\DTLminX`

```
  \exp_args:NV \tl_if_novalue:nT \l__dataplot_min_x_fp
   {
     \tl_if_empty:NTF \l__dataplot_min_x_tl
     {
       \PackageError { dataplot }
       { No ~ minimum ~ X ~ available! }
       {
         Check ~ that ~ your ~ filter ~ criteria ~
         hasn't ~ excluded ~ all ~ data
       }
       \fp_zero:N \l__dataplot_min_x_fp
       \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
     }
     {
       \datatool_set_fp:Nn \l__dataplot_min_x_fp
         { \l__dataplot_min_x_tl }
     }
   }
  \tl_set:Nx \DTLminX { \fp_to_tl:N \l__dataplot_min_x_fp }
```

Update `\DTLminY`

```
  \exp_args:NV \tl_if_novalue:nT \l__dataplot_min_y_fp
   {
     \tl_if_empty:NTF \l__dataplot_min_y_tl
```

```
      {
        \PackageError { dataplot }
        { No ~ minimum ~ Y ~ available! }
        {
          Check ~ that ~ your ~ filter ~ criteria ~
          hasn't ~ excluded ~ all ~ data
        }
        \fp_zero:N \l__dataplot_min_y_fp
        \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
      }
      {
        \datatool_set_fp:Nn \l__dataplot_min_y_fp
          { \l__dataplot_min_y_tl }
      }
    }
  \tl_set:Nx \DTLminY { \fp_to_tl:N \l__dataplot_min_y_fp }
```

Update `\DTLmaxX`

```
  \exp_args:NV \tl_if_novalue:nT \l__dataplot_max_x_fp
   {
     \tl_if_empty:NTF \l__dataplot_max_x_tl
     {
       \PackageError { dataplot }
       { No ~ maximum ~ X ~ available! }
       {
         Check ~ that ~ your ~ filter ~ criteria ~
         hasn't ~ excluded ~ all ~ data
       }
       \fp_zero:N \l__dataplot_max_x_fp
       \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
     }
     {
       \datatool_set_fp:Nn \l__dataplot_max_x_fp
         { \l__dataplot_max_x_tl }
     }
   }
  \tl_set:Nx \DTLmaxX { \fp_use:N \l__dataplot_max_x_fp }
```

Update `\DTLmaxY`

```
  \exp_args:NV \tl_if_novalue:nT \l__dataplot_max_y_fp
   {
     \tl_if_empty:NTF \l__dataplot_max_y_tl
     {
       \PackageError { dataplot }
       { No ~ maximum ~ Y ~ available! }
       {
         Check ~ that ~ your ~ filter ~ criteria ~
         hasn't ~ excluded ~ all ~ data
       }
       \fp_zero:N \l__dataplot_max_y_fp
       \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
```

```
        }
        {
          \datatool_set_fp:Nn \l__dataplot_max_y_fp
            { \l__dataplot_max_y_tl }
        }
      }
      \tl_set:Nx \DTLmaxY { \fp_use:N \l__dataplot_max_y_fp }
    }
```

Don’t bother checking if an error has already occurred.

```
    \__dataplot_do_plot:n
    {
      \fp_compare:nNnT
        { \l__dataplot_min_x_fp } > { \l__dataplot_max_x_fp }
      {
         \PackageError {dataplot}
         {
           Min ~ X ~ ( \fp_use:N \l__dataplot_min_x_fp ) ~
           > ~ Max ~ X ~ ( \fp_use:N \l__dataplot_min_x_fp )
         }
         {
           If ~ you ~ have used ~ the ~ `bounds' ~ setting ~, check ~
           that ~ it ~ is ~ specified ~ as ~ minX,minY,maxX,maxY
         }
         \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
      }
      \fp_compare:nNnT
        { \l__dataplot_min_y_fp } > { \l__dataplot_max_y_fp }
      {
         \PackageError {dataplot}
         {
           Min ~ Y ~ ( \fp_use:N \l__dataplot_min_y_fp ) ~
           > ~ Max ~ Y ~ ( \fp_use:N \l__dataplot_min_y_fp )
         }
         {
           If ~ you ~ have used ~ the ~ `bounds' ~ setting ~, check ~
           that ~ it ~ is ~ specified ~ as ~ minX,minY,maxX,maxY
         }
         \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
      }
    }
}
```

```
\__dataplot_x_y_bounded:nnnnnnTF
{⟨x⟩}{⟨min x⟩}{⟨max x⟩}
{⟨y⟩}{⟨min y⟩}{⟨max y⟩}
{⟨true⟩}{⟨false⟩}
```

Test if x is between min x and max x and y is between min y and max y

```
\cs_new:Nn \__dataplot_x_y_bounded:nnnnnnTF
{
  \bool_lazy_any:nTF
   {
     { \fp_compare_p:n { #1 < #2 } }
     { \fp_compare_p:n { #1 > #3 } }
     { \fp_compare_p:n { #4 < #5 } }
     { \fp_compare_p:n { #4 > #6 } }
   }
  { #8 } { #7 }
}
\cs_new:Nn \__dataplot_x_y_bounded:nnnnnnT
{
  \__dataplot_x_y_bounded:nnnnnnTF
  { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } { #7 } { }
}
```

Tick lists.

```
\fp_new:N \l__dataplot_min_gap_fp
```

Calculate x tick gap or construct tick list if x tics required.

```
\cs_new:Nn \__dataplot_calc_x_tics:
{
 \legacy_if:nT { DTLxtics }
  {
    \clist_if_empty:NTF \l__dataplot_x_tic_clist
    {
      \seq_clear:N \l__dataplot_x_tic_seq
      \tl_if_empty:NTF \l__dataplot_xtic_gap_tl
      {
```

No tick list or gap set by user. Get the min tick gap in data co-ordinates

```
        \fp_set:Nn \l__dataplot_min_gap_fp
        {
          (
            ( \DTLmintickgap - \l__dataplot_offset_x_fp )
              / \l__dataplot_scale_x_fp
          )
          / 65536
        }
```

construct tick list

```
        \__dataplot_construct_tick_list:NNNN
          \l__dataplot_min_x_fp
          \l__dataplot_max_x_fp
          \l__dataplot_min_gap_fp
          \l__dataplot_x_tic_seq
      }
      {
```

Gap set by user

```
        \fp_set:Nn
```

```
          \l__dataplot_min_gap_fp
          { \l__dataplot_xtic_gap_tl }
        \fp_compare:nTF
          { \l__dataplot_min_x_fp < \c_zero_fp < \l__dataplot_max_x_fp }
        {
          \__dataplot_construct_tick_list_over_zero:NNNN
            \l__dataplot_min_x_fp
            \l__dataplot_max_x_fp
            \l__dataplot_x_tic_seq
            \l__dataplot_min_gap_fp
        }
        {
          \__dataplot_construct_tick_list_with_gap:NNNN
            \l__dataplot_min_x_fp
            \l__dataplot_max_x_fp
            \l__dataplot_x_tic_seq
            \l__dataplot_min_gap_fp
        }
      }
    }
    {
```

Check that the provided tick marks are within the bounds and save to sequence.

```
      \__dataplot_to_fp_seq:NNnn
        \l__dataplot_x_tic_seq
        \l__dataplot_x_tic_clist
        \l__dataplot_min_x_fp
        \l__dataplot_max_x_fp
    }
    \__dataplot_calc_minor_x_tics:
    \__dataplot_calc_x_label_dim:
  }
}
```

Minor x ticks:

```
\cs_new:Nn \__dataplot_calc_minor_x_tics:
{
  \dim_zero:N \l__dataplot_x_tic_label_height_dim
```

Construct a list of x minor tick points if required

```
  \seq_clear:N \l__dataplot_x_minor_tic_seq
  \legacy_if:nT { DTLxminortics }
   {
    \__dataplot_construct_minor_ticks:NNNNN
      \l__dataplot_x_tic_seq
      \l__dataplot_min_x_fp
      \l__dataplot_max_x_fp
      \l__dataplot_scale_x_fp
      \l__dataplot_x_minor_tic_seq
  }
}
```

__dataplot_get_default_tic_label:NN
⟨*fp-var*⟩⟨*int-var*⟩

Get the default tic label for position ⟨*fp-var*⟩ and rounding value ⟨*int-var*⟩

```
\cs_new:Nn \__dataplot_get_default_tic_label:Nn
{
  \fp_set:Nn \l__datatool_tmpa_fp
    {
      round ( #1 , #2 )
    }
  \tl_set:Nx \l__dataplot_tic_label_tl
   { \fp_to_decimal:N \l__datatool_tmpa_fp }
  \datatool_pad_trailing_zeros:Nn
    \l__dataplot_tic_label_tl { #2 }
  \exp_args:NV \DTLdecimaltolocale \l__dataplot_tic_label_tl
   \l__dataplot_tmpa_tl
  \tl_set:Nx \l__dataplot_tic_label_tl
   {
     \exp_not:N \__datatool_datum:nnnn
      { \exp_not:V \l__dataplot_tic_label_tl }
      { \exp_not:V \l__dataplot_tmpa_tl }
      { }
      { \c_datatool_decimal_int }
   }
}
\cs_generate_variant:Nn \__dataplot_get_default_tic_label:Nn
 { NV , Nx }
```

Determine the height of the x tick labels and save default tick labels if required.

```
\cs_new:Nn \__dataplot_calc_x_label_dim:
{
  \seq_if_empty:NTF \l__dataplot_xtic_labels_seq
  {
    \seq_map_inline:Nn \l__dataplot_x_tic_seq
     {
```

Tick label not provided, use the x value. Reconstruct the floating point variable:

```
        \tl_set:Nn \l__datatool_tmpa_fp { ##1 }
```

Obtain the default label:

```
        \__dataplot_get_default_tic_label:NV
          \l__datatool_tmpa_fp \c@DTLplotroundXvar
```

Save for later:

```
        \seq_put_right:NV \l__dataplot_xtic_labels_seq
          \l__dataplot_tic_label_tl
```

Calculate height:

```
        \settoheight \dtl@tmplength
          { \l__dataplot_tic_label_tl }
```

Compare with current max height:

```
        \dim_compare:nNnT
          { \dtl@tmplength }
           >
          { \l__dataplot_x_tic_label_height_dim }
         {
           \dim_set_eq:NN
             \l__dataplot_x_tic_label_height_dim \dtl@tmplength
         }
      }
   }
   {
     \seq_map_inline:Nn  \l__dataplot_xtic_labels_seq
      {
        \datatool_measure_height:Nn \dtl@tmplength
          { \DTLplotdisplayXticklabel { ##1 } }
        \dim_compare:nNnT
          { \dtl@tmplength } > { \l__dataplot_x_tic_label_height_dim }
         {
           \dim_set_eq:NN \l__dataplot_x_tic_label_height_dim \dtl@tmplength
         }
      }
   }
 }
```

Calculate y tick gap or construct tick list if y tics required.

```
 \cs_new:Nn \__dataplot_calc_y_tics:
 {
  \dim_zero:N \l__dataplot_y_tic_label_width_dim
  \legacy_if:nT { DTLytics }
   {
     \clist_if_empty:NTF \l__dataplot_y_tic_clist
     {
       \seq_clear:N \l__dataplot_y_tic_seq
```

List of y tics not provided.

```
       \tl_if_empty:NTF \l__dataplot_ytic_gap_tl
       {
```

No y tic gap not provided. Get the min tick gap in data co-ordinates.

```
         \fp_set:Nn \l__dataplot_min_gap_fp
         {
           (
             ( \DTLmintickgap - \l__dataplot_offset_y_fp )
               / \l__dataplot_scale_y_fp
           )
           / 65536
         }
```

Construct tick list

```
         \__dataplot_construct_tick_list:NNNN
```

```
          \l__dataplot_min_y_fp
          \l__dataplot_max_y_fp
          \l__dataplot_min_gap_fp
          \l__dataplot_y_tic_seq
      }
      {
```

Gap set by user

```
        \fp_set:Nn
         \l__dataplot_min_gap_fp
         { \l__dataplot_ytic_gap_tl }
        \fp_compare:nTF
          { \l__dataplot_min_y_fp < \c_zero_fp < \l__dataplot_max_y_fp }
        {
          \__dataplot_construct_tick_list_over_zero:NNNN
            \l__dataplot_min_y_fp
            \l__dataplot_max_y_fp
            \l__dataplot_y_tic_seq
            \l__dataplot_min_gap_fp
        }
        {
          \__dataplot_construct_tick_list_with_gap:NNNN
            \l__dataplot_min_y_fp
            \l__dataplot_max_y_fp
            \l__dataplot_y_tic_seq
            \l__dataplot_min_gap_fp
        }
      }
    }
    {
```

Check that the provided tick marks are within the bounds and save to sequence.

```
      \__dataplot_to_fp_seq:NNnn
        \l__dataplot_y_tic_seq
        \l__dataplot_y_tic_clist
        \l__dataplot_min_y_fp
        \l__dataplot_max_y_fp
    }
    \__dataplot_calc_minor_y_tics:
    \__dataplot_calc_y_label_dim:
  }
}
```

Minor y ticks:

```
\cs_new:Nn \__dataplot_calc_minor_y_tics:
{
 \seq_clear:N \l__dataplot_y_minor_tic_seq
 \legacy_if:nT { DTLyminortics }
  {
    \__dataplot_construct_minor_ticks:NNNNN
      \l__dataplot_y_tic_seq
```

```
      \l__dataplot_min_y_fp
      \l__dataplot_max_y_fp
      \l__dataplot_scale_y_fp
      \l__dataplot_y_minor_tic_seq
  }
}
```

Determine the width of the y tick labels and save default tick labels if required.

```
\cs_new:Nn \__dataplot_calc_y_label_dim:
{
  \seq_if_empty:NTF\l__dataplot_ytic_labels_seq
  {
    \seq_map_inline:Nn \l__dataplot_y_tic_seq
     {
```

Tick label not provided, use the y value. Reconstruct the floating point variable:

```
        \tl_set:Nn \l__datatool_tmpa_fp { ##1 }
```

Obtain the default label:

```
        \__dataplot_get_default_tic_label:NV
          \l__datatool_tmpa_fp \c@DTLplotroundYvar
```

Save for later:

```
        \seq_put_right:NV \l__dataplot_ytic_labels_seq
          \l__dataplot_tic_label_tl
```

Calculate width:

```
        \settowidth \dtl@tmplength
          { \l__dataplot_tic_label_tl }
```

Compare with current max width:

```
        \dim_compare:nNnT
           { \dtl@tmplength }
            >
           { \l__dataplot_y_tic_label_width_dim }
         {
           \dim_set_eq:NN \l__dataplot_y_tic_label_width_dim \dtl@tmplength
         }
     }
  }
  {
    \seq_map_inline:Nn  \l__dataplot_ytic_labels_seq
     {
       \datatool_measure_width:Nn \dtl@tmplength
         { \DTLplotdisplayYticklabel { ##1 } }
       \dim_compare:nNnT
           { \dtl@tmplength }
            >
           { \l__dataplot_y_tic_label_width_dim }
         {
           \dim_set_eq:NN \l__dataplot_y_tic_label_width_dim \dtl@tmplength
         }
```

```
    }
  }
}
```

Do the plot:

```
\cs_new:Nn \__dataplot_do_plot:
{
```

Initialise:

```
  \tl_clear:N \l__dataplot_content_tl
  \tl_clear:N \l_dataplot_legend_tl
```

Enable `\dtlplothandlermark` for the begin and end hooks.

```
  \tl_clear:N \__dataplot_post_at_begin_tl
  \tl_set_eq:NN \dtlplothandlermark \@dtlplothandlermark
```

Start the picture.

```
  \begin{tikzpicture}
```

Set the x and y unit vectors.

```
    \pgfsetxvec{\pgfpoint{1pt}{0pt}}%
    \pgfsetyvec{\pgfpoint{0pt}{1pt}}%
```

Set the transformation matrix.

```
    \dataplot_apply_pgftransform:
```

Initial hook.

```
    \DTLplotatbegintikz
    \__dataplot_post_at_begin_tl
```

Plot grid if required

```
    \__dataplot_add_grid:
```

Plot box or axes:

```
    \__dataplot_add_axes:
```

Plot x tics if required.

```
    \__dataplot_add_x_tics:
```

Plot x label if required.

```
    \__dataplot_add_x_label:
```

Plot the x minor ticks if required

```
    \__dataplot_add_minor_x:
```

Plot y tics if required.

```
    \__dataplot_add_y_tics:
```

Plot y label if required.

```
    \__dataplot_add_y_label:
```

Plot the y minor ticks if required

```
    \__dataplot_add_minor_y:
```

Do the accumulated plot instructions if any remaining:

```
    \l__dataplot_content_tl
    \tl_clear:N \l__dataplot_content_tl
```

Reset transformation matrix. (Don't want marker shapes to be scaled or skewed.)

```
    \begin{scope}
     \pgftransformreset
```

Draw marks and lines for each database and the legend:

```
     \__dataplot_plot_data:
```

Retain local assignments after scope:

```
     \tl_put_right:Nx \l__dataplot_content_tl
      {
```

End current scope:

```
        \exp_not:N \end{scope}
        \exp_not:N \tl_set:Nn \exp_not:N \DTLminX { \DTLminX }
        \exp_not:N \tl_set:Nn \exp_not:N \DTLminY { \DTLminY }
        \exp_not:N \tl_set:Nn \exp_not:N \DTLmaxX { \DTLmaxX }
        \exp_not:N \tl_set:Nn \exp_not:N \DTLmaxY { \DTLmaxY }
        \exp_not:N \int_set:Nn
         \exp_not:N \l_dataplot_stream_index_int
            { \int_use:N \l_dataplot_stream_index_int }
      }
     \l__dataplot_content_tl
     \tl_clear:N \l__dataplot_content_tl
```

End hook

```
     \DTLplotatendtikz
```

Retain local assignments after scope:

```
     \tl_put_right:Nx \l__dataplot_content_tl
      {
```

End tikzpicture environment:

```
        \exp_not:N \end{tikzpicture}
        \exp_not:N \tl_set:Nn \exp_not:N \DTLminX { \DTLminX }
        \exp_not:N \tl_set:Nn \exp_not:N \DTLminY { \DTLminY }
        \exp_not:N \tl_set:Nn \exp_not:N \DTLmaxX { \DTLmaxX }
        \exp_not:N \tl_set:Nn \exp_not:N \DTLmaxY { \DTLmaxY }
        \exp_not:N \int_set:Nn
         \exp_not:N \l_dataplot_stream_index_int
            { \int_use:N \l_dataplot_stream_index_int }
      }
     \l__dataplot_content_tl
     \tl_clear:N \l__dataplot_content_tl
}
```

Add box or axes:

```
\cs_new:Nn \__dataplot_add_axes:
{
```

Determine whether to put a box around the plot

```
  \legacy_if:nT { DTLbox }
   {
     \tl_put_right:Nx \l__dataplot_content_tl
```

```
      {
        \exp_not:N \draw
         (
           \fp_to_decimal:N \l__dataplot_extended_min_x_fp ,
           \fp_to_decimal:N \l__dataplot_extended_min_y_fp
         ) ~
          -- ~
         (
           \fp_to_decimal:N \l__dataplot_extended_max_x_fp ,
           \fp_to_decimal:N \l__dataplot_extended_min_y_fp
         ) ~
           -- ~
         (
           \fp_to_decimal:N \l__dataplot_extended_max_x_fp ,
           \fp_to_decimal:N \l__dataplot_extended_max_y_fp
         ) ~
           -- ~
         (
           \fp_to_decimal:N \l__dataplot_extended_min_x_fp ,
           \fp_to_decimal:N \l__dataplot_extended_max_y_fp
         ) ~
          -- ~ cycle ; ~
      }
  }
```

Plot x axis if required.

```
  \legacy_if:nT { DTLxaxis }
   {
     \tl_put_right:Nx \l__dataplot_content_tl
      {
        \exp_not:N \draw [ \exp_not:V \DTLXAxisStyle ]
      }
     \bool_if:NTF \l__dataplot_zero_y_axis_bool
      {
```

x axis passes through $y = 0$.

```
       \tl_put_right:Nx \l__dataplot_content_tl
        {
          (
            \fp_to_decimal:N \l__dataplot_extended_min_x_fp ,
           0 )
        }
       \tl_if_empty:NF \l__dataplot_x_min_label_tl
        {
          \tl_put_right:Nx \l__dataplot_content_tl
           {
             ~ node [ \exp_not:V \l__dataplot_x_min_label_style_tl ]
             { \exp_not:V \l__dataplot_x_min_label_tl }
           }
        }
       \tl_put_right:Nx \l__dataplot_content_tl
```

```
        {
          --
          (
           \fp_to_decimal:N \l__dataplot_extended_max_x_fp ,
          0 )
        }
      }
      {
```

x axis passes through $y = \min_y$.

```
       \tl_put_right:Nx \l__dataplot_content_tl
       {
         (
           \fp_to_decimal:N \l__dataplot_extended_min_x_fp ,
           \DTLminY
         )
       }
       \tl_if_empty:NF \l__dataplot_x_min_label_tl
       {
         \tl_put_right:Nx \l__dataplot_content_tl
          {
            ~ node [ \exp_not:V \l__dataplot_x_min_label_style_tl ]
            { \exp_not:V \l__dataplot_x_min_label_tl }
          }
       }
       \tl_put_right:Nx \l__dataplot_content_tl
       {
         --
         (
          \fp_to_decimal:N \l__dataplot_extended_max_x_fp ,
          \DTLminY
         )
       }
     }
     \tl_if_empty:NF \l__dataplot_x_max_label_tl
      {
        \tl_put_right:Nx \l__dataplot_content_tl
         {
           ~ node [ \exp_not:V \l__dataplot_x_max_label_style_tl ]
           { \exp_not:V \l__dataplot_x_max_label_tl }
         }
      }
     \tl_put_right:Nn \l__dataplot_content_tl
      { ; ~ }
  }
```

Plot y axis if required.

```
  \legacy_if:nT { DTLyaxis }
   {
     \tl_put_right:Nx \l__dataplot_content_tl
      {
```

```
      \exp_not:N \draw [ \exp_not:V \DTLYAxisStyle ]
    }
  \bool_if:NTF \l__dataplot_zero_x_axis_bool
   {
```
y axis passes through $x = 0$.
```
    \tl_put_right:Nx \l__dataplot_content_tl
     {
       (0,
         \fp_to_decimal:N \l__dataplot_extended_min_y_fp
       )
     }
    \tl_if_empty:NF \l__dataplot_y_min_label_tl
     {
       \tl_put_right:Nx \l__dataplot_content_tl
        {
          ~ node [ \exp_not:V \l__dataplot_y_min_label_style_tl ]
          { \exp_not:V \l__dataplot_y_min_label_tl }
        }
     }
    \tl_put_right:Nx \l__dataplot_content_tl
     {
        --
       (0,
        \fp_to_decimal:N \l__dataplot_extended_max_y_fp
       )
     }
   }
   {
```
y axis passes through $x = \min_x$.
```
    \tl_put_right:Nx \l__dataplot_content_tl
     {
       (\DTLminX,
         \fp_to_decimal:N \l__dataplot_extended_min_y_fp
       )
     }
    \tl_if_empty:NF \l__dataplot_y_min_label_tl
     {
       \tl_put_right:Nx \l__dataplot_content_tl
        {
          ~ node [ \exp_not:V \l__dataplot_y_min_label_style_tl ]
          { \exp_not:V \l__dataplot_y_min_label_tl }
        }
     }
    \tl_put_right:Nx \l__dataplot_content_tl
     {
        --
       (\DTLminX,
         \fp_to_decimal:N \l__dataplot_extended_max_y_fp
       )
```

```
      }
    }
   \tl_if_empty:NF \l__dataplot_y_max_label_tl
    {
      \tl_put_right:Nx \l__dataplot_content_tl
       {
         ~ node [ \exp_not:V \l__dataplot_y_max_label_style_tl ]
         { \exp_not:V \l__dataplot_y_max_label_tl }
       }
    }
   \tl_put_right:Nn \l__dataplot_content_tl
    { ; ~ }
  }
 \l__dataplot_content_tl
 \tl_clear:N \l__dataplot_content_tl
}
```

Plot grid if required:

```
\cs_new:Nn \__dataplot_add_grid:
{
  \legacy_if:nT { DTLgrid }
   {
```

Draw minor grid lines if minor ticks enabled and if there is a line style set.

```
    \tl_if_empty:NF \DTLminorgridstyle
     {
       \legacy_if:nT { DTLxminortics }
        {
```

Iterate through minor x ticks.

```
          \seq_map_inline:Nn \l__dataplot_x_minor_tic_seq
           {
```

Reconstruct floating point variable.

```
            \tl_set:Nn \l__dataplot_x_fp { ##1 }
            \tl_set:Nx \l__dataplot_x_tl
              { \fp_to_decimal:N \l__dataplot_x_fp }
```

Draw minor grid line.

```
            \tl_put_right:Nx \l__dataplot_content_tl
             {
               \exp_not:N \draw [ \exp_not:V \DTLminorgridstyle ]
               ( \l__dataplot_x_tl ,
                 \fp_to_decimal:N \l__dataplot_extended_min_y_fp
               )
               --
               ( \l__dataplot_x_tl,
                 \fp_to_decimal:N \l__dataplot_extended_max_y_fp
               ); ~
             }
           }
        }
```

```
      \legacy_if:nT { DTLyminortics }
       {
```
Iterate through minor y ticks.
```
        \seq_map_inline:Nn \l__dataplot_y_minor_tic_seq
         {
```
Reconstruct floating point variable.
```
          \tl_set:Nn \l__dataplot_y_fp { ##1 }
          \tl_set:Nx \l__dataplot_y_tl
            { \fp_to_decimal:N \l__dataplot_y_fp }
```
Draw minor grid line.
```
          \tl_put_right:Nx \l__dataplot_content_tl
           {
             \exp_not:N \draw [ \exp_not:V \DTLminorgridstyle ]
             (
               \fp_to_decimal:N \l__dataplot_extended_min_x_fp ,
               \l__dataplot_y_tl
             )
             --
             (
               \fp_to_decimal:N \l__dataplot_extended_max_x_fp ,
               \l__dataplot_y_tl
             ); ~
           }
         }
       }
    }
```
Draw major grid lines.
```
    \seq_map_inline:Nn \l__dataplot_x_tic_seq
     {
       \tl_set:Nn \l__dataplot_x_fp { ##1 }
       \tl_set:Nx \l__dataplot_x_tl
         { \fp_to_decimal:N \l__dataplot_x_fp }
       \tl_put_right:Nx \l__dataplot_content_tl
       {
         \exp_not:N \draw [ \exp_not:V \DTLmajorgridstyle ]
         ( \l__dataplot_x_tl ,
           \fp_to_decimal:N \l__dataplot_extended_min_y_fp
         )
         --
         ( \l__dataplot_x_tl ,
           \fp_to_decimal:N \l__dataplot_extended_max_y_fp
         ); ~
       }
     }
    \seq_map_inline:Nn \l__dataplot_y_tic_seq
    {
      \tl_set:Nn \l__dataplot_y_fp { ##1 }
```

```
      \tl_set:Nx \l__dataplot_y_tl
       { \fp_to_decimal:N \l__dataplot_y_fp }
      \tl_put_right:Nx \l__dataplot_content_tl
       {
         \exp_not:N \draw [ \exp_not:V \DTLmajorgridstyle ]
         (
           \fp_to_decimal:N  \l__dataplot_extended_min_x_fp ,
           \l__dataplot_y_tl
         )
          --
         (
           \fp_to_decimal:N \l__dataplot_extended_max_x_fp ,
           \l__dataplot_y_tl
         ); ~
       }
    }
   \l__dataplot_content_tl
   \tl_clear:N \l__dataplot_content_tl
  }
}
```

Plot x tics if required:

```
\cs_new:Nn \__dataplot_add_x_tics:
{
 \legacy_if:nT { DTLxtics }
  {
```

Get tick length in terms of canvas co-ordinates

```
    \fp_set:Nn \l__dataplot_tick_length_fp
     {
       (
         (
           \dim_to_decimal_in_sp:n { \DTLticklength }
            - \l__dataplot_offset_y_fp
         )   / \l__dataplot_scale_y_fp
       )
       / 65536
     }
```

Get tick label offset in terms of canvas co-ordinates

```
     \dim_add:Nn \l__dataplot_x_tic_label_height_dim
       { \DTLticklabeloffset }
     \fp_set:Nn \l__dataplot_tic_label_offset_fp
       {
         (
           (
             \dim_to_decimal_in_sp:n
                { \l__dataplot_x_tic_label_height_dim }
             + \l__dataplot_offset_y_fp
           ) / \l__dataplot_scale_y_fp
         ) / 65536
```

```
    }
```
Iterate through tick list. The default tick labels should have already been added in `\__dataplot_calc_x_label_dim:`
```
  \seq_map_indexed_inline:Nn \l__dataplot_x_tic_seq
    {
```
Reconstruct the floating point variable:
```
      \tl_set:Nn \l__dataplot_x_fp { ##2 }
      \tl_set:Nn \l__dataplot_x_tl
        { \fp_use:N \l__dataplot_x_fp }
```
Fetch current tick label:
```
      \tl_set:Nx \l__dataplot_tic_label_tl
      {
        \seq_item:Nn \l__dataplot_xtic_labels_seq { ##1 }
      }
```
Draw tick.
```
    \legacy_if:nTF { DTLxticsin }
     {
```
Tick above the x-axis. (Tick `in`)
```
      \bool_if:NTF \l__dataplot_zero_y_axis_bool
       {
```
Enforce x axis to pass through $y = 0$.
```
          \fp_set_eq:NN \l__dataplot_y_fp
            \l__dataplot_tick_length_fp
          \tl_put_right:Nx \l__dataplot_content_tl
            {
              \exp_not:N \draw
               ( \l__dataplot_x_tl , 0 )
                --
               ( \l__dataplot_x_tl ,
                   \fp_to_decimal:N  \l__dataplot_y_fp ) ; ~
            }
          \fp_set:Nn \l__dataplot_y_fp
            { - \l__dataplot_tic_label_offset_fp }
       }
       {
```
Don't enforce x axis to pass through $y = 0$ (but this will occur if min y = 0).
```
          \fp_set:Nn \l__dataplot_y_fp
            {
              \l__dataplot_min_y_fp
               + \l__dataplot_tick_length_fp
            }
          \tl_put_right:Nx \l__dataplot_content_tl
            {
              \exp_not:N \draw
               ( \l__dataplot_x_tl , \DTLminY )
```

```
            --
           ( \l__dataplot_x_tl ,
             \fp_to_decimal:N \l__dataplot_y_fp ) ; ~
        }
       \fp_set:Nn \l__dataplot_y_fp
        {
          \l__dataplot_min_y_fp
          - \l__dataplot_tic_label_offset_fp
        }
     }
```

Draw the label unless 0 and omit zero label on.

```
    \bool_lazy_and:nnF
      { \l__dataplot_omit_zero_x_label_bool }
      {
        \fp_compare_p:nNn
          { \l__dataplot_x_fp } = { \c_zero_fp }
      }
    {
      \tl_put_right:Nx \l__dataplot_content_tl
       {
         \exp_not:N \draw
          ( \l__dataplot_x_tl ,
            \fp_to_decimal:N \l__dataplot_y_fp ) ~
          node [ \exp_not:V \l__dataplot_x_tick_label_style_tl ]
           {
             \exp_not:N \DTLplotdisplayXticklabel
              { \exp_not:V \l__dataplot_tic_label_tl }
           } ; ~
       }
    }
  }
  {
```

Tick below the x-axis. (Tick `out`)

```
    \bool_if:NTF \l__dataplot_zero_y_axis_bool
     {
      \fp_set:Nn \l__dataplot_y_fp
        { - \l__dataplot_tick_length_fp }
      \tl_put_right:Nx \l__dataplot_content_tl
        {
          \exp_not:N \draw
           ( \l__dataplot_x_tl , 0 )
            --
           ( \l__dataplot_x_tl ,
               \fp_to_decimal:N \l__dataplot_y_fp ) ; ~
        }
      \fp_sub:Nn \l__dataplot_y_fp
        { \l__dataplot_tic_label_offset_fp }
     }
     {
```

Don’t enforce x axis to pass through $y = 0$ (but this will occur if min y = 0).

```
\fp_set:Nn \l__dataplot_y_fp
 {
   \l__dataplot_min_y_fp
    - \l__dataplot_tick_length_fp
 }
\tl_put_right:Nx \l__dataplot_content_tl
  {
    \exp_not:N \draw
     ( \l__dataplot_x_tl , \DTLminY )
      --
     ( \l__dataplot_x_tl ,
         \fp_to_decimal:N \l__dataplot_y_fp ) ;
  }
\fp_sub:Nn \l__dataplot_y_fp
  { \l__dataplot_tic_label_offset_fp }
}
```

Draw the label unless 0 and omit zero label on.

```
\bool_lazy_and:nnF
 { \l__dataplot_omit_zero_x_label_bool }
 { \fp_compare_p:nNn { \l__dataplot_x_fp } = { \c_zero_fp } }
{
  \tl_put_right:Nx \l__dataplot_content_tl
    {
       \exp_not:N \draw
       ( \l__dataplot_x_tl ,
            \fp_to_decimal:N \l__dataplot_y_fp ) ~
       node [ \exp_not:V \l__dataplot_x_tick_label_style_tl ]
        {
          \exp_not:N \DTLplotdisplayXticklabel
           { \exp_not:V \l__dataplot_tic_label_tl }
        } ; ~
    }
  }
}
```

Draw box ticks, if box setting is on.

```
\legacy_if:nT { DTLbox }
 {
```

If min y isn’t 0 and the box setting is on, need to draw tick on the bottom of the box according to the box setting.

```
\__dataplot_draw_box_ticks_lower_x:n
 {
   \int_case:nnT { \l__dataplot_box_ticks_int }
    {
      { \c_one_int } % match
       {
         \legacy_if:nTF { DTLxticsin }
         {
```

```
                \fp_set:Nn \l__dataplot_y_fp
                 {
                   \l__dataplot_extended_min_y_fp
                    + \l__dataplot_tick_length_fp
                 }
               }
               {
                 \fp_set:Nn \l__dataplot_y_fp
                  {
                    \l__dataplot_extended_min_y_fp
                    - \l__dataplot_tick_length_fp
                  }
               }
             }
             { 2 } % in
             {
               \fp_set:Nn \l__dataplot_y_fp
                {
                  \l__dataplot_extended_min_y_fp
                  + \l__dataplot_tick_length_fp
                }
             }
             { 3 } % out
             {
               \fp_set:Nn \l__dataplot_y_fp
                {
                  \l__dataplot_extended_min_y_fp
                   - \l__dataplot_tick_length_fp
                }
             }
          }
          {
            \tl_put_right:Nx \l__dataplot_content_tl
             {
               \exp_not:N \draw
               ( \l__dataplot_x_tl ,
                 \fp_to_decimal:N \l__dataplot_extended_min_y_fp )
                --
               ( \l__dataplot_x_tl ,
                  \fp_to_decimal:N \l__dataplot_y_fp ) ; ~
             }
          }
       }
```

Draw the ticks on the opposite side of the box.

```
      \int_case:nnT { \l__dataplot_box_ticks_int }
       {
         { \c_one_int } % match
          {
            \legacy_if:nTF  { DTLxticsin }
```

```
                {
                  \fp_set:Nn \l__dataplot_y_fp
                   {
                     \l__dataplot_extended_max_y_fp
                     - \l__dataplot_tick_length_fp
                   }
                }
                {
                  \fp_set:Nn \l__dataplot_y_fp
                   {
                     \l__dataplot_extended_max_y_fp
                     + \l__dataplot_tick_length_fp
                   }
                }
              }
             { 2 } % in
              {
                \fp_set:Nn \l__dataplot_y_fp
                 {
                   \l__dataplot_extended_max_y_fp
                   - \l__dataplot_tick_length_fp
                 }
              }
             { 3 } % out
              {
                \fp_set:Nn \l__dataplot_y_fp
                 {
                   \l__dataplot_extended_max_y_fp
                    + \l__dataplot_tick_length_fp
                 }
              }
          }
          {
            \tl_put_right:Nx \l__dataplot_content_tl
             {
               \exp_not:N \draw
               ( \l__dataplot_x_tl ,
                 \fp_to_decimal:N \l__dataplot_extended_max_y_fp )
                --
               ( \l__dataplot_x_tl ,
                 \fp_to_decimal:N \l__dataplot_y_fp ) ; ~
             }
          }
      }
    }
   \l__dataplot_content_tl
   \tl_clear:N \l__dataplot_content_tl
  }
}
```

Plot x label if required.

```
\cs_new:Nn \__dataplot_add_x_label:
{
  \tl_if_empty:NF \l__dataplot_xlabel_tl
  {
```

Get baseline in terms of canvas co-ordinates

```
    \fp_add:Nn \l__dataplot_tic_label_offset_fp
      {
        (
          (
            \dim_to_decimal_in_sp:n { \baselineskip }
             + \l__dataplot_offset_y_fp
          ) / \l__dataplot_scale_y_fp
        ) / 65536
      }
```

Get halfway position

```
    \fp_set:Nn \l__dataplot_x_fp
      { \l__dataplot_min_x_fp + 0.5 * \l__dataplot_x_extent_fp  }
    \bool_if:NTF \l__dataplot_zero_y_axis_bool
     {
      \fp_set:Nn \l__dataplot_y_fp
        { - \l__dataplot_tic_label_offset_fp }
     }
     {
      \fp_set:Nn \l__dataplot_y_fp
        { \l__dataplot_min_y_fp - \l__dataplot_tic_label_offset_fp }
     }
    \tl_put_right:Nx \l__dataplot_content_tl
      {
        \exp_not:N \draw
        ( \fp_to_decimal:N \l__dataplot_x_fp ,
          \fp_to_decimal:N \l__dataplot_y_fp ) ~
        node[anchor=north] { \exp_not:V \l__dataplot_xlabel_tl }; ~
      }
    \l__dataplot_content_tl
    \tl_clear:N \l__dataplot_content_tl
  }
}
```

Plot the x minor ticks if required

```
\cs_new:Nn \__dataplot_add_minor_x:
{
 \legacy_if:nT { DTLxminortics }
  {
```

Get tick length in terms of canvas co-ordinates

```
    \fp_set:Nn \l__dataplot_tick_length_fp
     {
       (
```

```
        (
          \dim_to_decimal_in_sp:n { \DTLminorticklength }
           + \l__dataplot_offset_y_fp
        ) / \l__dataplot_scale_y_fp
      ) / 65536
    }
```

Iterate through minor ticks.

```
  \seq_map_inline:Nn \l__dataplot_x_minor_tic_seq
   {
```

Reconstruct floating point variable.

```
      \tl_set:Nn \l__dataplot_x_fp { ##1 }
      \tl_set:Nx \l__dataplot_x_tl
        { \fp_to_decimal:N \l__dataplot_x_fp }
```

Draw tick.

```
      \legacy_if:nTF { DTLxticsin }
       {
```

Tick above the x-axis. (Tick `in`)

```
        \bool_if:NTF \l__dataplot_zero_y_axis_bool
         {
```

Enforce x axis to pass through $y = 0$.

```
           \fp_set_eq:NN \l__dataplot_y_fp
              \l__dataplot_tick_length_fp
           \tl_put_right:Nx \l__dataplot_content_tl
            {
              \exp_not:N \draw
              ( \l__dataplot_x_tl , 0 )
               --
              ( \l__dataplot_x_tl ,
                 \fp_to_decimal:N \l__dataplot_y_fp ); ~
            }
          }
          {
```

Don't enforce x axis to pass through $y = 0$ (but this will occur if min y = 0).

```
           \fp_set:Nn \l__dataplot_y_fp
            {
              \l__dataplot_min_y_fp
               + \l__dataplot_tick_length_fp
            }
           \tl_put_right:Nx \l__dataplot_content_tl
            {
              \exp_not:N \draw
              ( \l__dataplot_x_tl , \DTLminY )
               --
              ( \l__dataplot_x_tl ,
                 \fp_to_decimal:N \l__dataplot_y_fp ); ~
            }
```

```
      }
    }
    {
```

Tick below the x-axis. (Tick `out`)

```
      \bool_if:NTF \l__dataplot_zero_y_axis_bool
       {
        \fp_set:Nn \l__dataplot_y_fp
          { - \l__dataplot_tick_length_fp }
        \tl_put_right:Nx \l__dataplot_content_tl
         {
           \exp_not:N \draw
           ( \l__dataplot_x_tl , 0 )
            --
           ( \l__dataplot_x_tl ,
              \fp_to_decimal:N \l__dataplot_y_fp ); ~
         }
       }
       {
```

Don't enforce x axis to pass through $y = 0$ (but this will occur if min y = 0).

```
        \fp_set:Nn \l__dataplot_y_fp
          {
            \l__dataplot_min_y_fp
             - \l__dataplot_tick_length_fp
          }
        \tl_put_right:Nx \l__dataplot_content_tl
         {
           \exp_not:N \draw
           ( \l__dataplot_x_tl , \DTLminY )
            --
           ( \l__dataplot_x_tl ,
              \fp_to_decimal:N \l__dataplot_y_fp ); ~
         }
       }
    }
```

Draw box ticks, if box setting is on.

```
    \legacy_if:nT { DTLbox }
     {
```

If min y isn't 0 and the box setting is on, need to draw tick on the bottom of the box according to the box setting.

```
        \__dataplot_draw_box_ticks_lower_x:n
         {
           \int_case:nnT { \l__dataplot_box_ticks_int }
            {
              { \c_one_int } % match
               {
                 \legacy_if:nTF { DTLxticsin }
                 {
```

```
                \fp_set:Nn \l__dataplot_y_fp
                 {
                   \l__dataplot_extended_min_y_fp
                    + \l__dataplot_tick_length_fp
                 }
               }
               {
                 \fp_set:Nn \l__dataplot_y_fp
                 {
                   \l__dataplot_extended_min_y_fp
                   - \l__dataplot_tick_length_fp
                 }
               }
             }
             { 2 } % in
             {
               \fp_set:Nn \l__dataplot_y_fp
                {
                  \l__dataplot_extended_min_y_fp
                  + \l__dataplot_tick_length_fp
                }
             }
             { 3 } % out
             {
               \fp_set:Nn \l__dataplot_y_fp
                {
                  \l__dataplot_extended_min_y_fp
                   - \l__dataplot_tick_length_fp
                }
             }
           }
           {
             \tl_put_right:Nx \l__dataplot_content_tl
              {
                \exp_not:N \draw
                ( \l__dataplot_x_tl ,
                  \fp_to_decimal:N \l__dataplot_extended_min_y_fp )
                 --
                ( \l__dataplot_x_tl ,
                   \fp_to_decimal:N \l__dataplot_y_fp ); ~
              }
           }
       }
```

Draw the ticks on the opposite side of the box.

```
      \int_case:nnT { \l__dataplot_box_ticks_int }
       {
         { \c_one_int } % match
          {
            \legacy_if:nTF  { DTLxticsin }
```

```
                {
                  \fp_set:Nn \l__dataplot_y_fp
                   {
                     \l__dataplot_extended_max_y_fp
                     - \l__dataplot_tick_length_fp
                   }
                }
                {
                  \fp_set:Nn \l__dataplot_y_fp
                   {
                     \l__dataplot_extended_max_y_fp
                     + \l__dataplot_tick_length_fp
                   }
                }
              }
             { 2 } % in
              {
                \fp_set:Nn \l__dataplot_y_fp
                 {
                   \l__dataplot_extended_max_y_fp
                   - \l__dataplot_tick_length_fp
                 }
              }
             { 3 } % out
              {
                \fp_set:Nn \l__dataplot_y_fp
                 {
                   \l__dataplot_extended_max_y_fp
                    + \l__dataplot_tick_length_fp
                 }
              }
          }
          {
            \tl_put_right:Nx \l__dataplot_content_tl
             {
               \exp_not:N \draw
               ( \l__dataplot_x_tl ,
                 \fp_to_decimal:N \l__dataplot_extended_max_y_fp )
                --
               ( \l__dataplot_x_tl ,
                 \fp_to_decimal:N \l__dataplot_y_fp ); ~
             }
          }
       }
    }
   \l__dataplot_content_tl
   \tl_clear:N \l__dataplot_content_tl
  }
}
```

Plot y tics if required:

```
\cs_new:Nn \__dataplot_add_y_tics:
{
 \legacy_if:nT { DTLytics }
  {
```

Get tick length in terms of canvas co-ordinates

```
    \fp_set:Nn \l__dataplot_tick_length_fp
     {
       (
         (
           \dim_to_decimal_in_sp:n { \DTLticklength }
            + \l__dataplot_offset_x_fp
         )   / \l__dataplot_scale_x_fp
       )
       / 65536
     }
```

Get tick label offset in terms of canvas co-ordinates

```
    \fp_set:Nn \l__dataplot_tic_label_offset_fp
      {
        (
          (
            \dim_to_decimal_in_sp:n { \DTLticklabeloffset }
            + \l__dataplot_offset_x_fp
          ) / \l__dataplot_scale_x_fp
        ) / 65536
      }
```

Iterate through tick list. The default tick labels should have already been added in `\__dataplot_calc_y_label_dim:`

```
    \seq_map_indexed_inline:Nn  \l__dataplot_y_tic_seq
      {
```

Reconstruct the floating point variable:

```
        \tl_set:Nn \l__dataplot_y_fp { ##2 }
        \tl_set:Nn \l__dataplot_y_tl
          { \fp_use:N \l__dataplot_y_fp }
```

Fetch current tick label:

```
        \tl_set:Nx \l__dataplot_tic_label_tl
        {
          \seq_item:Nn \l__dataplot_ytic_labels_seq { ##1 }
        }
```

Draw tick.

```
      \legacy_if:nTF { DTLyticsin }
       {
```

Tick to the right of the y-axis. (Tick `in`)

```
        \bool_if:NTF \l__dataplot_zero_x_axis_bool
         {
```

Enforce y axis to pass through $x = 0$.

```
      \fp_set_eq:NN \l__dataplot_x_fp
          \l__dataplot_tick_length_fp
      \tl_put_right:Nx \l__dataplot_content_tl
        {
          \exp_not:N \draw
           ( 0,  \l__dataplot_y_tl )
            --
           ( \fp_to_decimal:N  \l__dataplot_x_fp,
               \l__dataplot_y_tl ); ~
        }
      \fp_set:Nn \l__dataplot_x_fp
        { - \l__dataplot_tic_label_offset_fp }
    }
    {
```

Don't enforce y axis to pass through $x = 0$ (but this will occur if min x = 0).

```
      \fp_set:Nn \l__dataplot_x_fp
        {
          \l__dataplot_min_x_fp
          + \l__dataplot_tick_length_fp
        }
      \tl_put_right:Nx \l__dataplot_content_tl
        {
          \exp_not:N \draw
           (
             \DTLminX , \l__dataplot_y_tl
           )
            --
           ( \fp_to_decimal:N  \l__dataplot_x_fp,
               \l__dataplot_y_tl ); ~
        }
      \fp_set:Nn \l__dataplot_x_fp
        {
          \l__dataplot_min_x_fp
           - \l__dataplot_tic_label_offset_fp
        }
    }
```

Draw the label unless 0 and omit zero label on.

```
  \bool_lazy_and:nnF
    { \l__dataplot_omit_zero_y_label_bool }
    {
      \fp_compare_p:nNn
        { \l__dataplot_y_fp } = { \c_zero_fp }
    }
  {
    \tl_put_right:Nx \l__dataplot_content_tl
      {
        \exp_not:N \draw
```

```
            ( \fp_to_decimal:N \l__dataplot_x_fp,
                 \l__dataplot_y_tl ) ~
            node [ \exp_not:V \l__dataplot_y_tick_label_style_tl ]
             {
               \exp_not:N \DTLplotdisplayYticklabel
                { \exp_not:V \l__dataplot_tic_label_tl }
             }; ~
          }
        }
      }
      {
```

Tick to the left of the y-axis. (Tick `out`)

```
      \bool_if:NTF \l__dataplot_zero_x_axis_bool
       {
         \fp_set:Nn \l__dataplot_x_fp
           { - \l__dataplot_tick_length_fp }
         \tl_put_right:Nx \l__dataplot_content_tl
           {
             \exp_not:N \draw
              ( 0 ,  \l__dataplot_y_tl )
               --
              ( \fp_to_decimal:N \l__dataplot_x_fp,
                  \l__dataplot_y_tl  ) ; ~
           }
       }
       {
```

Don't enforce y axis to pass through $x = 0$ (but this will occur if min x = 0).

```
         \fp_set:Nn \l__dataplot_x_fp
           {
             \l__dataplot_min_x_fp
              - \l__dataplot_tick_length_fp
           }
         \tl_put_right:Nx \l__dataplot_content_tl
           {
             \exp_not:N \draw
              (
                \DTLminX , \l__dataplot_y_tl
              )
               --
              ( \fp_to_decimal:N \l__dataplot_x_fp,
                  \l__dataplot_y_tl  ) ; ~
           }
       }
```

Draw the label unless 0 and omit zero label on.

```
      \bool_lazy_and:nnF
        { \l__dataplot_omit_zero_y_label_bool }
        {
          \fp_compare_p:nNn
```

```
        { \l__dataplot_y_fp } = { \c_zero_fp }
    }
  {
    \fp_sub:Nn \l__dataplot_x_fp
      { \l__dataplot_tic_label_offset_fp }
    \tl_put_right:Nx \l__dataplot_content_tl
      {
        \exp_not:N \draw
         ( \fp_to_decimal:N \l__dataplot_x_fp,
              \l__dataplot_y_tl ) ~
         node [ \exp_not:V \l__dataplot_y_tick_label_style_tl ]
          {
            \exp_not:N \DTLplotdisplayYticklabel
             { \exp_not:V \l__dataplot_tic_label_tl }
          } ; ~
      }
    }
}
```

Draw box ticks, if box setting is on.

```
\legacy_if:nT { DTLbox }
 {
```

If min x isn't 0 and the box setting is on, need to draw tick on the left of the box according to the box setting.

```
    \__dataplot_draw_box_ticks_lower_y:n
     {
       \int_case:nnT { \l__dataplot_box_ticks_int }
        {
          { \c_one_int } % match
           {
             \legacy_if:nTF { DTLyticsin }
             {
               \fp_set:Nn \l__dataplot_x_fp
                {
                  \l__dataplot_extended_min_x_fp
                  + \l__dataplot_tick_length_fp
                }
             }
             {
               \fp_set:Nn \l__dataplot_x_fp
                {
                  \l__dataplot_extended_min_x_fp
                  - \l__dataplot_tick_length_fp
                }
             }
           }
          { 2 } % in
           {
             \fp_set:Nn \l__dataplot_x_fp
              {
```

```
                \l__dataplot_extended_min_x_fp
                 + \l__dataplot_tick_length_fp
              }
            }
           { 3 } % out
            {
              \fp_set:Nn \l__dataplot_x_fp
               {
                 \l__dataplot_extended_min_x_fp
                  - \l__dataplot_tick_length_fp
               }
            }
         }
         {
           \tl_put_right:Nx \l__dataplot_content_tl
            {
              \exp_not:N \draw
              (
                \fp_to_decimal:N \l__dataplot_extended_min_x_fp ,
                \l__dataplot_y_tl
               )
               --
              ( \fp_to_decimal:N \l__dataplot_x_fp,
                  \l__dataplot_y_tl ); ~
            }
         }
     }
```

Draw opposite tick.

```
    \int_case:nnT { \l__dataplot_box_ticks_int }
     {
       { \c_one_int } % match
        {
          \legacy_if:nTF { DTLyticsin }
           {
            \fp_set:Nn \l__dataplot_x_fp
             {
               \l__dataplot_extended_max_x_fp
                - \l__dataplot_tick_length_fp
             }
           }
           {
            \fp_set:Nn \l__dataplot_x_fp
             {
               \l__dataplot_extended_max_x_fp
               + \l__dataplot_tick_length_fp
             }
           }
         }
       { 2 } % in
```

```
              {
                \fp_set:Nn \l__dataplot_x_fp
                 {
                   \l__dataplot_extended_max_x_fp
                     - \l__dataplot_tick_length_fp
                 }
              }
             { 3 } % out
              {
                \fp_set:Nn \l__dataplot_x_fp
                 {
                   \l__dataplot_extended_max_x_fp
                   + \l__dataplot_tick_length_fp
                 }
              }
           }
           {
             \tl_put_right:Nx \l__dataplot_content_tl
              {
                \exp_not:N \draw
                (
                  \fp_to_decimal:N \l__dataplot_extended_max_x_fp ,
                  \l__dataplot_y_tl
                )
                 --
                ( \fp_to_decimal:N \l__dataplot_x_fp,
                    \l__dataplot_y_tl); ~
              }
           }
         }
      }
    \l__dataplot_content_tl
    \tl_clear:N \l__dataplot_content_tl
  }
}
```

Plot y label if required.

```
\cs_new:Nn \__dataplot_add_y_label:
{
  \tl_if_empty:NF \l__dataplot_ylabel_tl
  {
    \fp_set:Nn \l__dataplot_tic_label_offset_fp
    {
      (
       (
        \dim_to_decimal_in_sp:n
         {
           \baselineskip
           + \l__dataplot_y_tic_label_width_dim
           + \DTLticklabeloffset
```

```
          }
          + \l__dataplot_offset_x_fp
        )
         / \l__dataplot_scale_x_fp
      ) / 65536
    }
```
Get halfway position
```
    \bool_if:NTF \l__dataplot_zero_x_axis_bool
     {
       \fp_set:Nn \l__dataplot_x_fp
        { - \l__dataplot_tic_label_offset_fp }
     }
     {
       \fp_set:Nn \l__dataplot_x_fp
        { \l__dataplot_min_x_fp - \l__dataplot_tic_label_offset_fp }
     }
    \fp_set:Nn \l__dataplot_y_fp
     { \l__dataplot_min_y_fp + 0.5 * \l__dataplot_y_extent_fp }
    \tl_put_right:Nx \l__dataplot_content_tl
     {
       \exp_not:N \draw
        ( \fp_use:N \l__dataplot_x_fp,
            \fp_use:N \l__dataplot_y_fp ) ~
        node[rotate=90,anchor=south]
          { \exp_not:V \l__dataplot_ylabel_tl }; ~
     }
    \l__dataplot_content_tl
    \tl_clear:N \l__dataplot_content_tl
  }
}
```
Plot the y minor ticks if required
```
\cs_new:Nn \__dataplot_add_minor_y:
{
 \legacy_if:nT { DTLyminortics }
  {
```
Get tick length in terms of canvas co-ordinates
```
    \fp_set:Nn \l__dataplot_tick_length_fp
     {
       (
         (
           \dim_to_decimal_in_sp:n { \DTLminorticklength }
            + \l__dataplot_offset_x_fp
         ) / \l__dataplot_scale_x_fp
       ) / 65536
     }
```
Iterate through minor ticks.
```
    \seq_map_inline:Nn \l__dataplot_y_minor_tic_seq
```

```
    {
      \tl_set:Nn \l__dataplot_y_fp { ##1 }
      \tl_set:Nx \l__dataplot_y_tl
        { \fp_to_decimal:N \l__dataplot_y_fp }
     \legacy_if:nTF { DTLyticsin }
      {
```

Tick right of the y-axis. (Tick `in`)

```
          \bool_if:NTF \l__dataplot_zero_x_axis_bool
           {
```

Enforce y axis to pass through $x = 0$.

```
            \fp_set_eq:NN \l__dataplot_x_fp
               \l__dataplot_tick_length_fp
            \tl_put_right:Nx \l__dataplot_content_tl
             {
               \exp_not:N \draw
               ( 0 , \l__dataplot_y_tl )
                --
               ( \fp_to_decimal:N \l__dataplot_x_fp,
                  \l__dataplot_y_tl ); ~
             }
           }
           {
```

Don't enforce y axis to pass through $x = 0$ (but this will occur if min x = 0).

```
            \fp_set:Nn \l__dataplot_x_fp
              { \l__dataplot_min_x_fp + \l__dataplot_tick_length_fp }
            \tl_put_right:Nx \l__dataplot_content_tl
             {
               \exp_not:N \draw
               ( \DTLminX, \l__dataplot_y_tl )
                --
               ( \fp_to_decimal:N \l__dataplot_x_fp,
                  \l__dataplot_y_tl ); ~
             }
           }
      }
      {
```

Tick left of the y-axis. (Tick `out`)

```
          \bool_if:NTF \l__dataplot_zero_x_axis_bool
           {
             \fp_set:Nn \l__dataplot_x_fp
               { - \l__dataplot_tick_length_fp }
             \tl_put_right:Nx \l__dataplot_content_tl
              {
                \exp_not:N \draw
                ( 0 , \l__dataplot_y_tl )
                 --
                ( \fp_to_decimal:N \l__dataplot_x_fp ,
```

```
                \l__dataplot_y_tl ); ~
          }
        }
        {
```

Don't enforce y axis to pass through $x = 0$ (but this will occur if min x = 0).

```
          \fp_set:Nn \l__dataplot_x_fp
            { \l__dataplot_min_x_fp - \l__dataplot_tick_length_fp }
          \tl_put_right:Nx \l__dataplot_content_tl
           {
             \exp_not:N \draw
             ( \DTLminX, \l__dataplot_y_tl )
              --
             ( \fp_to_decimal:N \l__dataplot_x_fp ,
                \l__dataplot_y_tl ); ~
          }
        }
    }
```

Draw box ticks, if box setting is on.

```
    \legacy_if:nT { DTLbox }
     {
```

If min x isn't 0 and the box setting is on, need to draw tick on the left of the box according to the box setting.

```
        \__dataplot_draw_box_ticks_lower_y:n
         {
           \int_case:nnT { \l__dataplot_box_ticks_int }
            {
              { \c_one_int } % match
               {
                 \legacy_if:nTF { DTLyticsin }
                 {
                   \fp_set:Nn \l__dataplot_x_fp
                    {
                      \l__dataplot_extended_min_x_fp
                      + \l__dataplot_tick_length_fp
                    }
                 }
                 {
                   \fp_set:Nn \l__dataplot_x_fp
                    {
                      \l__dataplot_extended_min_x_fp
                      - \l__dataplot_tick_length_fp
                    }
                 }
               }
              { 2 } % in
               {
                 \fp_set:Nn \l__dataplot_x_fp
                  {
```

```
                \l__dataplot_extended_min_x_fp
                 + \l__dataplot_tick_length_fp
              }
            }
           { 3 } % out
            {
              \fp_set:Nn \l__dataplot_x_fp
               {
                 \l__dataplot_extended_min_x_fp
                  - \l__dataplot_tick_length_fp
               }
            }
         }
         {
           \tl_put_right:Nx \l__dataplot_content_tl
            {
              \exp_not:N \draw
              (
                \fp_to_decimal:N \l__dataplot_extended_min_x_fp ,
                \l__dataplot_y_tl
               )
               --
              ( \fp_to_decimal:N \l__dataplot_x_fp,
                  \l__dataplot_y_tl ); ~
            }
         }
     }
```

Draw opposite tick.

```
    \int_case:nnT { \l__dataplot_box_ticks_int }
     {
       { \c_one_int } % match
        {
          \legacy_if:nTF { DTLyticsin }
           {
            \fp_set:Nn \l__dataplot_x_fp
             {
               \l__dataplot_extended_max_x_fp
                - \l__dataplot_tick_length_fp
             }
           }
           {
            \fp_set:Nn \l__dataplot_x_fp
             {
               \l__dataplot_extended_max_x_fp
               + \l__dataplot_tick_length_fp
             }
           }
         }
       { 2 } % in
```

```
            {
              \fp_set:Nn \l__dataplot_x_fp
               {
                 \l__dataplot_extended_max_x_fp
                   - \l__dataplot_tick_length_fp
               }
            }
           { 3 } % out
            {
              \fp_set:Nn \l__dataplot_x_fp
               {
                 \l__dataplot_extended_max_x_fp
                 + \l__dataplot_tick_length_fp
               }
            }
        }
        {
          \tl_put_right:Nx \l__dataplot_content_tl
           {
             \exp_not:N \draw
             (
               \fp_to_decimal:N \l__dataplot_extended_max_x_fp ,
               \l__dataplot_y_tl
             )
              --
             ( \fp_to_decimal:N \l__dataplot_x_fp,
                 \l__dataplot_y_tl); ~
           }
        }
      }
   }
   \l__dataplot_content_tl
   \tl_clear:N \l__dataplot_content_tl
  }
}
```

Used to count the total number of plot streams. Primarily provided for use as a return value for the `plot` action, but may also be used in the legend hook to access the stream index.

```
\int_new:N \l_dataplot_stream_index_int
```

Plot marks and draw lines for each database.

```
\cs_new:Nn \__dataplot_plot_data:
{
```

Initialise:

```
  \tl_clear:N \l__dataplot_current_mark_tl
  \tl_clear:N \l__dataplot_current_mark_color_tl
  \tl_clear:N \l__dataplot_current_line_tl
  \tl_clear:N \l__dataplot_current_line_color_tl
  \int_zero:N \l_dataplot_stream_index_int
```

Iterate through each database

```
\seq_map_indexed_inline:Nn \l__dataplot_dbnames_seq
{
```

Reset styles at the start of each database unless corresponding group by setting on:

```
  \bool_if:NTF \l__dataplot_mark_group_bool
   {
      \__dataplot_get_style:NNN
        \l__dataplot_mark_seq \DTLplotmarks
        \l__dataplot_current_mark_tl
   }
   {
     \bool_if:NT \l__dataplot_no_group_mark_reset_bool
      {
        \__dataplot_set_style_from_clist:NN
          \l__dataplot_mark_seq \DTLplotmarks
      }
     \tl_clear:N \l__dataplot_current_mark_tl
   }
  \bool_if:NTF \l__dataplot_mark_color_group_bool
   {
     \__dataplot_get_style:NNN
       \l__dataplot_mark_colors_seq \DTLplotmarkcolors
       \l__dataplot_current_mark_color_tl
   }
   {
     \bool_if:NT \l__dataplot_no_group_mark_color_reset_bool
      {
        \__dataplot_set_style_from_clist:NN
          \l__dataplot_mark_colors_seq \DTLplotmarkcolors
      }
     \tl_clear:N \l__dataplot_current_mark_color_tl
   }
  \bool_if:NTF \l__dataplot_line_group_bool
   {
     \__dataplot_get_style:NNN
       \l__dataplot_line_seq \DTLplotlines
       \l__dataplot_current_line_tl
   }
   {
     \bool_if:NT \l__dataplot_no_group_line_reset_bool
      {
        \__dataplot_set_style_from_clist:NN
          \l__dataplot_line_seq \DTLplotlines
      }
     \tl_clear:N \l__dataplot_current_line_tl
   }
  \bool_if:NTF \l__dataplot_line_color_group_bool
   {
     \__dataplot_get_style:NNN
```

```
      \l__dataplot_line_colors_seq \DTLplotlinecolors
      \l__dataplot_current_line_color_tl
    }
    {
      \bool_if:NT \l__dataplot_no_group_line_color_reset_bool
       {
         \__dataplot_set_style_from_clist:NN
           \l__dataplot_line_colors_seq \DTLplotlinecolors
       }
      \tl_clear:N \l__dataplot_current_line_color_tl
    }
  \DTLsetup{ default-name = { ##2 } }
  \__dataplot_filtered_map:nnn
  {
```

Pre-map code.

```
    \int_incr:N \l_dataplot_stream_index_int
```

Current mark setting.

```
    \legacy_if:nT { DTLshowmarkers }
    {
```

Get the current plot mark if applicable.

```
      \bool_if:NF \l__dataplot_mark_group_bool
       {
         \__dataplot_get_style:NNN
           \l__dataplot_mark_seq \DTLplotmarks
           \l__dataplot_current_mark_tl
       }
```

Get the current plot mark colour.

```
      \bool_if:NF \l__dataplot_mark_color_group_bool
       {
         \__dataplot_get_style:NNN
           \l__dataplot_mark_colors_seq \DTLplotmarkcolors
           \l__dataplot_current_mark_color_tl
       }
```

Save the current marker and mark colour.

```
      \tl_if_empty:NTF \l__dataplot_current_mark_tl
       {
         \tl_clear:N \l__dataplot_current_mark_style_tl
       }
       {
        \tl_set_eq:NN
          \l__dataplot_current_mark_style_tl
          \l__dataplot_current_mark_tl
        \tl_if_empty:NF \l__dataplot_current_mark_color_tl
         {
           \tl_put_left:Nx \l__dataplot_current_mark_style_tl
            {
               \exp_not:N \pgfsetstrokecolor
```

```
                    { \l__dataplot_current_mark_color_tl }
                }
              }
            }
        }
```

Current line setting.

```
        \legacy_if:nT { DTLshowlines }
        {
```

Get the current plot line.

```
          \bool_if:NF \l__dataplot_line_group_bool
           {
             \__dataplot_get_style:NNN
               \l__dataplot_line_seq \DTLplotlines
               \l__dataplot_current_line_tl
           }
```

Get the current plot line colour.

```
           \bool_if:NF \l__dataplot_line_color_group_bool
           {
             \__dataplot_get_style:NNN
               \l__dataplot_line_colors_seq \DTLplotlinecolors
               \l__dataplot_current_line_color_tl
           }
```

Save the current marker and mark colour.

```
           \tl_if_empty:NTF \l__dataplot_current_line_tl
            {
              \tl_clear:N \l__dataplot_current_line_style_tl
            }
            {
             \tl_set_eq:NN
               \l__dataplot_current_line_style_tl
               \l__dataplot_current_line_tl
             \tl_if_empty:NF \l__dataplot_current_line_color_tl
              {
                \tl_put_left:Nx \l__dataplot_current_line_style_tl
                 {
                    \exp_not:N \pgfsetstrokecolor
                      { \l__dataplot_current_line_color_tl }
                 }
              }
            }
        }
```

Append this plot setting to the legend if applicable.

```
        \int_if_zero:nF \l__dataplot_legend_setting_int
        {
```

Get legend label.

```
           \seq_pop_left:NNF
```

```
      \l__dataplot_legend_labels_seq
      \l__datatool_tmpa_tl
    {
      \tl_clear:N \l__datatool_tmpa_tl
      \exp_args:NNx
       \dataplot_get_default_legend:Nnnnn
         \l__datatool_tmpa_tl
        { ##1 } { ##2 }
        { \l__dataplot_x_key_tl }
        { \l__dataplot_y_key_tl }
    }
```

Only add to the legend if the label isn't empty.

```
    \tl_if_empty:NF \l__datatool_tmpa_tl
     {
      \exp_args:Noox
       \DTLaddtoplotlegend
       { \l__dataplot_current_mark_style_tl }
       { \l__dataplot_current_line_style_tl }
       { \l__datatool_tmpa_tl }
     }
  }
```

Store stream in `\l__dataplot_stream_tl`:

```
  \tl_set:Nn \l__dataplot_stream_tl
    { \pgfplotstreamstart }
```

Only plot points that lie inside bounds.

```
  }
  {
    \datatool_set_fp:Nn \l__dataplot_x_fp
     {
       \l__dataplot_x_tl
     }
    \datatool_set_fp:Nn \l__dataplot_y_fp
     {
       \l__dataplot_y_tl
     }
    \tl_set:Nx \l__dataplot_decimal_x_tl
      { \fp_to_tl:N \l__dataplot_x_fp }
    \tl_set:Nx \l__dataplot_decimal_y_tl
      { \fp_to_tl:N \l__dataplot_y_fp }
    \__dataplot_x_y_bounded:nnnnnnT
      { \l__dataplot_x_fp }
        { \l__dataplot_min_x_fp } { \l__dataplot_max_x_fp }
      { \l__dataplot_y_fp }
        { \l__dataplot_min_y_fp } { \l__dataplot_max_y_fp }
    {
```

Apply transformation to co-ordinates

```
      \fp_set:Nn \l__dataplot_x_fp
```

```
        {
          \l__dataplot_x_fp * \l__dataplot_scale_x_fp
          + \l__dataplot_offset_x_fp
        }
      \tl_set:Nx \l__dataplot_x_tl
        { \fp_to_tl:N \l__dataplot_x_fp }
      \fp_set:Nn \l__dataplot_y_fp
        {
          \l__dataplot_y_fp * \l__dataplot_scale_y_fp
          + \l__dataplot_offset_y_fp
        }
      \tl_set:Nx \l__dataplot_y_tl
        { \fp_to_tl:N \l__dataplot_y_fp }
      \tl_put_right:Nx \l__dataplot_stream_tl
        {
          \exp_not:N \pgfplotstreampoint
           {
             \exp_not:N \pgfpointxy
              { \l__dataplot_x_tl }
              { \l__dataplot_y_tl }
           }
        }
    }
  }
  {
```

Add end of plot stream.

```
    \tl_put_right:Nn \l__dataplot_stream_tl
      { \pgfplotstreamend }
```

Draw path if applicable.

```
    \tl_if_empty:NF \l__dataplot_current_line_style_tl
     {
       \begin{scope}
         \l__dataplot_current_line_style_tl
         \pgfplothandlerlineto
         \l__dataplot_stream_tl
         \pgfusepath{stroke}
       \end{scope}
     }
    \tl_if_empty:NF \l__dataplot_current_mark_style_tl
     {
       \begin{scope}
         \exp_args:NV \pgfplothandlermark
           \l__dataplot_current_mark_style_tl
         \l__dataplot_stream_tl
         \pgfusepath{stroke}
       \end{scope}
     }
  }
}
```

Plot legend if required.

```
  \int_if_zero:nF \l__dataplot_legend_setting_int
  {
```

Prior to v3.0, `\DTLaddtoplotlegend` appended to `\dtl@legend` so allow for backward-compatibility (this may be removed in future):

```
    \bool_lazy_and:nnT
     { \tl_if_empty_p:N \l_dataplot_legend_tl }
     { \tl_if_exist_p:N \dtl@legend }
    {
      \tl_set_eq:NN \l_dataplot_legend_tl \dtl@legend
    }
  }
  \tl_if_empty:NF \l_dataplot_legend_tl
  {
    \__dataplot_do_legend:
  }
}
```

Do the legend:

```
\cs_new:Nn \__dataplot_do_legend:
{
  \tl_clear:N \l__dataplot_content_tl
  \dataplot_legend_add_begin:
  \dataplot_legend_add_end:
  \int_case:nnTF { \l__dataplot_legend_setting_int }
   {
     { 1 } % north
       {
         \fp_set:Nn \l__dataplot_x_fp
          {
            \l__dataplot_scale_x_fp *
              ( \l__dataplot_min_x_fp + 0.5 * \l__dataplot_x_extent_fp )
            + \l__dataplot_offset_x_fp
          }
         \fp_set:Nn \l__dataplot_y_fp
          {
            \l__dataplot_max_y_fp * \l__dataplot_scale_y_fp
            + \l__dataplot_offset_y_fp
            - \DTLlegendyoffset
          }
```

If the y offset is negative anchor at the south otherwise anchor north.

```
         \fp_compare:nNnTF
           { \DTLlegendyoffset } < { \c_zero_fp }
          {
            \tl_put_right:Nn \l__dataplot_content_tl
             {
               node [ anchor = south ]
             }
          }
```

```
        {
         \tl_put_right:Nn \l__dataplot_content_tl
          {
            node [ anchor = north ]
          }
        }
    }
  { 2 } % northeast
    {
      \fp_set:Nn \l__dataplot_x_fp
       {
         \l__dataplot_max_x_fp * \l__dataplot_scale_x_fp
           + \l__dataplot_offset_x_fp
           - \DTLlegendxoffset
       }
      \fp_set:Nn \l__dataplot_y_fp
       {
         \l__dataplot_max_y_fp * \l__dataplot_scale_y_fp
           + \l__dataplot_offset_y_fp
           - \DTLlegendyoffset
       }
      \tl_put_right:Nn \l__dataplot_content_tl
       {
         node [ anchor =
       }
```

If the y offset is negative anchor at the south otherwise anchor north.

```
      \fp_compare:nNnTF
        { \DTLlegendyoffset } < { \c_zero_fp }
       {
         \tl_put_right:Nn \l__dataplot_content_tl { south ~ }
       }
       {
         \tl_put_right:Nn \l__dataplot_content_tl { north ~ }
       }
```

If the x offset is negative anchor at the west otherwise anchor east.

```
      \fp_compare:nNnTF
        { \DTLlegendxoffset } < { \c_zero_fp }
       {
         \tl_put_right:Nn \l__dataplot_content_tl { west }
       }
       {
         \tl_put_right:Nn \l__dataplot_content_tl { east }
       }
      \tl_put_right:Nn \l__dataplot_content_tl
       {
          ]
       }
    }
  { 3 } % east
```

```
    {
      \fp_set:Nn \l__dataplot_x_fp
       {
         \l__dataplot_max_x_fp * \l__dataplot_scale_x_fp
          + \l__dataplot_offset_x_fp
          - \DTLlegendxoffset
       }
      \fp_set:Nn \l__dataplot_y_fp
       {
         \l__dataplot_offset_y_fp
         + \l__dataplot_scale_y_fp
           * ( \l__dataplot_min_y_fp + 0.5 * \l__dataplot_y_extent_fp )
       }
```

If the x offset is negative anchor at the west otherwise anchor east.

```
      \fp_compare:nNnTF
        { \DTLlegendxoffset } < { \c_zero_fp }
       {
        \tl_put_right:Nn \l__dataplot_content_tl
         {
           node [ anchor = west ]
         }
       }
       {
        \tl_put_right:Nn \l__dataplot_content_tl
         {
           node [ anchor = east ]
         }
       }
    }
  { 4 } % southeast
    {
      \fp_set:Nn \l__dataplot_x_fp
       {
         \l__dataplot_max_x_fp * \l__dataplot_scale_x_fp
         + \l__dataplot_offset_x_fp
         - \DTLlegendxoffset
       }
      \fp_set:Nn \l__dataplot_y_fp
       {
         \l__dataplot_min_y_fp * \l__dataplot_scale_y_fp
         + \l__dataplot_offset_y_fp
         + \DTLlegendyoffset
       }
      \tl_put_right:Nn \l__dataplot_content_tl
       {
         node [ anchor =
       }
```

If the y offset is negative anchor at the north otherwise anchor south.

```
      \fp_compare:nNnTF
```

```
          { \DTLlegendyoffset } < { \c_zero_fp }
         {
           \tl_put_right:Nn \l__dataplot_content_tl { north ~ }
         }
         {
           \tl_put_right:Nn \l__dataplot_content_tl { south ~ }
         }
```

If the x offset is negative anchor at the west otherwise anchor east.

```
        \fp_compare:nNnTF
          { \DTLlegendxoffset } < { \c_zero_fp }
         {
           \tl_put_right:Nn \l__dataplot_content_tl { west }
         }
         {
           \tl_put_right:Nn \l__dataplot_content_tl { east }
         }
        \tl_put_right:Nn \l__dataplot_content_tl
         {
           ]
         }
    }
  { 5 } % south
    {
      \fp_set:Nn \l__dataplot_x_fp
       {
         \l__dataplot_scale_x_fp
         * ( \l__dataplot_min_x_fp + 0.5 * \l__dataplot_x_extent_fp )
         + \l__dataplot_offset_x_fp
       }
      \fp_set:Nn \l__dataplot_y_fp
       {
         \l__dataplot_min_y_fp * \l__dataplot_scale_y_fp
          + \l__dataplot_offset_y_fp
          + \DTLlegendyoffset
       }
```

If the y offset is negative anchor at the north otherwise anchor south.

```
        \fp_compare:nNnTF
          { \DTLlegendyoffset } < { \c_zero_fp }
         {
          \tl_put_right:Nn \l__dataplot_content_tl
           {
             node [ anchor = north ]
           }
         }
         {
          \tl_put_right:Nn \l__dataplot_content_tl
           {
             node [ anchor = south ]
           }
```

```
      }
    }
  { 6 } % southwest
    {
      \fp_set:Nn \l__dataplot_x_fp
       {
         \l__dataplot_min_x_fp * \l__dataplot_scale_x_fp
         + \l__dataplot_offset_x_fp
         + \DTLlegendxoffset
       }
      \fp_set:Nn \l__dataplot_y_fp
       {
         \l__dataplot_min_y_fp * \l__dataplot_scale_y_fp
         + \l__dataplot_offset_y_fp
         + \DTLlegendyoffset
       }
      \tl_put_right:Nn \l__dataplot_content_tl
       {
         node [ anchor =
       }
```

If the y offset is negative anchor at the north otherwise anchor south.

```
      \fp_compare:nNnTF
        { \DTLlegendyoffset } < { \c_zero_fp }
       {
         \tl_put_right:Nn \l__dataplot_content_tl { north ~ }
       }
       {
         \tl_put_right:Nn \l__dataplot_content_tl { south ~ }
       }
```

If the x offset is negative anchor at the east otherwise anchor west.

```
      \fp_compare:nNnTF
        { \DTLlegendxoffset } < { \c_zero_fp }
       {
         \tl_put_right:Nn \l__dataplot_content_tl { east }
       }
       {
         \tl_put_right:Nn \l__dataplot_content_tl { west }
       }
      \tl_put_right:Nn \l__dataplot_content_tl
       {
          ]
       }
    }
  { 7 } % west
    {
      \fp_set:Nn \l__dataplot_x_fp
       {
         \l__dataplot_offset_x_fp
          + \l__dataplot_min_x_fp * \l__dataplot_scale_x_fp
```

```
          + \DTLlegendxoffset
        }
      \fp_set:Nn \l__dataplot_y_fp
        {
          \l__dataplot_scale_y_fp
           * ( \l__dataplot_min_y_fp + 0.5 * \l__dataplot_y_extent_fp )
          + \l__dataplot_offset_y_fp
        }
```

If the x offset is negative anchor at the east otherwise anchor west.

```
      \fp_compare:nNnTF
        { \DTLlegendxoffset } < { \c_zero_fp }
        {
         \tl_put_right:Nn \l__dataplot_content_tl
          {
            node [ anchor = east ]
          }
        }
        {
         \tl_put_right:Nn \l__dataplot_content_tl
          {
            node [ anchor = west ]
          }
        }
    }
  { 8 } % northwest
    {
      \fp_set:Nn \l__dataplot_x_fp
        {
          \l__dataplot_offset_x_fp
           + \l__dataplot_min_x_fp * \l__dataplot_scale_x_fp
           + \DTLlegendxoffset
        }
      \fp_set:Nn \l__dataplot_y_fp
        {
          \l__dataplot_offset_y_fp
           + \l__dataplot_max_y_fp * \l__dataplot_scale_y_fp
           - \DTLlegendyoffset
        }
      \tl_put_right:Nn \l__dataplot_content_tl
        {
          node [ anchor =
        }
```

If the y offset is negative anchor at the south otherwise anchor north.

```
      \fp_compare:nNnTF
        { \DTLlegendyoffset } < { \c_zero_fp }
        {
          \tl_put_right:Nn \l__dataplot_content_tl { south ~ }
        }
        {
```

```
            \tl_put_right:Nn \l__dataplot_content_tl { north ~ }
          }
```

If the x offset is negative anchor at the east otherwise anchor west.

```
          \fp_compare:nNnTF
            { \DTLlegendxoffset } < { \c_zero_fp }
          {
            \tl_put_right:Nn \l__dataplot_content_tl { east }
          }
          {
            \tl_put_right:Nn \l__dataplot_content_tl { west }
          }
          \tl_put_right:Nn \l__dataplot_content_tl
          {
             ]
          }
        }
    }
    {
      \tl_put_left:Nx \l__dataplot_content_tl
        {
          \exp_not:N \draw
           ( \fp_to_decimal:N \l__dataplot_x_fp,
            \fp_to_decimal:N \l__dataplot_y_fp ) ~
        }
      \tl_put_right:Nx \l__dataplot_content_tl
        {
          ~ {
            \exp_not:N \DTLformatlegend
             {
               \exp_not:V \l_dataplot_legend_tl
             }
          }; ~
        }
    }
    {
      \tl_put_right:Nx \l__dataplot_content_tl
        {
          \exp_not:N \DTLcustomlegend
          {
            \exp_not:V \l_dataplot_legend_tl
          }
        }
    }
   \l__dataplot_content_tl
   \tl_clear:N \l__dataplot_content_tl
 }
```

Syntax: ⟨*seq var*⟩ ⟨*clist var*⟩ ⟨*tl var*⟩

Get style setting from sequence, and reset ⟨*seq var*⟩ from ⟨*clist var*⟩ if empty:

```
\cs_new:Nn \__dataplot_get_style:NNN
{
  \seq_if_empty:NT #1
  {
    \__dataplot_set_style_from_clist:NN #1 #2
  }
  \seq_pop_left:NN #1 #3
```

Check for `\relax` to indicate no mark for the current set.

```
  \tl_if_eq:NnT #3 { \relax }
  {
    \tl_clear:N #3
  }
}
```

`\DTLplot`

`\DTLplot[`⟨*condition*⟩`]{`⟨*db list*⟩`}{`⟨*settings*⟩`}`

Creates a plot (inside a tikzpicture environment) of all the data given in the databases listed in ⟨*db list*⟩.

```
\NewDocumentCommand \DTLplot { o m m }
{
 \group_begin:
```

Initialise command to encapsulate plot code. Any errors should change this to do nothing.

```
  \cs_set_eq:NN \__dataplot_do_plot:n \use:n
```

Parse options:

```
  \keys_set:nn { datatool / plot } { #3 }
```

Check optional argument:

```
  \IfValueT { #1 }
   {
```

The optional argument overrides the condition setting:

```
      \cs_set:Nn \__dataplot_filter:T
      {
        \ifthenelse { #1 } { ##1 } { }
      }
   }
```

Store the list of database names in a sequence variable:

```
  \exp_args:NNx \seq_set_from_clist:Nn
    \l__dataplot_dbnames_seq { #2 }
```

Check that all the databases exist.

```
  \seq_map_inline:Nn \l__dataplot_dbnames_seq
   {
     \DTLifdbexists { ##1 } { }
      {
       \PackageError { dataplot }
```

```
        {
          Database ~ `##1' ~ doesn't ~ exist
        }
        {
          Check ~ you ~ have ~ spelt ~ the ~ database ~ name ~
          correctly ~ in ~ the ~ argument ~ of ~
          \token_to_str:N \DTLplot { \exp_not:n { #2 } } { ... }
        }
       \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
      }
   }
  \__dataplot_do_plot:n
   {
```

Convert the minor gap dimension to user co-ordinates:

```
    \fp_set:Nn \l__dataplot_min_minor_gap_fp
      { \DTLminminortickgap / 65536 }
```

Save the plot marks comma-separated list as a sequence:

```
    \__dataplot_set_style_from_clist:NN
      \l__dataplot_mark_seq \DTLplotmarks
```

Save the plot lines comma-separated list as a sequence:

```
    \__dataplot_set_style_from_clist:NN
      \l__dataplot_line_seq \DTLplotlines
```

Save the plot mark colours comma-separated list as a sequence:

```
    \__dataplot_set_style_from_clist:NN
      \l__dataplot_mark_colors_seq \DTLplotmarkcolors
```

Save the plot line colours comma-separated list as a sequence:

```
    \__dataplot_set_style_from_clist:NN
      \l__dataplot_line_colors_seq \DTLplotlinecolors
```

Clear the legend token list:

```
    \tl_clear:N \l_dataplot_legend_tl
```

Check the user has supplied at least one x variable:

```
    \clist_if_empty:NT \l__dataplot_x_keys_clist
    {
      \PackageError { dataplot }
      {Missing ~ x ~ setting ~ for ~ \token_to_str:N \DTLplot}
      {}
      \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
    }
```

Check the user has supplied at least one y variable:

```
    \seq_if_empty:NTF \l__dataplot_y_keys_seq
    {
      \PackageError { dataplot }
      {Missing ~ y ~ setting ~ for ~ \token_to_str:N \DTLplot}
      {}
      \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
```

```
      }
      {
```

Check that there aren't more x than y variables:

```
        \int_compare:nNnT
          { \clist_count:N \l__dataplot_x_keys_clist }
           >
          { \seq_count:N \l__dataplot_y_keys_seq }
        {
          \PackageError { dataplot }
          { Too ~ many ~ x ~ keys ~ listed ~ in ~ \token_to_str:N \DTLplot }
          {
             The ~ number ~ of ~ x ~ keys ~ must ~ be ~ less ~ than ~
             or ~ equal ~ to ~ the ~ number ~ of ~ y ~ keys
          }
          \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
        }
      }
     \__dataplot_init_and_do_plot:
   }
  \group_end:
 }
```

Initialisation code:

```
 \cs_new:Nn \__dataplot_init_and_do_plot:
 {
```

If user didn't specified bounds, compute the maximum and minimum x and y values over all the databases listed.

```
   \__dataplot_do_plot:n
    {
      \__dataplot_update_bounds:
    }
```

Do the plot code:

```
   \__dataplot_do_plot:n
    {
```

Calculate the extended bounds for the axes and box:

```
      \fp_set:Nn \l__dataplot_extended_min_x_fp
       {
         \l__dataplot_min_x_fp
         - \l__dataplot_extend_min_x_axis_fp
       }
      \fp_set:Nn \l__dataplot_extended_min_y_fp
       {
         \l__dataplot_min_y_fp
         - \l__dataplot_extend_min_y_axis_fp
       }
      \fp_set:Nn \l__dataplot_extended_max_x_fp
       {
```

```
      \l__dataplot_max_x_fp
      + \l__dataplot_extend_max_x_axis_fp
    }
  \fp_set:Nn \l__dataplot_extended_max_y_fp
    {
      \l__dataplot_max_y_fp
      + \l__dataplot_extend_max_y_axis_fp
    }
```

Determine whether to set the zero axis booleans:

```
  \bool_set_false:N \l__dataplot_zero_x_axis_bool
  \bool_set_false:N \l__dataplot_zero_y_axis_bool
  \bool_if:NF \l__dataplot_side_axes_bool
  {
    \fp_compare:nT
      {
        \l__dataplot_extended_min_x_fp
        < \c_zero_fp
        < \l__dataplot_extended_max_x_fp
      }
     {
```

X axis includes 0. That is, the y axis should intersect at x=0:

```
        \bool_set_true:N \l__dataplot_zero_x_axis_bool
      }
     \fp_compare:nT
      {
        \l__dataplot_extended_min_y_fp
        < \c_zero_fp
        < \l__dataplot_extended_max_y_fp
      }
     {
```

Y axis includes 0. That is, the x axis should intersect at y=0:

```
        \bool_set_true:N \l__dataplot_zero_y_axis_bool
      }
  }
```

Check the omit zero label setting:

```
  \int_case:nnF { \l__dataplot_omit_zero_label_action_int }
    {
      { 2 } % omit both
       {
         \bool_set_true:N \l__dataplot_omit_zero_x_label_bool
         \bool_set_true:N \l__dataplot_omit_zero_y_label_bool
       }
      { 3 } % omit x
       {
         \bool_set_true:N \l__dataplot_omit_zero_x_label_bool
         \bool_set_false:N \l__dataplot_omit_zero_y_label_bool
       }
```

```
      { 4 } % omit y
       {
         \bool_set_false:N \l__dataplot_omit_zero_x_label_bool
         \bool_set_true:N \l__dataplot_omit_zero_y_label_bool
       }
      { 5 } % don't omit
       {
         \bool_set_false:N \l__dataplot_omit_zero_x_label_bool
         \bool_set_false:N \l__dataplot_omit_zero_y_label_bool
       }
    }
    {% auto
      \bool_set_eq:NN
        \l__dataplot_omit_zero_x_label_bool
        \l__dataplot_zero_x_axis_bool
      \bool_set_eq:NN
        \l__dataplot_omit_zero_y_label_bool
        \l__dataplot_zero_y_axis_bool
    }
```

Determine whether or not to draw ticks along the minimum box edges.

```
  \bool_lazy_all:nTF
   {
    { \legacy_if_p:n { DTLbox } }
    { \bool_not_p:n { \l__dataplot_side_axes_bool } }
    { ! \fp_compare_p:nNn { \l__dataplot_extended_min_y_fp } = { \c_zero_fp } }
   }
   {
     \cs_set_eq:NN \__dataplot_draw_box_ticks_lower_x:n \use:n
   }
   {
     \cs_set_eq:NN \__dataplot_draw_box_ticks_lower_x:n \use_none:n
   }
  \bool_lazy_all:nTF
   {
    { \legacy_if_p:n { DTLbox } }
    { \bool_not_p:n { \l__dataplot_side_axes_bool } }
    { ! \fp_compare_p:nNn { \l__dataplot_extended_min_x_fp } = { \c_zero_fp } }
   }
   {
     \cs_set_eq:NN \__dataplot_draw_box_ticks_lower_y:n \use:n
   }
   {
     \cs_set_eq:NN \__dataplot_draw_box_ticks_lower_y:n \use_none:n
   }
```

Determine scaling factors and offsets.

```
  \dataplot_calc_transform:
```

If x tics specified, construct a list of x tic points if not already specified.

```
  \__dataplot_calc_x_tics:
```

If y tics specified, construct a list of y tic points if not already specified.

```
    \__dataplot_calc_y_tics:
```

Do the picture.

```
    \__dataplot_do_plot:
  }
}
```

Action ‘plot’:

```
\cs_new:cn { __datatool_action_ plot : }
{
 \group_begin:
```

This will be the primary return value.

```
  \int_zero:N \l_dataplot_stream_index_int
  \tl_clear:N \l__dataplot_content_tl
```

Initialise command to encapsulate plot code. Any errors should change this to do nothing.

```
  \cs_set_eq:NN \__dataplot_do_plot:n \use:n
```

Parse options if provided.

```
  \clist_if_empty:NF \l__datatool_action_options_clist
   {
     \keys_set:nV { datatool / plot }
       \l__datatool_action_options_clist
   }
```

The ‘name’ action setting may be a comma-separated list of database names. These will already be available in a sequence, so just copy it.

```
  \seq_set_eq:NN
     \l__dataplot_dbnames_seq \l__datatool_action_names_seq
```

Check that all the databases exist.

```
  \seq_map_inline:Nn \l__dataplot_dbnames_seq
   {
     \DTLifdbexists { ##1 } { }
      {
       \__datatool_action_error:nn
        {
          Database ~ `##1' ~ doesn't ~ exist
        }
        {
          Check ~ you ~ have ~ spelt ~ the ~ database ~ name ~
          correctly ~ in ~
          \token_to_str:N \DTLaction [name=
          { \seq_use::N \l__datatool_action_names_seq { , } } ]
          { \l__datatool_action_tl } ~ or ~ check ~ the ~ default-
name ~
          option ~ in ~ \token_to_str:N \DTLsetup
        }
       \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
```

```
    }
  }
\__dataplot_do_plot:n
  {
```

Convert the minor gap dimension to user co-ordinates:

```
\fp_set:Nn \l__dataplot_min_minor_gap_fp
  { \DTLminminortickgap / 65536 }
```

Save the plot marks comma-separated list as a sequence:

```
\__dataplot_set_style_from_clist:NN
  \l__dataplot_mark_seq \DTLplotmarks
```

Save the plot lines comma-separated list as a sequence:

```
\__dataplot_set_style_from_clist:NN
  \l__dataplot_line_seq \DTLplotlines
```

Save the plot mark colours comma-separated list as a sequence:

```
\__dataplot_set_style_from_clist:NN
  \l__dataplot_mark_colors_seq \DTLplotmarkcolors
```

Save the plot line colours comma-separated list as a sequence:

```
\__dataplot_set_style_from_clist:NN
  \l__dataplot_line_colors_seq \DTLplotlinecolors
```

Clear the legend token list:

```
\tl_clear:N \l_dataplot_legend_tl
```

Check the user has supplied at least one x variable:

```
\clist_if_empty:NT \l__dataplot_x_keys_clist
{
  \__datatool_action_error:nn
  {
    Missing ~ x ~ setting
  }
  {
    The ~ `x' ~ setting ~ needs ~ to ~ be ~ set ~ to ~
    the ~ column ~ key ~ to ~ be ~ used ~ for ~
    the ~ plot ~ x ~ values, ~ either ~ in ~
    \token_to_str:N
    \DTLaction [ options={x={...}}, ~ ... ]
      { \l__datatool_action_tl } ~ or ~ in ~
    \token_to_str:N \DTLsetup { plot = { x= { ... } } }
  }
  \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
}
```

Check the user has supplied at least one y variable:

```
\seq_if_empty:NTF \l__dataplot_y_keys_seq
{
  \__datatool_action_error:nn
  {
    Missing ~ y ~ setting
```

```
      }
      {
        The ~ `y' ~ setting ~ needs ~ to ~ be ~ set ~ to ~
        the ~ column ~ key ~ to ~ be ~ used ~ for ~
        the ~ plot ~ y ~ values, ~ either ~ in ~
        \token_to_str:N
        \DTLaction [ options={y={...}}, ~ ... ]
          { \l__datatool_action_tl } ~ or ~ in ~
        \token_to_str:N \DTLsetup { plot = { y= { ... } } }
      }
      \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
    }
    {
```

Check that there aren't more x than y variables:

```
      \int_compare:nNnT
        { \clist_count:N \l__dataplot_x_keys_clist }
         >
        { \seq_count:N \l__dataplot_y_keys_seq }
      {
        \__datatool_action_error:nn
        {
           Too ~ many ~ x ~ keys ~ listed ~ in ~
           x = { \l__dataplot_x_keys_clist }
        }
        {
           The ~ number ~ of ~ x ~ keys ~ must ~ be ~ less ~ than ~
           or ~ equal ~ to ~ the ~ number ~ of ~ y ~ keys
        }
        \cs_set_eq:NN \__dataplot_do_plot:n \use_none:n
      }
    }
  \__dataplot_do_plot:n
   {
```

Initialise and do the plot.

```
      \__dataplot_init_and_do_plot:
```

Secondary return properties. Calculating the bounds may have trigger an error, in which case don't set the secondary return values.

```
      \__dataplot_do_plot:n
       {
        \tl_set:Nx \l__dataplot_content_tl
         {
          \exp_not:N \__datatool_put_return_action_decimal:nn
           { min-x } { \DTLminX }
          \exp_not:N \__datatool_put_return_action_decimal:nn
           { min-y } { \DTLminY }
          \exp_not:N \__datatool_put_return_action_decimal:nn
           { max-x } { \DTLmaxX }
          \exp_not:N \__datatool_put_return_action_decimal:nn
```

```
                { max-y } { \DTLmaxY }
              \exp_not:N \__datatool_put_return_action_int:nn
                { stream-count }
                { \int_use:N \l_dataplot_stream_index_int }
              \exp_not:N \__datatool_put_return_action_int:nn
                { x-count } { \dataplot_x_key_count: }
              \exp_not:N \__datatool_put_return_action_int:nn
                { y-count } { \dataplot_y_key_count: }
              \exp_not:N \__datatool_put_return_action_int:nn
                { name-count } { \dataplot_db_count: }
            }
          }
        }
      }
```

Primary return property and end group.

```
  \tl_put_left:Nx \l__dataplot_content_tl
    {
      \exp_not:N \group_end:
      \exp_not:N \tl_set:Nn
        \exp_not:N \l__datatool_action_return_tl
         { \int_use:N \l_dataplot_stream_index_int }
    }
  \l__dataplot_content_tl
}
\cs_new:Nn \__dataplot_set_style_from_clist:NN
{
  \seq_set_from_clist:NN #1 #2
```

The sequence must contain at least one item:

```
  \seq_if_empty:NT #1
   {
     \seq_put_right:Nn #1 { }
   }
}
```

`\dtl@getbounds` Extract bounds function removed in version 3.0. See `\__dataplot_update_bounds:nn`

`\__dataplot_construct_tick_list:NNNN`⟨*min-fp*⟩⟨*max-fp*⟩⟨*min-gap-fp*⟩⟨*seq*⟩

```
\cs_new:Nn \__dataplot_construct_tick_list:NNNN
{
  \fp_compare:nTF { #1 < \c_zero_fp < #2 }
  {
```

Tick list straddles the origin.

```
    \fp_set:Nn \l__dataplot_width_fp { - ( #1 ) }
```

```
    \__dataplot_default_gap:NNN
      \l__dataplot_neg_gap_fp \l__dataplot_width_fp  #3
    \__dataplot_default_gap:NNN
      \l__dataplot_pos_gap_fp #2 #3
    \fp_set:Nn \l__dataplot_gap_fp
       { max ( \l__dataplot_neg_gap_fp, \l__dataplot_pos_gap_fp ) }
```

Don’t construct a list if minimum gap is greater than plot width

```
    \fp_compare:nF { \l__dataplot_gap_fp > \l__dataplot_width_fp }
    {
      \__dataplot_construct_tick_list_over_zero:NNNN
       #1 #2 #4 \l__dataplot_gap_fp
    }
  }
  {
```

Tick list doesn’t straddle the origin.

```
    \fp_set:Nn \l__dataplot_width_fp { #2 - #1 }
    \__dataplot_default_gap:NNN
      \l__dataplot_gap_fp \l__dataplot_width_fp #3
    \fp_compare:nTF { \l__dataplot_gap_fp < #3 }
    {
```

Don’t construct a list if minimum gap is greater than plot width

```
      \fp_compare:nTF { #3 > \l__dataplot_width_fp }
      {
         \seq_put_right:NV #4 #1
         \seq_put_right:NV #4 #2
      }
      {
        \__dataplot_construct_tick_list_with_gap:NNNN
         #1 #2 #4 #3
      }
    }
    {
      \__dataplot_construct_tick_list_with_gap:NNNN
       #1 #2 #4 \l__dataplot_gap_fp
    }
  }
}
```

`\__dataplot_default_gap:NNN` ⟨*fp-var*⟩ {⟨*extent*⟩} {⟨*max gap*⟩}

```
\cs_new:Nn \__dataplot_default_gap:NNN
{
  \fp_set:Nn #1 { #2 / 10 }
  \fp_compare:nNnT { #1 } < { #3 }
   {
     \fp_set:Nn #1 { #2 / 5 }
     \fp_compare:nNnT { #1 } < { #3 }
     {
       \fp_set_eq:NN #1 #3
```

```
      }
    }
   \fp_compare:nTF { #1 <= 0.1 }
    {
      \fp_set:Nn #1 { 0.1 }
    }
    {
      \fp_compare:nTF { #1 <= 0.5 }
       {
         \fp_set:Nn #1 { 0.5 }
       }
       {
         \fp_compare:nTF { #1 <= 0.5 }
          {
            \fp_set:Nn #1 { 0.5 }
          }
          {
            \fp_compare:nTF { #1 <= \c_one_fp }
             {
               \fp_set:NN #1 \c_one_fp
             }
             {
               \fp_compare:nTF { #1 <= 5 }
                {
                  \fp_set:Nn #1 { 5 }
                }
                {
                  \fp_compare:nTF { #1 <= 100 }
                   {
                     \fp_set:Nn #1 { 10 }
                   }
                   {
                     \fp_compare:nTF { #1 <= 1000 }
                      {
                        \fp_set:Nn #1 { 100 }
                      }
                      {
                        \fp_set:Nn #1 { 1000 * floor ( #1 / 1000 ) }
                      }
                   }
                }
             }
          }
       }
    }
}
```

\dtl@constructticklist

```
\dtl@constructticklist{⟨min⟩}{⟨max⟩}{⟨min gap⟩}{⟨list⟩}
```

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and store in ⟨*list*⟩ (a control sequence.)

Version 3.0: replaced with `\__dataplot_construct_tick_list:NNNN`⟨*min-fp*⟩⟨*max-fp*⟩⟨*seq*⟩⟨*min-gap-fp*⟩

```
\dataplot_add_end_tick:NNN ⟨fp var1⟩ ⟨fp var2⟩
⟨gap fp⟩ ⟨seq⟩
```

Determine whether or not to add end tick. The first argument is the possible overshoot, the second argument is the calculated or provided maximum and the third is the tick gap.

```
\cs_new:Nn \dataplot_add_end_tick:NNNN
{
  \fp_compare:nNnT
     { #1 - #2 } < { 0.1 * #3 }
   {
     \seq_put_right:NV #4 #2
   }
}
```

```
\__dataplot_construct_tick_list_with_gap:NNNN
⟨min-fp⟩⟨max-fp⟩⟨seq⟩⟨gap-fp⟩
```

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and store in ⟨*list*⟩ (a control sequence) using the gap given by ⟨*gap*⟩ where the gap is given in user co-ordinates.

```
\cs_new:Nn \__dataplot_construct_tick_list_with_gap:NNNN
{
  \fp_set:Nn \l__datatool_tmpa_fp
     { #4 * ( floor ( #1 / #4 ) + 1 ) }
  \fp_compare:nNnT
     { #1 - \l__datatool_tmpa_fp } < { 0.5 * #4 }
   {
     \seq_put_right:NV #3 #1
   }
  \fp_while_do:nn { #1 < \l__datatool_tmpa_fp < #2 }
   {
     \seq_put_right:NV #3 \l__datatool_tmpa_fp
     \fp_add:Nn \l__datatool_tmpa_fp { #4 }
   }
  \fp_compare:nNnT
     { \l__datatool_tmpa_fp - #2 } < { 0.5 * #4 }
   {
     \seq_put_right:NV #3 #2
   }
```

```
}
```

constructticklistwithgap Version 3.0: replaced `\dtl@constructticklistwithgap` with `\__dataplot_construct_tick_list_w`

`\__dataplot_construct_tick_list_over_zero:NNNN` ⟨*min-fp*⟩⟨*max-fp*⟩⟨*seq*⟩⟨*gap-fp*⟩

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and store in ⟨*list*⟩ (a control sequence) using the gap given by ⟨*gap*⟩ where the tick list straddles zero.

```
\cs_new:Nn \__dataplot_construct_tick_list_over_zero:NNNN
{
  \fp_compare:nNnF { #4 } < { \c_dataplot_smallest_gap_fp }
   {
     \seq_put_right:NV #3 \c_zero_fp
     \fp_set_eq:NN \l__datatool_tmpa_fp #4
     \fp_while_do:nn { \c_zero_fp < \l__datatool_tmpa_fp < #2 }
     {
       \seq_put_right:NV #3 \l__datatool_tmpa_fp
       \fp_add:Nn \l__datatool_tmpa_fp { #4 }
     }
     \dataplot_add_end_tick:NNNN
       \l__datatool_tmpa_fp #2 #4 #3
     \fp_compare:nF { #1 = \c_zero_fp }
     {
       \fp_set:Nn \l__datatool_tmpa_fp { - ( #4 ) }
       \fp_while_do:nn { #1 < \l__datatool_tmpa_fp < \c_zero_fp }
        {
          \seq_put_left:NV #3 \l__datatool_tmpa_fp
          \fp_sub:Nn \l__datatool_tmpa_fp { #4 }
        }
       \fp_compare:nNnT
          { #1 - \l__datatool_tmpa_fp } < { 0.5 * #4 }
        {
          \seq_put_left:NV #3 #1
        }
     }
   }
}
```

Construct minor list. Syntax: ⟨*major tic seq*⟩⟨*min fp*⟩⟨*max fp*⟩⟨*scale fp*⟩⟨*minor tic seq*⟩

```
\cs_new:Nn \__dataplot_construct_minor_ticks:NNNNN
{
```

Calculate minor gap, which will be stored in `\l__dataplot_gap_fp`

```
  \int_compare:nNnTF
    { \seq_count:N #1 } > { 3 }
   {
```

```
      \tl_set:Nx \l__dataplot_x_fp
        { \seq_item:Nn #1 { 2 } }
      \tl_set:Nx \l__dataplot_y_fp
        { \seq_item:Nn #1 { 3 } }
      \__dataplot_calc_minor_gap:NNN
        \l__dataplot_x_fp \l__dataplot_y_fp
        #4
    }
    {
      \int_compare:nNnTF
        { \seq_count:N #1 } > { \c_one_int }
       {
         \tl_set:Nx \l__dataplot_x_fp
          { \seq_item:Nn #1 { 1 } }
         \tl_set:Nx \l__dataplot_y_fp
          { \seq_item:Nn #1 { 2 } }
         \__dataplot_calc_minor_gap:NNN
          \l__dataplot_x_fp \l__dataplot_y_fp
          #4
       }
       {
         \__dataplot_calc_minor_gap:NNN
          #2 #3 #4
       }
    }
```

Previous tick

```
    \fp_set_eq:NN \l__dataplot_x_fp #2
    \seq_map_indexed_inline:Nn #1
     {
```

Current major tick

```
        \tl_set:Nn \l__dataplot_y_fp { ##2 }
        \int_compare:nNnT { ##1 } > { \c_one_int }
         {
```

Construct minor ticks between them:

```
           \int_compare:nNnTF { ##1 } = { 2 }
            {
              \__dataplot_construct_reverse_tick_list:NNNN
               \l__dataplot_x_fp
               \l__dataplot_y_fp
               #5
               \l__dataplot_gap_fp
            }
            {
              \__dataplot_construct_tick_list_with_gap_excl:NNNN
               \l__dataplot_x_fp
               \l__dataplot_y_fp
               #5
               \l__dataplot_gap_fp
```

```
            }
        }
```
Previous major tick for next iteration:
```
        \fp_set_eq:NN \l__dataplot_x_fp \l__dataplot_y_fp
    }
```
Final ticks
```
    \fp_set_eq:NN \l__dataplot_y_fp #3
    \fp_compare:nNnT
     { \l__dataplot_min_minor_gap_fp }
     <
     { \l__dataplot_y_fp - \l__dataplot_x_fp }
    {
      \__dataplot_construct_tick_list_with_gap_excl:NNNN
        \l__dataplot_x_fp
        \l__dataplot_y_fp
        #5
        \l__dataplot_gap_fp
    }
}
```
Calculate minor gap and stores the result in `\l__dataplot_gap_fp`
```
\cs_new:Nn \__dataplot_calc_minor_gap:NNN
{
  \fp_set:Nn \l__dataplot_width_fp
   {
     ( #2 - #1 ) * #3
   }
  \fp_set:Nn \l__dataplot_gap_fp
   { \l__dataplot_width_fp / 10 }
  \fp_compare:nT
   { \l__dataplot_gap_fp < \l__dataplot_min_minor_gap_fp }
  {
    \fp_set:Nn \l__dataplot_gap_fp
     { \l__dataplot_width_fp / 4}
    \fp_compare:nT
     { \l__dataplot_gap_fp < \l__dataplot_min_minor_gap_fp }
    {
      \fp_set:Nn \l__dataplot_gap_fp
       { \l__dataplot_width_fp / 2}
      \fp_compare:nT
       { \l__dataplot_gap_fp < \l__dataplot_min_minor_gap_fp }
      {
        \fp_set_eq:NN \l__dataplot_gap_fp \l__dataplot_width_fp
      }
    }
  }
  \fp_set:Nn \l__dataplot_gap_fp
   { \l__dataplot_gap_fp  / #3 }
}
```

onstructticklistwithgapz Version 3.0: replaced `\dtl@constructticklistwithgapz` with `\__dataplot_construct_tick_list_w`
⟨*min-fp*⟩⟨*max-fp*⟩⟨*seq*⟩⟨*gap-fp*⟩

l@constructminorticklist Version 3.0: removed

`\dtl@constructminorticklist{`⟨*min*⟩`}{`⟨*max*⟩`}{`⟨*scale factor*⟩`}{`⟨*list*⟩`}`

Constructs a list of minor tick points between ⟨*min*⟩ and ⟨*max*⟩ and append to ⟨*list*⟩ (a control sequence.)

`\__dataplot_construct_tick_list_with_gap_excl:NNNN` ⟨*min-fp*⟩⟨*max-fp*⟩⟨*seq*⟩⟨*gap-fp*⟩

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and appends to ⟨*seq*⟩ using the gap given by ⟨*gap*⟩ where the gap is given in user co-ordinates. The end points are not included in the list.

```
\cs_new:Nn \__dataplot_construct_tick_list_with_gap_excl:NNNN
{
  \fp_compare:nTF
    { #1 < \c_zero_fp < #2 }
   {
     \__dataplot_construct_tick_list_over_zero:NNNN
      #1 #2 #3 #4
   }
   {
    \fp_compare:nNnF { #4 } < { \c_dataplot_smallest_gap_fp }
     {
       \fp_set:Nn \l__datatool_tmpa_fp { #1 + #4 }
       \fp_while_do:nn { #1 < \l__datatool_tmpa_fp < #2 }
       {
         \seq_put_right:NV #3 \l__datatool_tmpa_fp
         \fp_add:Nn \l__datatool_tmpa_fp { #4 }
       }
     }
  }
}
```

Syntax: ⟨*min-fp*⟩⟨*max-fp*⟩⟨*seq*⟩⟨*gap-fp*⟩

```
\cs_new:Nn \__dataplot_construct_reverse_tick_list:NNNN
{
  \fp_compare:nNnF { #4 } < { \c_dataplot_smallest_gap_fp }
   {
     \fp_set:Nn \l__datatool_tmpa_fp { #2 - #4 }
     \fp_while_do:nn { #1 < \l__datatool_tmpa_fp < #2 }
     {
       \seq_put_left:NV #3 \l__datatool_tmpa_fp
       \fp_sub:Nn \l__datatool_tmpa_fp { #4 }
```

```
    }
  }
}
```

__dataplot_construct_tick_list_with_gap_incl:NNNN ⟨*min-fp*⟩⟨*max-fp*⟩⟨*seq*⟩⟨*gap-fp*⟩

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and appends to ⟨*seq*⟩ using the gap given by ⟨*gap*⟩ where the gap is given in user co-ordinates. The end points are included in the list if the gap is large enough.

```
\cs_new:Nn \__dataplot_construct_tick_list_with_gap_incl:NNNN
{
  \fp_compare:nTF
    { #1 < \c_zero_fp < #2 }
   {
     \__dataplot_construct_tick_list_over_zero:NNNN
      #1 #2 #3 #4
   }
   {
    \fp_compare:nNnF { #4 } < { \c_dataplot_smallest_gap_fp }
     {
       \fp_set:Nn \l__datatool_tmpa_fp { #1 + #4 }
       \seq_put_right:NV #3 #1
       \fp_while_do:nn { #1 < \l__datatool_tmpa_fp < #2 }
       {
         \seq_put_right:NV #3 \l__datatool_tmpa_fp
         \fp_add:Nn \l__datatool_tmpa_fp { #4 }
       }
       \fp_compare:nNnT
         { \l__datatool_tmpa_fp - #2 } < { 0.1 * #4 }
        {
          \seq_put_right:NV #3 #2
        }
     }
  }
}
```

nstructticklistwithgapex Version 3.0: removed

\dtl@constructticklistwithgapex{⟨*min*⟩}{⟨*max*⟩}{⟨*list*⟩}{⟨*gap*⟩}

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and store in ⟨*list*⟩ (a control sequence) using the gap given by ⟨*gap*⟩ where the gap is given in user co-ordinates. The end points are excluded from the list.

\DTLaddtoplotlegend

\DTLaddtoplotlegend{⟨*marker*⟩}{⟨*line style*⟩}{⟨*label*⟩}

Adds entry to legend.

```
\newcommand*{\DTLaddtoplotlegend}[3]{%
 \tl_if_empty:NF \l_dataplot_legend_tl
 {
   \tl_put_right:Nn \l_dataplot_legend_tl { \\ }
 }
 \tl_clear:N \l__datatool_tmpc_tl
 \tl_if_empty:nF { #2 }
 {
   \tl_if_eq:nnF { #2 } { \relax }
   {
     \tl_put_right:Nn \l__datatool_tmpc_tl
      {
        \begin{pgfscope}
          #2
          \pgfpathmoveto{\pgfpoint{-10pt}{0pt}}
          \pgfpathlineto{\pgfpoint{10pt}{0pt}}
          \pgfusepath{stroke}
        \end{pgfscope}
      }
    }
 }
 \tl_if_empty:nF { #1 }
 {
   \tl_if_eq:nnF { #1 } { \relax }
   {
     \tl_put_right:Nn \l__datatool_tmpc_tl { #1 }
   }
 }
 \tl_if_empty:NF \l__datatool_tmpc_tl
 {
   \tl_put_right:Nn \l_dataplot_legend_tl
     { \begin { pgfpicture } }
   \tl_put_right:NV \l_dataplot_legend_tl \l__datatool_tmpc_tl
   \tl_put_right:Nn \l_dataplot_legend_tl
     { \end { pgfpicture } }
   \tl_put_right:Nn \l_dataplot_legend_tl { & #3 }
 }
}
```

\dataplot_get_default_legend:Nnnnn ⟨*tl-var*⟩{⟨*database index*⟩}{⟨*database name*⟩}{⟨*x key*⟩}{⟨*y key*⟩}

Get the default legend label (if legend labels not set). The ⟨*tl-var*⟩ is the token list variable in which to store the legend label. The current x key index can be accessed

with `\l_dataplot_x_key_int` and the current y key index can be accessed with `\l_dataplot_x_key_int`.

```
\cs_new:Nn \dataplot_get_default_legend:Nnnnn
{
  \int_compare:nNnTF
   { \dataplot_x_key_count: } > { \c_one_int }
  {
    \tl_put_right:Nx #1
     {
       \exp_not:N \DTLplotlegendxy
        [ #2 ] { #3 }
        [ \int_use:N \l_dataplot_x_key_int ] { #4 }
        [ \int_use:N \l_dataplot_y_key_int ] { #5 }
     }
  }
  {
    \int_compare:nNnT
     { \dataplot_y_key_count: } > { \c_one_int }
    {
      \tl_put_right:Nx #1
       {
         \exp_not:N \DTLplotlegendy
          [ #2 ] { #3 }
          [ \int_use:N \l_dataplot_y_key_int ] { #5 }
       }
    }
  }
 \tl_if_empty:NTF #1
  {
    \tl_set:Nx #1
     {
       \exp_not:N \DTLplotlegendname [ #2 ] { #3 }
     }
  }
  {
    \int_compare:nNnT
       { \dataplot_db_count: } > { \c_one_int }
     {
       \tl_put_left:Nx #1
        {
          \exp_not:N \DTLplotlegendname [ #2 ] { #3 }
          \exp_not:N \DTLplotlegendnamesep
        }
     }
  }
}
```

In the following commands, ⟨*db index*⟩ is the index of the database name in the list supplied to `\DTLplot`, ⟨*x index*⟩ is the index of the ⟨*x key*⟩ in the `x` list, and ⟨*y index*⟩ the index of the ⟨*y key*⟩ in the `y` list. This is not the same as the column index (which

can be obtained with `\dtlcolumnindex{`⟨*db-name*⟩`}{`⟨*key*⟩`}`).

`\DTLplotlegendxy`

```
\DTLplotlegendxy[⟨db index⟩]{⟨db-name⟩}[⟨x index⟩]{⟨x key⟩}[⟨y index⟩]{⟨y key⟩}
```

x / y label in legend.

```
\NewDocumentCommand \DTLplotlegendxy { O{0} m O{0} m O{0} m }
 {
   \DTLplotlegendx [ #1 ] { #2 } [ #3 ] { #4 }
   \DTLplotlegendxysep
   \DTLplotlegendy [ #1 ] { #2 } [ #5 ] { #6 }
 }
```

`\DTLplotlegendxysep` Separator between x and y labels in legend.

```
\newcommand \DTLplotlegendxysep
 {
  \c_space_tl / \c_space_tl
 }
```

`\DTLplotlegendnamesep` Separator after name in legend.

```
\newcommand \DTLplotlegendnamesep { ~ }
```

`\DTLplotlegendname`

```
\DTLplotlegendxy[⟨db index⟩]{⟨db-name⟩}
```

Database label in legend.

```
\NewDocumentCommand \DTLplotlegendname { O{0} m }
 {
   \prop_get:NnNTF \l_dataplot_legend_names_prop
      { #2 } \l_dataplot_legend_name_tl
    { \l_dataplot_legend_name_tl }
    { #2 }
 }
```

`\DTLplotlegendsetname` Allow the user to supply a database name to legend text mapping.

```
\prop_new:N \l_dataplot_legend_names_prop
\tl_new:N \l_dataplot_legend_name_tl
\NewDocumentCommand \DTLplotlegendsetname { m m }
{
  \prop_put:Nnn \l_dataplot_legend_names_prop { #1 } { #2 }
}
```

`\DTLplotlegendx`

```
\DTLplotlegendx[⟨db index⟩]{⟨db-name⟩}[⟨x index⟩]{⟨x-key⟩}
```

x label in legend.

```
\NewDocumentCommand \DTLplotlegendx { O{0} m O{0} m }
 {
```

If a mapping has been supplied for the x key, use that.

```
    \prop_get:NnNTF \l_dataplot_legend_xlabels_prop
       { #4 } \l_dataplot_legend_xlabel_tl
     { \l_dataplot_legend_xlabel_tl }
     {
       \DTLaction
         [ name = { #2 }, key = { #4 } ]
         { column ~ data }
       \DTLuse { header }
     }
  }
```

`\DTLplotlegendsetxlabel` Allow the user to supply alternative text for the x label to legend text mapping.

```
\prop_new:N \l_dataplot_legend_xlabels_prop
\tl_new:N \l_dataplot_legend_xlabel_tl
\NewDocumentCommand \DTLplotlegendsetxlabel { m m }
{
  \prop_put:Nnn \l_dataplot_legend_xlabels_prop { #1 } { #2 }
}
```

`\DTLplotlegendy`

```
\DTLplotlegendy[⟨db index⟩]{⟨db-name⟩}[⟨y index⟩]
{⟨y-key⟩}
```

```
\NewDocumentCommand \DTLplotlegendy { O{0} m O{0} m }
 {
```

If a mapping has been supplied for the y key, use that.

```
    \prop_get:NnNTF
       \l_dataplot_legend_ylabels_prop
         { #4 }
       \l_dataplot_legend_ylabel_tl
     {
       \l_dataplot_legend_ylabel_tl
     }
     {
       \DTLaction
         [ name = { #2 }, key = { #4 } ]
         { column ~ data }
       \DTLuse { header }
     }
  }
```

`\DTLplotlegendsetylabel` Allow the user to supply alternative text for the y label to legend text mapping.

```
\prop_new:N \l_dataplot_legend_ylabels_prop
\tl_new:N \l_dataplot_legend_ylabel_tl
\NewDocumentCommand \DTLplotlegendsetylabel { m m }
{
  \prop_put:Nnn \l_dataplot_legend_ylabels_prop { #1 } { #2 }
}
```

The following commands may be used in the above definition to access additional information. Get the total number of databases:

```
\cs_new:Nn \dataplot_db_count:
{
  \seq_count:N \l__dataplot_dbnames_seq
}
```

Get the total number of x keys provided. NB this needs to count the original clist not the sequence which has items popped from it.

```
\cs_new:Nn \dataplot_x_key_count:
{
  \clist_count:N \l__dataplot_x_keys_clist
}
```

Get the total number of y keys provided:

```
\cs_new:Nn \dataplot_y_key_count:
{
  \seq_count:N \l__dataplot_y_keys_seq
}
```

```
\ExplSyntaxOff
```

30 person.sty

30.1 Package Declaration

Package identification:

```
\NeedsTeXFormat{LaTeX2e}
```

Rollback releases:

```
\DeclareRelease{v2.32}{2019-09-27}{person-2019-09-27.sty}
\DeclareCurrentRelease{v3.4.4}{2026-07-31}
```

Declare package:

```
\ProvidesPackage{person}[2026/07/31 v3.4.4 (NLCT)]
```

No longer requires the ifthen package. However, ifthen is automatically loaded by datatool-base, so this will have no noticeable effect. This package only really requires datatool-base not datatool. Provide an option to omit loading datatool.

```
\@person@datatoolsty
```

```
\newcommand\@person@datatoolsty{datatool}
```

Define shortcut commands.

```
\ExplSyntaxOn
\cs_new:Nn \__person_define_shortcuts:
{
  \newcommand \they { \peoplepronoun }
  \newcommand \They { \Peoplepronoun }
  \newcommand \them { \peopleobjpronoun }
  \newcommand \Them { \Peopleobjpronoun }
  \newcommand \their { \peoplepossadj }
  \newcommand \Their { \Peoplepossadj }
  \newcommand \theirs { \peopleposspronoun }
  \newcommand \Theirs { \Peopleposspronoun }
  \newcommand \you { \peoplepronounii }
  \newcommand \You { \Peoplepronounii }
  \newcommand \thee { \peopleobjpronounii }
  \newcommand \Thee { \Peopleobjpronounii }
  \newcommand \your { \peoplepossadjii }
  \newcommand \Your { \Peoplepossadjii }
  \newcommand \yours { \peopleposspronounii }
  \newcommand \Yours { \Peopleposspronounii }
  \newcommand \siblings { \peoplesibling }
  \newcommand \Siblings { \Peoplesibling }
  \newcommand \children { \peoplechild }
  \newcommand \Children { \Peoplechild }
  \newcommand \parents { \peopleparent }
  \newcommand \Parents { \Peopleparent }
  \cs_set:Nn \__person_define_shortcuts: { }
}
```

This will allow datatool-base and (if also required) datatool options to be passed to person:

```
\keys_define:nn { datatool }
{
  base-only .code:n =
   {
     \tl_set:Nn \@person@datatoolsty {datatool-base}
   } ,
  base-only .value_forbidden:n = true ,
  datatool .code:n =
   {
     \tl_set:Nn \@person@datatoolsty {datatool}
   } ,
  datatool .value_forbidden:n = true ,
  shortcuts .code:n = { \__person_define_shortcuts: } ,
  shortcuts .value_forbidden:n = true
}
\ExplSyntaxOff
```

Process options:

```
\IfPackageLoadedTF{\@person@datatoolsty}
```

```
{
  \ProcessKeyOptions[datatool]
}
{
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{\@person@datatoolsty}}
  \ProcessOptions
}

\RequirePackage{\@person@datatoolsty}

\ExplSyntaxOn
```

Remove the package option keys so they can't be used with `\DTLsetup`.

```
\keys_define:nn { datatool }
{
  base-only .undefine: ,
  datatool .undefine: ,
  shortcuts .undefine: ,
}
```

Default behaviour is to locally define people. This allows the throwaway anon label to be used within a scoped context.

```
\bool_new:N \l__person_local_bool
\bool_set_true:N \l__person_local_bool
```

Define options specific to person commands.

```
\keys_define:nn { datatool/person }
{
  local .bool_set:N = \l__person_local_bool ,
  global .bool_set_inverse:N = \l__person_local_bool ,
  shortcuts .code:n = { \__person_define_shortcuts: } ,
  shortcuts .value_forbidden:n = true
}
```

Allow these keys to be set in `\DTLsetup{person={...}}`

```
\keys_define:nn { datatool }
{
  person .code:n = { \keys_set:nn { datatool/person } { #1 } }
}
```

30.2 Defining People

`people` Keep count of the number of people who have been defined. This really should have been an internal variable, but retained for backward compatibility. The people label sequence can now simply be counted. (NB `\thepeople` was documented prior to v3.0.)

```
\newcounter{people}
```

`\PersonTotalCount`

```
\newcommand* \PersonTotalCount
  { \int_use:c { c@people } }
```

`\PersonMaleCount` Keep count of the number of male people who have been defined:

```
\int_new:N \l__person_male_int
\newcommand* \PersonMaleCount
  { \int_use:N \l__person_male_int }
```

`\PersonFemaleCount` Keep count of the number of female people who have been defined:

```
\int_new:N \l__person_female_int
\newcommand* \PersonFemaleCount
  { \int_use:N \l__person_female_int }
```

`\PersonNonBinaryCount` Keep count of the number of non-binary people who have been defined:

```
\int_new:N \l__person_nonbinary_int
\newcommand* \PersonNonBinaryCount
  { \int_use:N \l__person_nonbinary_int }
```

`\PersonUnknownCount` Keep count of the number of people who have been defined with the gender set to unknown:

```
\int_new:N \l__person_unknown_int
\newcommand* \PersonUnknownGenderCount
  { \int_use:N \l__person_unknown_int }
```

The label scratch variable is a public one to allow the label to be referenced in the `\newperson` hooks.

```
\tl_new:N \l_person_label_tl
```

Scratch variables.

```
\tl_new:N \l__person_full_name_tl
\tl_new:N \l__person_forenames_tl
\tl_new:N \l__person_name_tl
\tl_new:N \l__person_gender_tl
\tl_new:N \l__person_title_tl
\tl_new:N \l__person_surname_tl
```

`person` Version 3.0: Temporary counter person removed.

Define deprecated command. The deprecated command is provided rather than defined to help avoid conflict. Syntax: ⟨*old cmd*⟩⟨*new cmd*⟩

```
\cs_new:Nn \__person_define_deprecated_cmd:NN
{
 \providecommand #1
  {
  \PackageWarning { person }
   {
     \token_to_str:N #1 \c_space_tl ~ deprecated ~
     and ~ may ~ be ~ removed ~ in ~ future. ~ Use ~
     \token_to_str:N #2 \c_space_tl ~
     instead
   }
   #2
```

```
  }
}
```

`\@people@list` Keep a list of labels for each person who has been defined. Version 3.0: replaced `\@people@list` with:

```
\seq_new:N \l__person_people_seq
```

Internal constant gender identifiers. These are used to form token list variable names. The gender supplied when defining a person may be any of the defined labels, but it will be converted to the applicable constant value for internal use.

`\@male@label` Version 3.0: changed `\@male@label` to:

```
\tl_const:Nn \c_person_male_label_tl { male }
```

`\@female@label` Version 3.0: changed `\@female@label` to :

```
\tl_const:Nn \c_person_female_label_tl { female }
```

Non-binary:

```
\tl_const:Nn \c_person_nonbinary_label_tl { nonbinary }
```

Special case where the gender is unknown. This will usually be treated the same as non-binary.

```
\tl_const:Nn \c_person_unknown_label_tl { unknown }
```

Do something depending on whether none, one or multiple people defined.

```
\cs_new:Nn \__person_case:nnn
{
  \int_case:nnF { \seq_count:N \l__person_people_seq }
   {
     { \c_zero_int } { #1 }
     { \c_one_int } { #2 }
   }
   { #3 }
}
```

Set person label scratch variable:

```
\cs_new:Nn \person_set_label:n
{
  \__person_set_label:Nn \l_person_label_tl { #1 }
}
```

Supply token list variable:

```
\cs_new:Nn \__person_set_label:Nn
{
  \tl_set:Nx #1
   { \text_purify:n { \tl_trim_spaces:n { #2 } } }
}
```

`\@get@firstperson` Version 3.0 removed `\@get@firstperson`

`\@@get@firstperson` Version 3.0 removed `\@@get@firstperson`

`\malelabels` List of labels that can be used to indicate that a person is male (when defining a person using `\newperson`). Version 3.0: replaced `\malelabels` with an internal clist variable to reduce the possibility of a command name clash.

```
\clist_new:N \g_person_male_label_clist
```

`\PersonSetMaleLabels`

```
\NewDocumentCommand \PersonSetMaleLabels { m }
{
  \clist_gset:Ne \g_person_male_label_clist { #1 }
}
```

Initialise:

```
\PersonSetMaleLabels {Male,MALE,M,m}
```

`\PersonAddMaleLabel` Adds a label to the list of male labels.

```
\NewDocumentCommand \PersonAddMaleLabel { m }
{
  \clist_gput_right:Ne \g_person_male_label_clist { #1 }
}
```

`\addmalelabel` Old command.

```
\__person_define_deprecated_cmd:NN
  \addmalelabel \PersonAddMaleLabel
```

`\femalelabels` List of labels that can be used to indicate that a person is female (when defining a person using `\newperson`). Version 3.0: replaced `\femalelabels` with an internal clist variable to reduce the possibility of a command name clash.

```
\clist_new:N \g_person_female_label_clist
```

`\PersonSetFemaleLabels`

```
\NewDocumentCommand \PersonSetFemaleLabels { m }
{
  \clist_gset:Ne \g_person_female_label_clist { #1 }
}
```

Initialise:

```
\PersonSetFemaleLabels {Female,FEMALE,F,f}
```

`\addfemalelabel` Adds a label to the list of female labels.

```
\__person_define_deprecated_cmd:NN
 \addfemalelabel \PersonAddFemaleLabel
```

`\PersonAddFemaleLabel` Adds a label to the list of female labels.

```
\NewDocumentCommand \PersonAddFemaleLabel { m }
{
  \clist_gput_right:Ne \g_person_female_label_clist { #1 }
}
```

List of labels that can be used to indicate that a person is non-binary (when defining a person using \newperson).

```
\clist_new:N \g_person_nonbinary_label_clist
```

PersonSetNonBinaryLabels

```
\NewDocumentCommand \PersonSetNonBinaryLabels { m }
{
  \clist_gset:Ne \g_person_nonbinary_label_clist { #1 }
}
```

Initialise:

```
\PersonSetNonBinaryLabels
{
 non-binary, Nonbinary, Non-Binary, NONBINARY,
 NON-BINARY, N, n
}
```

\PersonAddNonBinaryLabel Adds a label to the list of non-binary labels.

```
\NewDocumentCommand \PersonAddNonBinaryLabel { m }
{
  \clist_gput_right:Ne \g_person_nonbinary_label_clist { #1 }
}
```

\PersonIfMaleLabel Determines if first argument is contained in the list of male labels. (One level expansion is performed on the first object in first argument.) This always first checks the constant \c_person_male_label_tl so that it doesn't need to be included in the male labels list. If true does second argument, otherwise does third argument.

```
\NewDocumentCommand \PersonIfMaleLabel { m m m }
{
 \exp_args:NNo \tl_if_eq:NnTF \c_person_male_label_tl { #1 }
  { #2 }
  {
    \clist_if_in:NoTF \g_person_male_label_clist { #1 } { #2 } { #3 }
  }
}
```

\ifmalelabel Version 3.0: deprecated.

```
\__person_define_deprecated_cmd:NN
 \ifmalelabel \PersonIfMaleLabel
```

\PersonIfFemaleLabel Determines if first argument is contained in the list of female labels. (One level expansion is performed on the first object in first argument.) This always first checks the constant \c_person_female_label_tl so that it doesn't need to be included in the \g_person_female_label_clist list. If true does second argument, otherwise does third argument.

```
\NewDocumentCommand \PersonIfFemaleLabel { m m m }
{
 \exp_args:NNo \tl_if_eq:NnTF \c_person_female_label_tl { #1 }
```

```
  { #2 }
  {
    \clist_if_in:NoTF \g_person_female_label_clist { #1 } { #2 } { #3 }
  }
}
```

`\iffemalelabel` Version 3.0: deprecated.

```
\__person_define_deprecated_cmd:NN
 \iffemalelabel \PersonIfFemaleLabel
```

`\PersonIfNonBinaryLabel` Determines if first argument is contained in the list of non-binary gender labels. (One level expansion is performed on the first object in first argument.) This always first checks the constant `\c_person_nonbinary_label_tl` so that it doesn't need to be included in the `\g_person_nonbinary_label_clist` list. If true, does second argument, otherwise does third argument.

```
\NewDocumentCommand \PersonIfNonBinaryLabel { m m m }
{
 \exp_args:NNo \tl_if_eq:NnTF
    \c_person_nonbinary_label_tl { #1 }
  { #2 }
  {
    \clist_if_in:NoTF \g_person_nonbinary_label_clist
      { #1 } { #2 } { #3 }
  }
}
```

`\newperson` Define a new person. The optional argument specifies a label with which to refer to that person. If omitted, `anon` is used. If more than one person is defined, the optional argument will be required to specify a unique label. The mandatory arguments are the person's full name, their familiar name and their gender.

Hook at the start of `\newperson`

```
\tl_new:N \g__person_new_init_tl
\cs_new:Nn \person_new_appto_start:n
{
 \tl_gput_right:N \g__person_new_init_tl { #1 }
}
```

Hook at the end of `\newperson`

```
\tl_new:N \g__person_new_finish_tl
\cs_new:Nn \person_new_appto_end:n
{
 \tl_gput_right:N \g__person_new_finish_tl { #1 }
}
```

Version 3.0 added a starred form which uses a key=value interface.

```
\NewDocumentCommand{\newperson}{ s O{anon} }
{
  \person_set_label:n { #2 }
  \tl_clear:N \l__person_full_name_tl
```

```
  \tl_clear:N \l__person_forenames_tl
  \tl_clear:N \l__person_name_tl
  \tl_clear:N \l__person_gender_tl
  \tl_clear:N \l__person_title_tl
  \tl_clear:N \l__person_surname_tl
  \g__person_new_init_tl
  \IfBooleanTF { #1 }
  { \__person_new:n }
  { \__person_new:nnn }
}

\cs_new:Nn \__person_new:nnn
{
  \person_if_exist:nTF { \l_person_label_tl }
  {
    \PackageError { person }
     {
        Person ~ ` \l_person_label_tl ' ~ has ~ already ~ been ~ defined
     }
     {
       Each ~ defined ~ person ~ must ~ have ~ a ~ unique ~ label
     }
  }
  {
    \tl_if_empty:NTF \l_person_label_tl
    {
      \PackageError { person }
       { Empty ~ person ~ label ~ not ~ permitted}
       {
         You ~ need ~ to ~ supply ~ a ~ non-empty ~
         label ~ to ~ identify ~ the ~ person
       }
    }
    {
      \tl_set:Ne \l__person_full_name_tl { \tl_trim_spaces:n { #1 } }
      \tl_set:Ne \l__person_name_tl { \tl_trim_spaces:n { #2 } }
      \datatool_if_null_or_empty:nTF { #3 }
       {
         \tl_set_eq:NN \l__person_gender_tl \c_person_unknown_label_tl
       }
       {
         \tl_set:Ne \l__person_gender_tl { \tl_trim_spaces:n { #3 } }
       }
      \__person_new:
    }
  }
}
```

Starred form has key=value interface. First define keys.

```
\keys_define:nn { datatool/person }
```

```
{
  fullname .tl_set:N = \l__person_full_name_tl ,
  expand-once-fullname .code:n =
   { \tl_set:No \l__person_full_name_tl { #1 } } ,
  expand-fullname .tl_set_x:N = \l__person_full_name_tl ,
  forenames .tl_set:N = \l__person_forenames_tl ,
  expand-once-forenames .code:n =
   { \tl_set:No \l__person_forenames_tl { #1 } } ,
  expand-forenames .tl_set_x:N = \l__person_forenames_tl ,
  name .tl_set:N = \l__person_name_tl ,
  expand-once-name .code:n =
   { \tl_set:No \l__person_name_tl { #1 } },
  expand-name .tl_set_x:N = \l__person_name_tl ,
  gender .code:n =
   {
    \datatool_if_null_or_empty:nTF { #1 }
     {
       \tl_set_eq:NN \l__person_gender_tl \c_person_unknown_label_tl
     }
     { \tl_set:Ne \l__person_gender_tl { #1 } }
   },
  title .tl_set:N = \l__person_title_tl ,
  expand-once-title .code:n =
   { \tl_set:No \l__person_title_tl { #1 } },
  expand-title .tl_set_x:N = \l__person_title_tl ,
  surname .tl_set:N = \l__person_surname_tl ,
  expand-once-surname .code:n =
   { \tl_set:No \l__person_surname_tl { #1 } },
  expand-surname .tl_set_x:N = \l__person_surname_tl ,
}
\cs_new:Nn \__person_new:n
{
  \person_if_exist:nTF { \l_person_label_tl }
  {
    \PackageError { person }
     {
        Person ~ ` \l_person_label_tl ' ~ has ~ already ~ been ~ defined
     }
     {
       Each ~ defined ~ person ~ must ~ have ~ a ~ unique ~ label
     }
  }
  {
    \tl_if_empty:NTF \l_person_label_tl
    {
      \PackageError { person }
       { Empty ~ person ~ label ~ not ~ permitted}
       {
         You ~ need ~ to ~ supply ~ a ~ non-empty ~
```

```
          label ~ to ~ identify ~ the ~ person
        }
     }
     {
       \keys_set:nn { datatool/person } { #1 }
       \__person_new:
     }
   }
}
\cs_new:Nn \__person_new:
{
  \bool_lazy_or:nnTF
      { \tl_if_empty_p:N \l__person_gender_tl }
      { \tl_if_eq_p:NN \l__person_gender_tl \c_person_unknown_label_tl }
    {
```

Unknown mostly behaves like non-binary but doesn't increment the nonbinary counter.

```
      \bool_if:NTF \l__person_local_bool
       {
         \tl_set:cV
          { person@ \l_person_label_tl @gender }
          \c_person_unknown_label_tl
       }
       {
         \tl_gset:cV
          { person@ \l_person_label_tl @gender }
          \c_person_unknown_label_tl
       }
    }
    {
      \exp_args:NV \PersonIfMaleLabel \l__person_gender_tl
       {
         \bool_if:NTF \l__person_local_bool
          {
            \tl_set:cV
             { person@ \l_person_label_tl @gender }
             \c_person_male_label_tl
            \int_incr:N \l__person_male_int
          }
          {
            \tl_gset:cV
             { person@ \l_person_label_tl @gender }
             \c_person_male_label_tl
            \int_gincr:N \l__person_male_int
          }
       }
       {
         \exp_args:NV \PersonIfFemaleLabel \l__person_gender_tl
          {
            \bool_if:NTF \l__person_local_bool
```

```
    {
      \tl_set:cV
       { person@ \l_person_label_tl @gender }
       \c_person_female_label_tl
      \int_incr:N \l__person_female_int
    }
    {
      \tl_gset:cV
       { person@ \l_person_label_tl @gender }
       \c_person_female_label_tl
      \int_gincr:N \l__person_female_int
    }
  }
  {

    \exp_args:NV \PersonIfNonBinaryLabel \l__person_gender_tl
    {
      \bool_if:NTF \l__person_local_bool
       {
        \tl_set:cV
         { person@ \l_person_label_tl @gender }
         \c_person_nonbinary_label_tl
        \int_incr:N \l__person_nonbinary_int
       }
       {
        \tl_gset:cV
         { person@ \l_person_label_tl @gender }
         \c_person_nonbinary_label_tl
        \int_gincr:N \l__person_nonbinary_int
       }
    }
    {
      \tl_if_eq:NN \l__person_gender_tl \c_person_unknown_label_tl
      {
        \bool_if:NTF \l__person_local_bool
         {
          \tl_set:cV
           { person@ \l_person_label_tl @gender }
           \c_person_unknown_label_tl
          \int_incr:N \l__person_unknown_int
         }
         {
          \tl_gset:cV
           { person@ \l_person_label_tl @gender }
           \c_person_unknown_label_tl
          \int_gincr:N \l__person_unknown_int
         }
      }
      {
        \PackageError { person }
```

```
                {
                  Unknown ~ gender ~ ` \l__person_gender_tl ' ~
                  for ~ person ~ ` \l_person_label_tl '
                }
                {
                  Known ~ gender ~ identifiers ~ are: ~
                  \clist_use:Nn \g_person_male_label_clist { , ~ }  ~
                  or ~ \clist_use:Nn \g_person_female_label_clist { , ~ } ~
                  or ~ \clist_use:Nn \g_person_nonbinary_label_clist { , ~ }
                  or ~ \c_person_unknown_label_tl
                }
               \bool_if:NTF \l__person_local_bool
                {
                  \tl_set:cV
                    { person@ \l_person_label_tl @gender }
                    \c_person_unknown_label_tl
                  \int_incr:N \l__person_unknown_int
                }
                {
                  \tl_gset:cV
                    { person@ \l_person_label_tl @gender }
                    \c_person_unknown_label_tl
                  \int_gincr:N \l__person_unknown_int
                }
             }
           }
         }
       }
 }
\tl_if_empty:NT \l__person_full_name_tl
 {
   \tl_if_empty:NTF \l__person_forenames_tl
    {
     \tl_if_empty:NTF \l__person_name_tl
      {
        \tl_if_empty:NTF \l__person_title_tl
         {
           \tl_set_eq:NN
             \l__person_full_name_tl
             \l__person_surname_tl
         }
         {
          \tl_set:Nx \l__person_full_name_tl
           {
             \exp_not:V \l__person_title_tl
             \exp_not:N \persontitlesurnamesep
             \exp_not:V \l__person_surname_tl
           }
         }
      }
```

```
      {
       \tl_if_empty:NTF \l__person_surname_tl
        {
          \tl_set_eq:NN
            \l__person_full_name_tl
            \l__person_name_tl
        }
        {
          \tl_set:Nx \l__person_full_name_tl
           {
            \exp_not:V \l__person_name_tl
            \c_space_tl
            \exp_not:V \l__person_surname_tl
           }
        }
      }
    }
    {
       \tl_if_empty:NTF \l__person_surname_tl
        {
          \tl_set_eq:NN
            \l__person_full_name_tl
            \l__person_forenames_tl
        }
        {
          \tl_set:Nx \l__person_full_name_tl
           {
            \exp_not:V \l__person_forenames_tl
            \c_space_tl
            \exp_not:V \l__person_surname_tl
           }
        }
    }
 }
\tl_if_empty:NT \l__person_forenames_tl
 {
   \tl_set_eq:NN
     \l__person_forenames_tl
     \l__person_name_tl
 }
\bool_if:NTF \l__person_local_bool
 {
   \tl_set_eq:cN
     { person@ \l_person_label_tl @name }
     \l__person_name_tl
   \tl_set_eq:cN
     { person@ \l_person_label_tl @surname }
     \l__person_surname_tl
   \tl_set_eq:cN
     { person@ \l_person_label_tl @title }
```

```
      \l__person_title_tl
    \tl_set_eq:cN
      { person@ \l_person_label_tl @fullname }
      \l__person_full_name_tl
    \tl_set_eq:cN
      { person@ \l_person_label_tl @forenames }
      \l__person_forenames_tl
    \seq_put_right:NV \l__person_people_seq \l_person_label_tl
    \int_incr:N \c@people
  }
  {
    \tl_gset_eq:cN
      { person@ \l_person_label_tl @name }
      \l__person_name_tl
    \tl_gset_eq:cN
      { person@ \l_person_label_tl @surname }
      \l__person_surname_tl
    \tl_gset_eq:cN
      { person@ \l_person_label_tl @title }
      \l__person_title_tl
    \tl_gset_eq:cN
      { person@ \l_person_label_tl @fullname }
      \l__person_full_name_tl
    \tl_gset_eq:cN
      { person@ \l_person_label_tl @forenames }
      \l__person_forenames_tl
    \seq_gput_right:NV \l__person_people_seq \l_person_label_tl
    \int_gincr:N \c@people
  }
 \g__person_new_finish_tl
}
```

Syntax: {⟨*label*⟩}{⟨*attribute name*⟩}{⟨*value*⟩}

```
\cs_new:Nn \person_set_attribute:nnn
{
  \bool_if:NTF \l__person_local_bool
   {
     \tl_set_eq:cn { person@ #1 @ #2 } { #3 }
   }
   {
     \tl_gset_eq:cn { person@ #1 @ #2 } { #3 }
   }
}
\cs_generate_variant:Nn \person_set_attribute:nnn { nnV }
```

Syntax: {⟨*person-label*⟩}{⟨*attribute*⟩}

```
\cs_new:Nn \person_get_attribute:nn
{
  \tl_use:c { person@ #1 @ #2 }
}
```

Syntax: ⟨*tl var*⟩ {⟨*person-label*⟩}{⟨*attribute*⟩}

```
\cs_new:Nn \person_get_attribute:Nnn
{
  \tl_seq_eq:Nc #1 { person@ #2 @ #3 }
}
```

Syntax: {⟨*person-label*⟩}{⟨*attribute*⟩}

```
\cs_new:Nn \person_unset_attribute:nn
{
  \bool_if:NTF \l__person_local_bool
   {
     \csundef { person@ #1 @ #2 }
   }
   {
     \csgundef { person@ #1 @ #2 }
   }
}
```

30.3 Remove People

`\removeperson` Removes person identified by their label from the list.

```
\NewDocumentCommand \removeperson { O{anon} }
{
  \person_set_label:n { #1 }
  \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \__person_remove:x { \l_person_label_tl }
  }
}
```

Internal commands don't check for existence.

```
\cs_new:Nn \__person_remove:
{
  \bool_if:NTF \l__person_local_bool
   {
    \seq_remove_all:NV \l__person_people_seq \l_person_label_tl
    \int_decr:N \c@people
```

If the gender is unknown there won't be an associated counter.

```
    \int_if_exist:cT
     { l__person_ \csuse{ person@ \l_person_label_tl @gender } _int }
     {
       \int_decr:c
        { l__person_ \csuse { person@ \l_person_label_tl @gender } _int }
     }
```

Undefine associated control sequences:

```
    \csundef { person@ \l_person_label_tl @name }
    \csundef { person@ \l_person_label_tl @fullname }
    \csundef { person@ \l_person_label_tl @forenames }
```

```
    \csundef { person@ \l_person_label_tl @gender }
    \csundef { person@ \l_person_label_tl @title }
    \csundef { person@ \l_person_label_tl @surname }
   }
   {
```

Globally remove label from list of people.

```
    \seq_gremove_all:NV \l__person_people_seq \l_person_label_tl
```

Decrement number of people:

```
    \int_gdecr:N \c@people
```

Decrement gender counter if it exists:

```
    \int_if_exist:cT
     { l__person_ \csuse{ person@ \l_person_label_tl @gender } _int }
     {
       \int_gdecr:c
        { l__person_ \csuse { person@ \l_person_label_tl @gender } _int }
     }
```

Undefine associated control sequences:

```
    \csgundef { person@ \l_person_label_tl @name }
    \csgundef { person@ \l_person_label_tl @fullname }
    \csgundef { person@ \l_person_label_tl @forenames }
    \csgundef { person@ \l_person_label_tl @gender }
    \csgundef { person@ \l_person_label_tl @title }
    \csgundef { person@ \l_person_label_tl @surname }
   }
  \g__person_remove_tl
}
\cs_new:Nn \__person_remove:n
{
  \person_set_label:n { #1 }
  \__person_remove:
}
\cs_generate_variant:Nn \__person_remove:n { x }
```

Hook for remove person:

```
\tl_new:N \g__person_remove_tl
\cs_new:Nn \person_remove_appto:n
{
 \tl_gput_right:N \g__person_remove_tl { #1 }
}
```

`\removepeople` Removes the people listed.

```
\NewDocumentCommand \removepeople { m }
{
  \clist_map_function:nN { #1 } \__person_remove:n
}
```

`\removeallpeople` Removes everyone.

```
\NewDocumentCommand \removeallpeople { }
{
  \bool_if:NTF \l__person_local_bool
   {
    \seq_map_inline:Nn  \l__person_people_seq
     {
      \person_set_label:n { ##1 }
      \csundef{person@ \l_person_label_tl @name}
      \csundef{person@ \l_person_label_tl @fullname}
      \csundef{person@ \l_person_label_tl @forenames}
      \csundef{person@ \l_person_label_tl @gender}
      \csundef{person@ \l_person_label_tl @title}
      \csundef{person@ \l_person_label_tl @surname}
      \g__person_remove_tl
     }
    \int_zero:c { c@people }
    \int_zero:N \l__person_male_int
    \int_zero:N \l__person_female_int
    \int_zero:N \l__person_nonbinary_int
    \seq_clear:N \l__person_people_seq
   }
   {
    \seq_map_inline:Nn  \l__person_people_seq
     {
      \person_set_label:n { ##1 }
      \csgundef{person@ \l_person_label_tl @name}
      \csgundef{person@ \l_person_label_tl @fullname}
      \csgundef{person@ \l_person_label_tl @forenames}
      \csgundef{person@ \l_person_label_tl @gender}
      \csgundef{person@ \l_person_label_tl @title}
      \csgundef{person@ \l_person_label_tl @surname}
      \g__person_remove_tl
     }
    \int_gzero:c { c@people }
    \int_gzero:N \l__person_male_int
    \int_gzero:N \l__person_female_int
    \int_gzero:N \l__person_nonbinary_int
    \seq_gclear:N \l__person_people_seq
   }
}
```

30.4 Conditionals and Loops

`\ifpersonexists` If person whose label is given by the first argument exists, then do the second argument otherwise do third argument.

```
\newcommand{\ifpersonexists}[3]{
 \exp_args:Ne \person_if_exist:nTF
  { \tl_trim_spaces:n { #1 } }
  { #2 } { #3 }
```

```
}

\prg_new_conditional:Npnn \person_if_exist:n #1 { p, T, F, TF }
{
  \tl_if_exist:cTF { person@ #1 @name}
  { \prg_return_true: }
  { \prg_return_false: }
}
\cs_new:Nn \person_if_exist_or_err:nT
{
 \person_if_exist:nTF
  { #1 } { #2 }
  {
    \PackageError { person }
     {
       Person ~ ` #1 ' ~ doesn't ~ exist
     }
     {
       Check ~ that ~ you ~ have ~ correctly ~ spelt ~ the ~ label
     }
  }
}
```

Case-command. Syntax: ⟨*gender tl-var*⟩ {⟨*invalid*⟩} {⟨*male*⟩} {⟨*female*⟩} {⟨*non-binary*⟩} {⟨*unknown*⟩}

```
\cs_new:Nn \person_gender_case:Nnnnnn
{
  \tl_if_eq:NNTF #1 \c_person_male_label_tl
   { #3 }
   {
    \tl_if_eq:NNTF #1 \c_person_female_label_tl
     { #4 }
     {
      \tl_if_eq:NNTF #1 \c_person_nonbinary_label_tl
       { #5 }
       {
         \tl_if_eq:NNTF #1 \c_person_unknown_label_tl
          { #6 } { #2 }
       }
     }
   }
}
```

As above but label supplied as a token list.

```
\cs_new:Nn \person_gender_case:nnnnnn
{
  \tl_if_eq:NnTF \c_person_male_label_tl { #1 }
   { #3 }
   {
    \tl_if_eq:NnTF \c_person_female_label_tl { #1 }
```

```
      { #4 }
      {
       \tl_if_eq:NnTF \c_person_nonbinary_label_tl { #1 }
        { #5 }
        {
          \tl_if_eq:NnTF \c_person_unknown_label_tl { #1 }
           { #6 } { #2 }
        }
      }
    }
}
\cs_generate_variant:Nn \person_gender_case:nnnnnn
 { xnnnnn }
```

Similar but issue an error for invalid case and treat as unknown:

```
\cs_new:Nn \person_gender_case:Nnnnn
{
  \person_gender_case:Nnnnnn #1
   {
     \PackageError { person }
      {
        Unrecognised ~ internal ~ gender ~ label ~ `#1'. ~
        Treating ~ as ~ ` \c_person_unknown_label_tl '
      }
      {
        Internal ~ gender ~ labels ~ must ~ be ~ one ~ of ~
        the ~ following: ` \c_person_male_label_tl ', ~
        ` \c_person_female_label_tl ', ~
        ` \c_person_nonbinary_label_tl ', ~
        ` \c_person_unknown_label_tl '
      }
     #5
   }
   { #2 } { #3 } { #4 } { #5 }
}
```

Gender label provided as a token list:

```
\cs_new:Nn \person_gender_case:nnnnn
{
  \person_gender_case:nnnnnn { #1 }
   {
     \PackageError { person }
      {
        Unrecognised ~ internal ~ gender ~ label ~ `#1'. ~
        Treating ~ as ~ ` \c_person_unknown_label_tl '
      }
      {
        Internal ~ gender ~ labels ~ must ~ be ~ one ~ of ~
        the ~ following: ` \c_person_male_label_tl ', ~
        ` \c_person_female_label_tl ', ~
        ` \c_person_nonbinary_label_tl ', ~
```

```
          ` \c_person_unknown_label_tl '
        }
       #5
    }
    { #2 } { #3 } { #4 } { #5 }
}
\cs_generate_variant:Nn  \person_gender_case:nnnnn
 { xnnnn }
```

Just need to know if the label is valid:

```
\cs_new:Nn \person_do_if_valid_gender:nT
{
  \person_gender_case:nnnnnn { #1 }
   {
     \PackageError { person }
      {
        Unrecognised ~ internal ~ gender ~ label ~ `#1'
      }
      {
        Internal ~ gender ~ labels ~ must ~ be ~ one ~ of ~
        the ~ following: ` \c_person_male_label_tl ', ~
        ` \c_person_female_label_tl ', ~
        ` \c_person_nonbinary_label_tl ', ~
        ` \c_person_unknown_label_tl '
      }
   }
   { #2 } { #2 } { #2 } { #2 }
}
\cs_generate_variant:Nn  \person_do_if_valid_gender:nT
 { xT }
```

Test if all defined people the same gender. Syntax: {⟨*all male*⟩}{⟨*all female*⟩}{⟨*all non-binary*⟩}{⟨*other*⟩}

```
\cs_new:Nn \person_all_gender_case:nnnn
{
  \int_compare:nNnTF
    { \c@people } = { \l__person_male_int }
  { #1 }
  {
    \int_compare:nNnTF
      { \c@people } = { \l__person_female_int }
    { #2 }
    {
      \int_compare:nNnTF
        { \c@people } = { \l__person_nonbinary_int }
      { #3 } { #4 }
    }
  }
}
```

`\PersonIfMale` If the person given by the label in the first argument is male, do the second argument,

otherwise do the third argument.

```
\NewDocumentCommand \PersonIfMale { m m m }
{
  \person_set_label:n { #1 }
  \person_if_exist_or_err:nT { \l_person_label_tl  }
   {
     \tl_if_eq:cNTF
      { person@ \l_person_label_tl @gender }
       \c_person_male_label_tl
      {#2} {#3}
   }
}
```

`\ifmale` If the person given by the label in the first argument is male, do the second argument, otherwise do the third argument.

```
\__person_define_deprecated_cmd:NN
 \ifmale \PersonIfMale
```

`\PersonIfAllMale` If all people listed in first argument are male, do the second argument otherwise do the third argument. If the first argument is omitted, all defined people are checked.

```
\NewDocumentCommand \PersonIfAllMale { o m m }
{
 \IfNoValueTF { #1 }
 {
   \int_compare:nNnTF
     { \c@people } = { \l__person_male_int }
    { #2 } { #3 }
 }
 {
   \cs_set_eq:NN \__person_do:TF \cs_use_i:nn
   \clist_map_inline:nn { #1 }
    {
      \tl_if_eq:cNF
        { person@ \text_purify:n { ##1 } @gender}
        \c_person_male_label_tl
      {
        \cs_set_eq:NN \__person_do:TF \cs_use_ii:nn
        \clist_map_break:
      }
    }
   \__person_do:TF { #2 } { #3 }
 }
}
```

`\ifallmale`

```
\__person_define_deprecated_cmd:NN
 \ifallmale \PersonIfAllMale
```

`\PersonIfFemale` If the person given by the label in the first argument is female, do the second argument, otherwise do the third argument.

```
\NewDocumentCommand \PersonIfFemale { m m m }
{
  \person_set_label:n { #1 }
  \person_if_exist_or_err:nT { \l_person_label_tl }
   {
     \tl_if_eq:cNTF
      { person@ \l_person_label_tl @gender }
       \c_person_female_label_tl
      {#2} {#3}
   }
}
```

`\iffemale` If the person given by the label in the first argument is female, do the second argument, otherwise do the third argument.

```
\__person_define_deprecated_cmd:NN
 \iffemale \PersonIfFemale
```

`\PersonIfAllFemale` If all people listed in first argument are female, do the second argument otherwise do the third argument.

```
\NewDocumentCommand \PersonIfAllFemale { o m m }
{
 \IfNoValueTF { #1 }
 {
   \int_compare:nNnTF
     { \c@people } = { \l__person_female_int }
    { #2 } { #3 }
 }
 {
   \cs_set_eq:NN \__person_do:TF \cs_use_i:nn
   \clist_map_inline:nn { #1 }
    {
      \tl_if_eq:cNF
        { person@ \text_purify:n { ##1 } @gender}
        \c_person_female_label_tl
      {
        \cs_set_eq:NN \__person_do:TF \cs_use_ii:nn
        \clist_map_break:
      }
    }
   \__person_do:TF { #2 } { #3 }
 }
}
```

`\ifallfemale` If all people listed in first argument are female, do the second argument otherwise do the third argument.

```
\__person_define_deprecated_cmd:NN
 \ifallfemale \PersonIfAllFemale
```

`\PersonIfNonBinary` If the person given by the label in the first argument is non-binary, do the second argument, otherwise do the third argument. NB this does false for `unknown`.

```
\NewDocumentCommand \PersonIfNonBinary { m m m }
{
  \person_set_label:n { #1 }
  \person_if_exist_or_err:nT { \l_person_label_tl }
   {
     \tl_if_eq:cNTF
      { person@ \l_person_label_tl @gender }
       \c_person_nonbinary_label_tl
      { #2 } { #3 }
   }
}
```

`\PersonIfAllNonBinary` If all people listed in first argument are nonbinary, do the second argument otherwise do the third argument. NB this does false for `unknown`.

```
\NewDocumentCommand \PersonIfAllNonBinary { o m m }
{
 \IfNoValueTF { #1 }
 {
   \int_compare:nNnTF
     { \c@people } = { \l__person_nonbinary_int }
    { #2 } { #3 }
 }
 {
   \cs_set_eq:NN \__person_do:TF \cs_use_i:nn
   \clist_map_inline:nn { #1 }
    {
      \tl_if_eq:cNF
        { person@ \text_purify:n { ##1 } @gender}
        \c_person_nonbinary_label_tl
      {
        \cs_set_eq:NN \__person_do:TF \cs_use_ii:nn
        \clist_map_break:
      }
    }
   \__person_do:TF { #2 } { #3 }
 }
}
```

`\PersonIfUnknownGender` If the person given by the label in the first argument has the gender set to unknown, do the second argument, otherwise do the third argument.

```
\NewDocumentCommand \PersonIfUnknownGender { m m m }
{
  \person_set_label:n { #1 }
  \person_if_exist_or_err:nT { \l_person_label_tl }
   {
     \tl_if_eq:cNTF
      { person@ \l_person_label_tl @gender }
       \c_person_unknown_label_tl
      { #2 } { #3 }
```

```
    }
}
```

`PersonIfAllUnknownGender` If all people listed in first argument have the gender set to unknown, do the second argument otherwise do the third argument.

```
\NewDocumentCommand \PersonIfAllUnknownGender { o m m }
{
 \IfNoValueTF { #1 }
 {
   \int_compare:nNnTF
     { \c@people } = { \l__person_unknown_int }
    { #2 } { #3 }
 }
 {
   \cs_set_eq:NN \__person_do:TF \cs_use_i:nn
   \clist_map_inline:nn { #1 }
    {
      \tl_if_eq:cNF
        { person@ \text_purify:n { ##1 } @gender}
        \c_person_unknown_label_tl
      {
        \cs_set_eq:NN \__person_do:TF \cs_use_ii:nn
        \clist_map_break:
      }
    }
   \__person_do:TF { #2 } { #3 }
 }
}
```

`\foreachpersonbreak` Break out of loops.

```
\newcommand{\foreachpersonbreak}
{
  \PackageError { person }
  {
    \token_to_str:N \foreachpersonbreak \c_space_tl ~
    can't ~ be ~ used ~ outside ~ of
    \token_to_str:N \forallpeople \c_space_tl ~
    or \token_to_str:N \foreachperson
  }
  { }
}
```

`\foreachperson`

```
\foreachperson(⟨name cs⟩,⟨full name cs⟩,⟨gender cs⟩,⟨label cs⟩)\in{⟨list⟩}\do{⟨body⟩}
```

Iterates through list of people the `\in{`⟨*list*⟩`}` is optional. If omitted, the list of all defined people is used.

```
\newcommand \foreachperson { \__person_foreach:wn }
```

```
\cs_new:Npn \__person_foreach:wn ( #1 , #2 , #3 , #4 ) #5
{
  \tl_if_eq:nnTF { #5 } { \in }
   {
     \__person_foreach:NNNNnn #1 #2 #3 #4
   }
   {
     \renewcommand \foreachpersonbreak { \seq_break: }
     \seq_map_inline:Nn \l__person_people_seq
      {
        \__person_foreach_fn:NNNNnn #1 #2 #3 #4 { ##1 } { #5 }
      }
   }
}
```

List of labels provided:

```
\cs_new:Nn \__person_foreach:NNNNnn
{
  \renewcommand \foreachpersonbreak { \clist_break: }
  \clist_map_inline:nn { #5 }
   {
     \__person_foreach_fn:NNNNxn #1 #2 #3 #4
       { \text_purify:n { \tl_trim_spaces:n { ##1 } } } { #6 }
   }
}
```

Loop function. Syntax: ⟨*name-tl-var*⟩ ⟨*full-name-tl-var*⟩ ⟨*gender-tl-var*⟩ ⟨*label-tl-var*⟩ {⟨*label*⟩} {⟨*body*⟩}

```
\cs_new:Nn \__person_foreach_fn:NNNNnn
{
  \person_if_exist_or_err:nT { #5 }
   {
     \tl_set:Nn #4 { #5 }
     \tl_set_eq:Nc #1 { person@ #5 @name }
     \tl_set_eq:Nc #2 { person@ #5 @fullname }
     \tl_set:Nx #3 { \__person_language:nn { #5 } { gender } }
     #6
   }
}
\cs_generate_variant:Nn \__person_foreach_fn:NNNNnn
 { NNNNxn }
```

`\@foreachperson` Version 3.0: removed `\@foreachperson`

`\forallpeople` `\forallpeople[`⟨*list*⟩`]{`⟨*label-cs*⟩`}{`⟨*body*⟩`}`

```
\NewDocumentCommand \forallpeople { o m +m }
{
```

```
  \IfValueTF { #1 }
   {
     \renewcommand \foreachpersonbreak { \clist_break: }
     \clist_map_inline:nn { #1 }
      {
        \__person_set_label:Nn #2 { ##1 }
        \person_if_exist_or_err:nT { #2 }
         { #3 }
      }
   }
   {
     \renewcommand \foreachpersonbreak { \seq_break: }
     \seq_map_inline:Nn \l__person_people_seq
      {
        \tl_set:Nn #2 { ##1 }
```

The person should exist unless something unexpected has occurred.

```
        \person_if_exist_or_err:nT { #2 }
         { #3 }
      }
   }
}
```

30.5 Predefined Words

These commands should be redefined if you are writing in another language, but note that these are structured according to English grammar.

Version 3.0 has renamed some commands to help avoid conflict. The old command names are retained for backward-compatibility but may be removed in a later version.

\PersonSetLocalisation

`\PersonSetLocalisation{`⟨*gender*⟩`}{`⟨*type*⟩`}{`⟨*value*⟩`}`

These are now token list variables in the form `\g__person_`⟨*gender*⟩`_`⟨*type*⟩`_tl` and can be set with the following command. NB the gender label must be one of the internal labels (“male”, “female”, “nonbinary” or “unknown”).

```
\NewDocumentCommand \PersonSetLocalisation
   { m m m }
{
 \person_do_if_valid_gender:nT { #1 }
  {
    \tl_gset:cn { g__person_ #1 _ #2 _tl } { #3 }
  }
}
```

Define new localisation token list variable:

```
\cs_new:Nn \person_new_localisation:nnn
{
```

```
 \person_do_if_valid_gender:nT { #1 }
  {
    \tl_new:c { g__person_ #1 _ #2 _tl }
    \tl_gset:cn { g__person_ #1 _ #2 _tl } { #3 }
  }
}
```

Get localisation token list variable.

```
\cs_new:Nn \person_get_localisation:nn
{
  \tl_if_exist:cTF {  g__person_ #1 _ #2 _tl }
   { \tl_use:c { g__person_ #1 _ #2 _tl } }
   {
     \tl_if_exist:cTF { g__person_unknown_ #2 _tl }
      {
        \tl_use:c { g__person_unknown_ #2 _tl }
      }
      {
       \tl_if_exist:cTF { g__person_nonbinary_ #2 _tl }
        {
          \tl_use:c { g__person_nonbinary_ #2 _tl }
        }
        {
          ??
          \NoCaseChange
           {
             \PackageWarning { person }
              {
                Unknown ~ localisation ~ combination: ~
                gender = ` #1 ' ~ and ~ type = ` #2 '
              }
           }
        }
      }
   }
}
\cs_generate_variant:Nn \person_get_localisation:nn { vn, Vn }
```

30.5.1 Third person pronouns (subject)

```
\cs_new:Nn \__person_deprecated_lang_cs:Nnnn
 {
   \tl_if_exist:NTF #1
    {
      \person_new_localisation:nnn { #2 } { #3 } { #4 }
    }
    {
      \newcommand* #1 { #4 }
      \person_new_localisation:nnn { #2 } { #3 } { #1 }
    }
```

```
}
```

`\malepronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \malepronoun
  { male } { pronoun } { he }
```

`\femalepronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femalepronoun
   { female } { pronoun } { she }
```

Non-binary pronoun:

```
\person_new_localisation:nnn { nonbinary } { pronoun } { they }
```

`\pluralpronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \pluralpronoun
  { unknown } { pluralpronoun } { they }
```

Pronouns (object).

`\maleobjpronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \maleobjpronoun
  { male } { objpronoun } { him }
```

`\femaleobjpronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femaleobjpronoun
  { female } { objpronoun } { her }

\person_new_localisation:nnn
  { nonbinary } { objpronoun } { them }
```

`\pluralobjpronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \pluralobjpronoun
  { unknown } { pluralobjpronoun } { them }
```

Pronouns (possessive adjective).

`\malepossadj` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \malepossadj
  { male } { possadj } { his }
```

`\femalepossadj` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femalepossadj
  { female } { possadj } { her }
```

```
\person_new_localisation:nnn
  { nonbinary } { possadj } { their }
```

`\pluralpossadj` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \pluralpossadj
  { unknown } { pluralpossadj } { their }
```

Pronouns (possessive pronoun).

`\maleposspronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \maleposspronoun
  { male } { posspronoun } { his }
```

`\femaleposspronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femaleposspronoun
  { female } { posspronoun } { hers }

\person_new_localisation:nnn
  { nonbinary } { posspronoun } { theirs }
```

`\pluralposspronoun` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \pluralposspronoun
  { unknown } { pluralposspronoun } { theirs }
```

30.5.2 Second person

These are the same for all genders in English, but version 3.0 now provides support in the event of other languages.

```
\person_new_localisation:nnn { unknown } { pronoun2 } { you }
\person_new_localisation:nnn { unknown } { pluralpronoun2 } { you }
```

Objective pronoun.

```
\person_new_localisation:nnn
  { unknown } { objpronoun2 } { you }
\person_new_localisation:nnn
  { unknown } { pluralobjpronoun2 } { you }
```

Possessive adjective.

```
\person_new_localisation:nnn
  { unknown } { possadj2 } { your }
\person_new_localisation:nnn
  { unknown } { pluralpossadj2 } { your }
```

Possessive pronoun.

```
\person_new_localisation:nnn
  { unknown } { posspronoun2 } { yours }
\person_new_localisation:nnn
  { unknown } { pluralpossprounoun2 } { yours }
```

30.5.3 Relationship

`\malechild` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \malechild
  { male } { child } { son }
```

`\femalechild` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femalechild
  { female } { child } { daughter }

\person_new_localisation:nnn
  { nonbinary } { child } { child }
```

`\pluralchild` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \pluralchild
  { unknown } { pluralchild } { children }
```

`\malechildren` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \malechildren
  { male } { pluralchild } { sons }
```

`\femalechildren` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femalechildren
  { female } { pluralchild } { daughters }

\person_new_localisation:nnn
  { nonbinary } { pluralchild } { children }
```

`\maleparent` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \maleparent
  { male } { parent } { father }
```

`\femaleparent` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femaleparent
  { female } { parent } { mother }

\person_new_localisation:nnn
  { nonbinary } { parent } { parent }
```

`\pluralparent` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \pluralparent
  { unknown } { pluralparent } { parents }
```

`\malesibling` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \malesibling
  { male } { sibling } { brother }
```

`\femalesibling` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femalesibling
  { female } { sibling } { sister }

\person_new_localisation:nnn
  { nonbinary } { sibling } { sibling }
```

`\pluralsibling` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \pluralsibling
  { unknown } { pluralsibling } { siblings }
```

`\malesiblings` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \malesiblings
  { male } { pluralsibling } { brothers }
```

`\femalesiblings` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femalesiblings
  { female } { pluralsibling } { sisters }

\person_new_localisation:nnn
  { nonbinary } { pluralsibling } { siblings }
```

30.5.4 Other

`\andname` Version 3.0: removed definition of `\andname` (datatool-base now provides `\DTLandname`)

`\malename` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \malename
  { male } { gender } { male }
```

`\femalename` Old pre 3.0 command. This may be removed in future.

```
\__person_deprecated_lang_cs:Nnnn
 \femalename
  { female } { gender } { female }

\person_new_localisation:nnn
  { nonbinary } { gender } { non-binary }
\person_new_localisation:nnn
  { unknown } { gender } { unknown }
```

There are no plural versions of `gender`.

30.6 Displaying Information (Name and Title)

`\personsep` Separator to use between people (but not the between the last two).

```
\newcommand*{\personsep}{\DTLlistformatsep}
```

`\personlastsep` Separator to use between last two people if the list contains more than two people.

```
\newcommand*{\personlastsep}{
 \DTLlistformatoxford
 \DTLlistformatlastsep
}
```

`\twopeoplesep` Separator to use when list only contains two people.

```
\newcommand*{\twopeoplesep}{\DTLlistformatlastsep}
```

List of people. The provided function should encapsulate the person label.

```
\cs_new:Nn \person_display_list:N
{
  \int_case:nnF { \seq_count:N \l__person_people_seq }
   {
     { \c_zero_int }
      {
        \PackageWarning { person }
         { No ~ people ~ defined }
      }
     { \c_one_int }
      {
        \exp_args:Nx #1
         { \seq_item:Nn \l__person_people_seq { \c_one_int } }
      }
     { 2 }
      {
        \exp_args:Nx #1
         { \seq_item:Nn \l__person_people_seq { \c_one_int } }
        \twopeoplesep
        \exp_args:Nx #1
         { \seq_item:Nn \l__person_people_seq { 2 } }
      }
   }
   {
     \seq_map_indexed_inline:Nn \l__person_people_seq
      {
        \int_case:nnF { ##1 }
         {
           { \seq_count:N \l__person_people_seq }
             { \personlastsep }
           { \c_one_int } { }
         }
         { \personsep }
        #1 { ##2 }
```

```
      }
    }
}
```

`\personfullname` The person's full name can be displayed using `\personfullname[`⟨*label*⟩`]`, where ⟨*label*⟩ is the unique label used when defining that person. If ⟨*label*⟩ is omitted, `anon` is used.

```
\NewDocumentCommand \personfullname { O{anon} }
{
 \person_set_label:n { #1 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \__person_full_name:n { \l_person_label_tl }
  }
}

\cs_new:Nn \__person_full_name:n
{
  \tl_use:c { person@ #1 @fullname }
}
\cs_generate_variant:Nn \__person_full_name:n { x }
```

`\peoplefullname` List all defined people's full names.

```
\NewDocumentCommand \peoplefullname { }
{
  \person_display_list:N \__person_full_name:n
}
```

`\personforenames` The person's forenames can be displayed using `\personforenames[`⟨*label*⟩`]`, where ⟨*label*⟩ is the unique label used when defining that person. If ⟨*label*⟩ is omitted, `anon` is used.

```
\NewDocumentCommand \personforenames { O{anon} }
{
 \person_set_label:n { #1 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \__person_forenames:n { \l_person_label_tl }
  }
}

\cs_new:Nn \__person_forenames:n
{
  \tl_use:c { person@ #1 @forenames }
}
\cs_generate_variant:Nn \__person_forenames:n { x }
```

`\peopleforenames` List all defined people's full names.

```
\NewDocumentCommand \peopleforenames { }
{
```

```
  \person_display_list:N \__person_forenames:n
}
```

`\personname` As `\personfullname`, but for the person's familiar or first name.

```
\NewDocumentCommand \personname { O{anon} }
{
 \person_set_label:n { #1 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \__person_name:n { \l_person_label_tl }
  }
}

\cs_new:Nn \__person_name:n
{
  \tl_use:c { person@ #1 @name }
}
\cs_generate_variant:Nn \__person_name:n { x }
```

`\peoplename` List all defined people's familiar names.

```
\NewDocumentCommand \peoplename { }
{
  \person_display_list:N \__person_name:n
}
```

`\personsurname` As `\personforenames`, but for the person's surname.

```
\NewDocumentCommand \personsurname { O{anon} }
{
 \person_set_label:n { #1 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \__person_surname:n { \l_person_label_tl }
  }
}

\cs_new:Nn \__person_surname:n
{
  \tl_use:c { person@ #1 @surname }
}
\cs_generate_variant:Nn \__person_surname:n { x }
```

`\peoplesurname` List all defined people's surnames.

```
\NewDocumentCommand \peoplesurname { }
{
  \person_display_list:N \__person_surname:n
}
```

`\persontitlesurname` The person's title and surname or full name can be displayed using `\persontitlesurname [`⟨*label*⟩`]`, where ⟨*label*⟩ is the unique label used when defining that person. If ⟨*label*⟩ is omitted, `anon` is used.

```
\NewDocumentCommand \persontitlesurname { O{anon} }
{
 \person_set_label:n { #1 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \__person_title_surname:n { \l_person_label_tl }
  }
}

\cs_new:Nn \__person_title_surname:n
{
  \tl_if_empty:cTF { person@ #1 @surname }
  { \__person_full_name:n { #1 } }
  {
    \tl_if_empty:cTF { person@#1@title }
    { \__person_full_name:n { #1 } }
    {
     \tl_use:c { person@ #1 @title }
     \persontitlesurnamesep
     \tl_use:c { person@ #1 @surname}
    }
  }
}
```

\persontitlesurnamesep

```
\ExplSyntaxOff
\newcommand{\persontitlesurnamesep}{\ }
\ExplSyntaxOn
```

\peopletitlesurname List all defined people's title and surname or full names. This iterates through all labels in \@people@list.

```
\NewDocumentCommand \peopletitlesurname { }
{
  \person_display_list:N \__person_title_surname:n
}
```

30.7 Displaying Language-Sensitive Information

Display language text for a given person identified by their label.

```
\cs_new:Nn \__person_language:nn
{
  \person_get_localisation:vn
    { person@ #1 @gender } { #2 }
}
\cs_generate_variant:Nn \__person_language:nn { xn }
```

Sentence case:

```
\cs_new:Nn \__person_Language:nn
{
```

```
  \text_titlecase_first:n
   {
     \__person_language:nn { #1 } { #2 }
   }
}
\cs_generate_variant:Nn \__person_Language:nn { xn }
```

Common function for higher level user commands. No case-change:

```
\cs_new:Nn \person_language_text:nn
{
 \person_set_label:n { #1 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \__person_language:nn { \l_person_label_tl } { #2 }
  }
}
```

Sentence case:

```
\cs_new:Nn \person_Language_text:nn
{
 \person_set_label:n { #1 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \__person_Language:nn { \l_person_label_tl } { #2 }
  }
}
```

Plural, unless only one person defined. The argument is the language tag.

```
\cs_new:Nn \person_language_all_text:n
{
 \__person_case:nnn
  {
    \PackageWarning { person } { No ~ people ~ defined }
    \person_get_localisation:nn { unknown } { plural#1 }
  }
  {
    \exp_args:Nx \person_language_text:nn
      { \seq_item:Nn \l__person_people_seq { 1 } } { #1 }
  }
  {
    \person_all_gender_case:nnnn
     {
       \person_get_localisation:nn { male } { plural#1 }
     }
     {
       \person_get_localisation:nn { female } { plural#1 }
     }
     {
       \person_get_localisation:nn { nonbinary } { plural#1 }
     }
     {
```

```
      \person_get_localisation:nn { unknown } { plural#1 }
    }
  }
}
```

Sentence case:

```
\cs_new:Nn \person_Language_all_text:n
{
 \__person_case:nnn
  {
    \PackageWarning { person } { No ~ people ~ defined }
    \text_titlecase_first:n
     {
       \person_get_localisation:nn { unknown } { plural#1 }
     }
  }
  {
    \exp_args:Nx \person_Language_text:nn
      { \seq_item:Nn \l__person_people_seq { 1 } } { #1 }
  }
  {
    \person_all_gender_case:nnnn
     {
       \text_titlecase_first:n
        {
          \person_get_localisation:nn { male } { plural#1 }
        }
     }
     {
       \text_titlecase_first:n
        {
          \person_get_localisation:nn { female } { plural#1 }
        }
     }
     {
       \text_titlecase_first:n
        {
          \person_get_localisation:nn { nonbinary } { plural#1 }
        }
     }
     {
       \text_titlecase_first:n
        {
          \person_get_localisation:nn { unknown } { plural#1 }
        }
     }
  }
}
```

30.7.1 Third-Person Reference

`\personpronoun` Display the pronoun according to the person's gender.

```
\NewDocumentCommand \personpronoun { O{anon} }
{
  \person_language_text:nn { #1 } { pronoun }
}
```

`\Personpronoun` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personpronoun { O{anon} }
{
  \person_Language_text:nn { #1 } { pronoun }
}
```

`\peoplepronoun` Pronoun for all defined people.

```
\NewDocumentCommand \peoplepronoun { }
{
 \person_language_all_text:n { pronoun }
}
```

`\Peoplepronoun` As above, but first letter in upper case.

```
\NewDocumentCommand \Peoplepronoun { }
{
 \person_Language_all_text:n { pronoun }
}
```

`\personobjpronoun` Display the objective pronoun according to the person's gender.

```
\NewDocumentCommand \personobjpronoun { O{anon} }
{
  \person_language_text:nn { #1 } { objpronoun }
}
```

`\Personobjpronoun` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personobjpronoun { O{anon} }
{
  \person_Language_text:nn { #1 } { objpronoun }
}
```

`\peopleobjpronoun` Objective pronoun for all defined people.

```
\NewDocumentCommand \peopleobjpronoun { }
{
  \person_language_all_text:n { objpronoun }
}
```

`\Peopleobjpronoun` As above, but first letter in upper case

```
\NewDocumentCommand \Peopleobjpronoun { }
{
  \person_Language_all_text:n { objpronoun }
}
```

`\personpossadj` Display the possessive adjective according to the person's gender.

```
\NewDocumentCommand \personpossadj { O{anon} }
{
  \person_language_text:nn { #1 } { possadj }
}
```

`\Personpossadj` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personpossadj { O{anon} }
{
  \person_Language_text:nn { #1 } { possadj }
}
```

`\peoplepossadj` Possessive adjective for all defined people.

```
\NewDocumentCommand \peoplepossadj { }
{
 \person_language_all_text:n { possadj }
}
```

`\Peoplepossadj` As above, but first letter in upper case.

```
\NewDocumentCommand \Peoplepossadj { }
{
 \person_Language_all_text:n { possadj }
}
```

`\personposspronoun` Display possessive pronoun according to the person's gender.

```
\NewDocumentCommand \personposspronoun { O{anon} }
{
  \person_language_text:nn { #1 } { posspronoun }
}
```

`\Personposspronoun` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personposspronoun { O{anon} }
{
  \person_Language_text:nn { #1 } { posspronoun }
}
```

`\peopleposspronoun` Possessive pronoun for all defined people.

```
\NewDocumentCommand \peopleposspronoun { }
{
 \person_language_all_text:n { posspronoun }
}
```

`\Peopleposspronoun` As above, but first letter in upper case

```
\NewDocumentCommand \Peopleposspronoun { }
{
 \person_Language_all_text:n { posspronoun }
}
```

30.7.2 Second-Person Reference

`\personpronounii` Display the pronoun according to the person's gender.

```
\NewDocumentCommand \personpronounii { O{anon} }
{
  \person_language_text:nn { #1 } { pronoun2 }
}
```

`\Personpronounii` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personpronounii { O{anon} }
{
  \person_Language_text:nn { #1 } { pronoun2 }
}
```

`\peoplepronounii` Pronoun for all defined people.

```
\NewDocumentCommand \peoplepronounii { }
{
 \person_language_all_text:n { pronoun2 }
}
```

`\Peoplepronounii` As above, but first letter in upper case.

```
\NewDocumentCommand \Peoplepronounii { }
{
 \person_Language_all_text:n { pronoun2 }
}
```

`\personobjpronounii` Display the objective pronoun according to the person's gender.

```
\NewDocumentCommand \personobjpronounii { O{anon} }
{
  \person_language_text:nn { #1 } { objpronoun2 }
}
```

`\Personobjpronounii` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personobjpronounii { O{anon} }
{
  \person_Language_text:nn { #1 } { objpronoun2 }
}
```

`\peopleobjpronounii` Objective pronoun for all defined people.

```
\NewDocumentCommand \peopleobjpronounii { }
{
  \person_language_all_text:n { objpronoun2 }
}
```

`\Peopleobjpronounii` As above, but first letter in upper case

```
\NewDocumentCommand \Peopleobjpronounii { }
{
  \person_Language_all_text:n { objpronoun2 }
}
```

`\personpossadjii` Display the possessive adjective according to the person's gender.

```
\NewDocumentCommand \personpossadjii { O{anon} }
{
  \person_language_text:nn { #1 } { possadj2 }
}
```

`\Personpossadjii` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personpossadjii { O{anon} }
{
  \person_Language_text:nn { #1 } { possadj2 }
}
```

`\peoplepossadjii` Possessive adjective for all defined people.

```
\NewDocumentCommand \peoplepossadjii { }
{
 \person_language_all_text:n { possadj2 }
}
```

`\Peoplepossadjii` As above, but first letter in upper case.

```
\NewDocumentCommand \Peoplepossadjii { }
{
 \person_Language_all_text:n { possadj2 }
}
```

`\personposspronounii` Display possessive pronoun according to the person's gender.

```
\NewDocumentCommand \personposspronounii { O{anon} }
{
  \person_language_text:nn { #1 } { posspronoun2 }
}
```

`\Personposspronounii` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personposspronounii { O{anon} }
{
  \person_Language_text:nn { #1 } { posspronoun2 }
}
```

`\peopleposspronounii` Possessive pronoun for all defined people.

```
\NewDocumentCommand \peopleposspronounii { }
{
 \person_language_all_text:n { posspronoun2 }
}
```

`\Peopleposspronounii` As above, but first letter in upper case

```
\NewDocumentCommand \Peopleposspronounii { }
{
 \person_Language_all_text:n { posspronoun2 }
}
```

30.7.3 Relationship Reference

`\personchild` Display this person's relationship to their parent.

```
\NewDocumentCommand \personchild { O{anon} }
{
  \person_language_text:nn { #1 } { child }
}
```

`\Personchild` As above, but make first letter uppercase.

```
\NewDocumentCommand \Personchild { O{anon} }
{
  \person_Language_text:nn { #1 } { child }
}
```

`\peoplechild` Child relationship for all. (That is, all defined people have the same parents.)

```
\NewDocumentCommand \peoplechild { }
{
  \person_language_all_text:n { child }
}
```

`\Peoplechild` As above but first letter is made uppercase.

```
\NewDocumentCommand \Peoplechild { }
{
  \person_Language_all_text:n { child }
}
```

`\personparent` Display this person's relationship to their child (father / mother / parent).

```
\NewDocumentCommand \personparent { O{anon} }
{
  \person_language_text:nn { #1 } { parent }
}
```

`\Personparent` As above, but make the first letter uppercase.

```
\NewDocumentCommand \Personparent { O{anon} }
{
  \person_Language_text:nn { #1 } { parent }
}
```

`\peopleparent` The parental term for all defined people.

```
\NewDocumentCommand \peopleparent { }
{
 \person_language_all_text:n { parent }
}
```

`\Peopleparent` As above, but make first letter uppercase.

```
\NewDocumentCommand \Peopleparent { }
{
  \person_Language_all_text:n { parent }
}
```

`\personsibling` Display this person's relationship to their siblings.

```
\NewDocumentCommand \personsibling { O{anon} }
{
  \person_language_text:nn { #1 } { sibling }
}
```

`\Personsibling` As above but make first letter uppercase.

```
\NewDocumentCommand \Personsibling { O{anon} }
{
  \person_Language_text:nn { #1 } { sibling }
}
```

`\peoplesibling` Sibling term for all defined people.

```
\NewDocumentCommand \peoplesibling { }
{
  \person_language_all_text:n { sibling }
}
```

`\Peoplesibling`

```
\NewDocumentCommand \Peoplesibling { }
{
  \person_Language_all_text:n { sibling }
}
```

30.8 Extracting Information

`\persongender` Displays the given person's gender using localisation.

```
\NewDocumentCommand \persongender { m }
{
  \person_language_text:nn { #1 } { gender }
}
```

`\Persongender` As above but converts the first letter to uppercase.

```
\NewDocumentCommand \Persongender { m }
{
  \person_Language_text:nn { #1 } { gender }
}
```

`\getpersongender` Gets person's gender name (language-sensitive) and stores in first argument which must be a control sequence.

```
\NewDocumentCommand \getpersongender { m m }
{
 \person_set_label:n { #2 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \tl_set:Ne #1
     { \__person_language:nn { \l_person_label_tl } { gender } }
  }
}
```

`\getpersongenderlabel` Gets person's internal gender label and stores in first argument which must be a control sequence.

```
\NewDocumentCommand \getpersongenderlabel { m m }
{
 \person_set_label:n { #2 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \tl_set:Nx #1
     { person@ \l_person_label_tl @gender }
  }
}
```

`\getpersonname` Gets person's name and stores in first argument which must be a control sequence.

```
\NewDocumentCommand \getpersonname { m m }
{
 \person_set_label:n { #2 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \tl_set_eq:Nc #1
     { person@ \l_person_label_tl @name }
  }
}
```

`\getpersonforenames` Gets person's forenames and stores in first argument which must be a control sequence.

```
\NewDocumentCommand \getpersonforenames { m m }
{
 \person_set_label:n { #2 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \tl_set_eq:Nc #1
     { person@ \l_person_label_tl @forenames }
  }
}
```

`\getpersonfullname` Gets person's full name and stores in first argument which must be a control sequence.

```
\NewDocumentCommand \getpersonfullname { m m }
{
 \person_set_label:n { #2 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \tl_set_eq:Nc #1
     { person@ \l_person_label_tl @fullname }
  }
}
```

`\getpersonsurname` Gets person's surname and stores in first argument which must be a control sequence.

```
\NewDocumentCommand \getpersonsurname { m m }
```

```
{
 \person_set_label:n { #2 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \tl_set_eq:Nc #1
     { person@ \l_person_label_tl @surname }
  }
}
```

`\getpersontitle` Gets person's title and stores in first argument which must be a control sequence.

```
\NewDocumentCommand \getpersontitle { m m }
{
 \person_set_label:n { #2 }
 \person_if_exist_or_err:nT { \l_person_label_tl }
  {
    \tl_set_eq:Nc #1
     { person@ \l_person_label_tl @title }
  }
}

\ExplSyntaxOff
```

30.9 Localisation Support

`\RequirePersonDialect`

```
\newcommand*{\RequirePersonDialect}[1]{%
 \TrackLangRequireDialect{person}{#1}%
}

\datatool@load@locales{%
  \AnyTrackedLanguages
  {%
    \ForEachTrackedDialect{\@dtl@thisdialect}%
    {%
      \RequirePersonDialect{\@dtl@thisdialect}%
    }%
  }%
  {}%
}
```

30.10 Rollback v2.32 (databar-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{databar}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{xkeyval}
\RequirePackage{dataplot}[=v2.32]
\newif\ifDTLcolorbarchart
\DTLcolorbarcharttrue
\DeclareOption{color}{\DTLcolorbarcharttrue}
```

```
\DeclareOption{gray}{\DTLcolorbarchartfalse}
\newcommand*{\DTLbarXlabelalign}{left,rotate=-90}
\newcommand*{\DTLbarYticklabelalign}{right}
\define@boolkey{databar}[DTL]{verticalbars}[true]{%
\ifDTLverticalbars
 \def\DTLbarXlabelalign{left,rotate=-90}%
 \def\DTLbarYticklabelalign{right}
\else
 \def\DTLbarXlabelalign{right}%
 \def\DTLbarYticklabelalign{center}
\fi}
\DTLverticalbarstrue
\DeclareOption{vertical}{\DTLverticalbarstrue
 \def\DTLbarXlabelalign{left,rotate=-90}%
 \def\DTLbarYticklabelalign{right}
}
\DeclareOption{horizontal}{\DTLverticalbarsfalse
 \def\DTLbarXlabelalign{right}%
 \def\DTLbarYticklabelalign{center}
}
\ProcessOptions
\RequirePackage{datatool}[=v2.32]
\RequirePackage{tikz}
\newlength\DTLbarchartlength
\DTLbarchartlength=3in
\newlength\DTLbarwidth
\DTLbarwidth=1cm
\newlength\DTLbarlabeloffset
\setlength\DTLbarlabeloffset{10pt}
\newcommand*{\DTLBarXAxisStyle}{-}
\newcommand*{\DTLBarYAxisStyle}{-}
\newcounter{DTLbarroundvar}
\setcounter{DTLbarroundvar}{1}
\newcommand*{\DTLbardisplayYticklabel}[1]{#1}
\newcommand*{\DTLdisplaylowerbarlabel}[1]{#1}
\newcommand*{\DTLdisplaylowermultibarlabel}[1]{#1}
\newcommand*{\DTLdisplayupperbarlabel}[1]{#1}
\newcommand*{\DTLdisplayuppermultibarlabel}[1]{#1}
\newcommand*{\DTLbaratbegintikz}{}
\newcommand*{\DTLbaratendtikz}{}
\newif\ifDTLbarxaxis
\newif\ifDTLbaryaxis
\newif\ifDTLbarytics
\newcount\@dtl@barcount
\newcommand*{\DTLsetbarcolor}[2]{%
\expandafter\def\csname dtlbar@segcol\romannumeral#1\endcsname{#2}%
}
\newcommand*{\DTLgetbarcolor}[1]{%
\csname dtlbar@segcol\romannumeral#1\endcsname}
\newcommand*{\DTLdobarcolor}[1]{%
```

```
\expandafter\color\expandafter
{\csname dtlbar@segcol\romannumeral#1\endcsname}}
\newcommand*{\DTLdocurrentbarcolor}{%
\ifnum\dtlforeachlevel=0\relax
  \PackageError{databar}{Can't use
  \string\DTLdocurrentbarcolor\space outside
  \string\DTLbarchart}{}%
\else
  \expandafter\DTLdobarcolor\expandafter{%
  \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname}%
\fi}
\newcommand*{\DTLbaroutlinecolor}{black}
\newlength\DTLbaroutlinewidth
\DTLbaroutlinewidth=0pt
\ifDTLcolorbarchart
 \DTLsetbarcolor{1}{red}
 \DTLsetbarcolor{2}{green}
 \DTLsetbarcolor{3}{blue}
 \DTLsetbarcolor{4}{yellow}
 \DTLsetbarcolor{5}{magenta}
 \DTLsetbarcolor{6}{cyan}
 \DTLsetbarcolor{7}{orange}
 \DTLsetbarcolor{8}{white}
\else
 \DTLsetbarcolor{1}{black!15}
 \DTLsetbarcolor{2}{black!25}
 \DTLsetbarcolor{3}{black!35}
 \DTLsetbarcolor{4}{black!45}
 \DTLsetbarcolor{5}{black!55}
 \DTLsetbarcolor{6}{black!65}
 \DTLsetbarcolor{7}{black!75}
 \DTLsetbarcolor{8}{black!85}
\fi
\newcommand*{\DTLeverybarhook}{}
\define@key{databar}{max}{\def\DTLbarmax{#1}}
\define@key{databar}{length}{\DTLbarchartlength=#1\relax
}
\define@key{databar}{maxdepth}{%
\ifnum#1>0\relax
 \PackageError{databar}{depth must be zero or negative}{}%
\else
 \def\DTLnegextent{#1}%
\fi}
\define@choicekey{databar}{axes}[\var\nr]{both,x,y,none}{%
\ifcase\nr\relax
 % both
  \DTLbarxaxistrue
  \DTLbaryaxistrue
  \DTLbaryticstrue
\or
```

```
 % x only
  \DTLbarxaxistrue
  \DTLbaryaxisfalse
  \DTLbaryticsfalse
\or
 % y only
  \DTLbarxaxisfalse
  \DTLbaryaxistrue
  \DTLbaryticstrue
\or
 % neither
  \DTLbarxaxisfalse
  \DTLbaryaxisfalse
  \DTLbaryticsfalse
\fi
}
\define@key{databar}{variable}{%
 \def\DTLbarvariable{#1}%
}
\define@key{databar}{variables}{%
 \def\dtlbar@variables{#1}%
}
\define@key{databar}{barwidth}{\setlength{\DTLbarwidth}{#1}}
\define@key{databar}{barlabel}{%
\def\dtl@barlabel{#1}}
\def\dtl@barlabel{}
\define@key{databar}{multibarlabels}{%
\def\dtl@multibarlabels{#1}}
\def\dtl@multibarlabels{}
\define@key{databar}{groupgap}{\def\dtlbar@groupgap{#1}}
\def\dtlbar@groupgap{1}
\define@key{databar}{upperbarlabel}{%
\def\dtl@upperbarlabel{#1}}
\def\dtl@upperbarlabel{}
\define@key{databar}{uppermultibarlabels}{%
\def\dtl@uppermultibarlabels{#1}}
\def\dtl@uppermultibarlabels{}
\define@key{databar}{yticpoints}{%
\def\dtlbar@yticlist{#1}\DTLbaryticstrue\DTLbaryaxistrue}
\let\dtlbar@yticlist=\relax
\define@key{databar}{yticgap}{%
\def\dtlbar@yticgap{#1}\DTLbaryticstrue\DTLbaryaxistrue}
\let\dtlbar@yticgap=\relax
\define@key{databar}{yticlabels}{%
\def\dtlbar@yticlabels{#1}\DTLbaryticstrue\DTLbaryaxistrue}
\let\dtlbar@yticlabels=\relax
\define@key{databar}{ylabel}{%
\def\dtlbar@ylabel{#1}}
\let\dtlbar@ylabel=\relax
\newcommand*{\DTLbarchart}[4][\boolean{true}]{%
```

```
{%
  \undef\DTLbarvariable
  \undef\DTLbarmax
  \undef\DTLnegextent
  \disable@keys{databar}{variables,multibarlabels,%
    uppermultibarlabels,groupgap}%
  \setkeys{databar}{#2}%
  \ifundef{\DTLbarvariable}%
  {%
    \PackageError{databar}%
    {\string\DTLbarchart\space missing variable}%
    {You haven't use the "variable" key}%
  }%
  {%
    \ifundef{\DTLbarmax}%
    {%
      \@sDTLforeach[#1]{#3}{#4}{%
        \expandafter\DTLconverttodecimal\expandafter
          {\DTLbarvariable}{\dtl@barvar}%
        \ifundef{\DTLbarmax}%
        {%
          \let\DTLbarmax=\dtl@barvar
        }%
        {%
          \let\dtl@old=\DTLbarmax
          \dtlmax{\DTLbarmax}{\dtl@old}{\dtl@barvar}%
        }%
      }%
      \ifx\dtlbar@yticgap\relax
      \else
        \let\dtl@thistick=\dtlbar@yticgap
        \whiledo{\DTLisFPopenbetween{\dtl@thistick}{0}{\DTLbarmax}}%
        {%
          \dtladd{\dtl@thistick}{\dtl@thistick}{\dtlbar@yticgap}%
        }%
        \let\DTLbarmax=\dtl@thistick
      \fi
    }%
    {}%
    \ifundef{\DTLnegextent}%
    {%
      \def\DTLnegextent{0}%
      \@sDTLforeach[#1]{#3}{#4}{%
        \expandafter\DTLconverttodecimal\expandafter
          {\DTLbarvariable}{\dtl@barvar}%
        \let\dtl@old=\DTLnegextent
        \DTLmin{\DTLnegextent}{\dtl@old}{\dtl@barvar}%
      }%
      \ifx\dtlbar@yticgap\relax
      \else
```

```
    \ifthenelse{\DTLisFPlt{\DTLnegextent}{0}}%
    {%
    \edef\dtl@thistick{0}%
    \whiledo{\DTLisFPclosedbetween{\dtl@thistick}{\DTLnegextent}{0}}{%
      \dtlsub{\dtl@thistick}{\dtl@thistick}{\dtlbar@yticgap}%
    }%
    \let\DTLnegextent=\dtl@thistick
    }{}%
  \fi
}%
{}%
\@dtl@tmpcount=\DTLbarchartlength
\dtlsub{\dtl@extent}{\DTLbarmax}{\DTLnegextent}%
\dtldiv{\dtl@unit}{\number\@dtl@tmpcount}{\dtl@extent}%
 \setlength{\dtl@yticlabelwidth}{0pt}%
 \ifDTLbarytics
   \ifx\dtlbar@yticlist\relax
     \ifx\dtlbar@yticgap\relax
       \@dtl@tmpcount=\DTLmintickgap
       \divide\@dtl@tmpcount by 65536\relax
       \dtldiv{\dtl@mingap}{\number\@dtl@tmpcount}{\dtl@unit}%
       \dtl@constructticklist\DTLnegextent\DTLbarmax
         \dtl@mingap\dtlbar@yticlist
     \else
       \dtl@constructticklistwithgapz
         \DTLnegextent\DTLbarmax\dtlbar@yticlist\dtlbar@yticgap
     \fi
   \fi
   \ifx\dtlbar@ylabel\relax
   \else
     \ifx\dtlbar@yticlabels\relax
       \@for\dtl@thislabel:=\dtlbar@yticlist\do{%
         \dtlround{\dtl@thislabel}{\dtl@thislabel}
                  {\c@DTLbarroundvar}%
         \ifDTLverticalbars
           \settowidth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
         \else
           \settoheight{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \edef\@dtl@h{\the\dtl@tmplength}%
           \settodepth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \addtolength{\dtl@tmplength}{\@dtl@h}%
           \addtolength{\dtl@tmplength}{\baselineskip}%
         \fi
         \ifdim\dtl@tmplength>\dtl@yticlabelwidth
           \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
         \fi
       }%
```

```
      \else
        \@for\dtl@thislabel:=\dtlbar@yticlabels\do{%
          \ifDTLverticalbars
            \settowidth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
          \else
            \settoheight{\dtl@tmplength}{%
               \DTLbardisplayYticklabel{\dtl@thislabel}}%
            \edef\@dtl@h{\the\dtl@tmplength}%
            \settodepth{\dtl@tmplength}{%
               \DTLbardisplayYticklabel{\dtl@thislabel}}%
            \addtolength{\dtl@tmplength}{\@dtl@h}%
            \addtolength{\dtl@tmplength}{\baselineskip}%
          \fi
          \ifdim\dtl@tmplength>\dtl@yticlabelwidth
            \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
          \fi
        }%
      \fi
    \fi
  \fi
\edef\DTLbarchartwidth{\expandafter\number\csname dtlrows@#3\endcsname}
\begin{tikzpicture}
\ifDTLverticalbars
  \pgfsetyvec{\pgfpoint{0pt}{\dtl@unit sp}}%
  \pgfsetxvec{\pgfpoint{\DTLbarwidth}{0pt}}%
\else
  \pgfsetxvec{\pgfpoint{\dtl@unit sp}{0pt}}%
  \pgfsetyvec{\pgfpoint{0pt}{\DTLbarwidth}}%
\fi
\DTLbaratbegintikz
\def\@dtl@start{0}%
\@sDTLforeach[#1]{#3}{#4}{%
\expandafter\let\expandafter\@dtl@bar
  \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname%
\expandafter\DTLconverttodecimal\expandafter
  {\DTLbarvariable}{\dtl@variable}%
\begin{scope}
 \DTLdocurrentbarcolor
 \ifDTLverticalbars
   \fill (\@dtl@start,0) -- (\@dtl@start,\dtl@variable)
      -- (\@dtl@bar,\dtl@variable) -- (\@dtl@bar,0) -- cycle;
 \else
   \fill (0,\@dtl@start) -- (\dtl@variable,\@dtl@start)
      -- (\dtl@variable,\@dtl@bar) -- (0,\@dtl@bar) -- cycle;
 \fi
\end{scope}
\begin{scope}
\ifdim\DTLbaroutlinewidth>0pt
 \expandafter\color\expandafter{\DTLbaroutlinecolor}
```

```
 \ifDTLverticalbars
   \draw (\@dtl@start,0) -- (\@dtl@start,\dtl@variable)
      -- (\@dtl@bar,\dtl@variable) -- (\@dtl@bar,0) -- cycle;
 \else
   \draw (0,\@dtl@start) -- (\dtl@variable,\@dtl@start)
      -- (\dtl@variable,\@dtl@bar) -- (0,\@dtl@bar) -- cycle;
 \fi
\fi
\end{scope}
\ifDTLverticalbars
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\@dtl@start.5}{0}}
           {\noexpand\pgfpoint{0pt}{-\noexpand\DTLbarlabeloffset}}},
     \DTLbarXlabelalign
  }%
  \edef\DTLstartpt{\noexpand\pgfpointxy{\@dtl@start.5}{0}}%
\else
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{0}{\@dtl@start.5}}
           {\noexpand\pgfpoint{-\noexpand\DTLbarlabeloffset}{0pt}}},
     \DTLbarXlabelalign
  }%
  \edef\DTLstartpt{\noexpand\pgfpointxy{0}{\@dtl@start.5}}%
\fi
 \expandafter\pgftext\expandafter[\dtl@textopt]{%
   \DTLdisplaylowerbarlabel{\dtl@barlabel}}
\ifDTLverticalbars
  \expandafter\DTLifnumlt\expandafter{\DTLbarvariable}{0}%
  {
    \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\@dtl@start.5}{\dtl@variable}}
           {\noexpand\pgfpoint{0pt}{-\noexpand\DTLbarlabeloffset}}}
    }%
  }{%
    \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\@dtl@start.5}{\dtl@variable}}
           {\noexpand\pgfpoint{0pt}{\noexpand\DTLbarlabeloffset}}}
    }%
  }
  \edef\DTLendpt{\noexpand\pgfpointxy{\@dtl@start.5}{\dtl@variable}}%
\else
  \expandafter\DTLifnumlt\expandafter{\DTLbarvariable}{0}%
  {
    \edef\dtl@textopt{right,
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\dtl@variable}{\@dtl@start.5}}
```

```
          {\noexpand\pgfpoint{-\noexpand\DTLbarlabeloffset}{0pt}}}
    }%
  }{%
    \edef\dtl@textopt{left,
      at={\noexpand\pgfpointadd
          {\noexpand\pgfpointxy{\dtl@variable}{\@dtl@start.5}}
          {\noexpand\pgfpoint{\noexpand\DTLbarlabeloffset}{0pt}}}
    }%
  }
  \edef\DTLendpt{\noexpand\pgfpointxy{\dtl@variable}{\@dtl@start.5}}%
\fi
 \expandafter\pgftext\expandafter[\dtl@textopt]{%
    \DTLdisplayupperbarlabel{\dtl@upperbarlabel}}
\def\DTLmidpt{\pgfpointlineattime{0.5}{\DTLstartpt}{\DTLendpt}}%
  \DTLeverybarhook
  \edef\@dtl@start{\number\@dtl@bar}%
}%
\ifDTLbarxaxis
  \ifDTLverticalbars
    \expandafter\draw\expandafter[\DTLBarXAxisStyle]
      (0,0) -- (\DTLbarchartwidth,0);
  \else
    \expandafter\draw\expandafter[\DTLBarXAxisStyle]
      (0,0) -- (0,\DTLbarchartwidth);
  \fi
\fi
\ifDTLbaryaxis
  \ifDTLverticalbars
    \expandafter\draw\expandafter[\DTLBarYAxisStyle]
      (0,\DTLnegextent) -- (0,\DTLbarmax);
  \else
    \expandafter\draw\expandafter[\DTLBarYAxisStyle]
      (\DTLnegextent,0) -- (\DTLbarmax,0);
  \fi
\fi
\ifx\dtlbar@yticlist\relax
\else
  \@for\dtl@thistick:=\dtlbar@yticlist\do{%
    \ifDTLverticalbars
      \pgfpathmoveto{\pgfpointxy{0}{\dtl@thistick}}
      \pgfpathlineto{
        \pgfpointadd{\pgfpointxy{0}{\dtl@thistick}}
                    {\pgfpoint{-\DTLticklength}{0pt}}}
    \else
      \pgfpathmoveto{\pgfpointxy{\dtl@thistick}{0}}
      \pgfpathlineto{
        \pgfpointadd{\pgfpointxy{\dtl@thistick}{0}}
                    {\pgfpoint{0pt}{-\DTLticklength}}}
    \fi
    \pgfusepath{stroke}
```

```
      \ifx\dtlbar@yticlabels\relax
         \dtlround{\dtl@thislabel}{\dtl@thistick}
                  {\c@DTLbarroundvar}%
      \else
         \dtl@chopfirst\dtlbar@yticlabels\dtl@thislabel\dtl@rest
         \let\dtlbar@yticlabels=\dtl@rest
      \fi
      \ifDTLverticalbars
        \edef\dtl@textopt{\DTLbarYticklabelalign,%
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{0}{\dtl@thistick}}
               {\noexpand\pgfpoint{-\noexpand\DTLticklabeloffset}{0pt}},
          }}%
      \else
        \edef\dtl@textopt{\DTLbarYticklabelalign,
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{\dtl@thistick}{0}}
               {\noexpand\pgfpoint{0pt}{-\noexpand\DTLticklabeloffset}}
          }}%
      \fi
      \expandafter\pgftext\expandafter[\dtl@textopt]{%
        \DTLbardisplayYticklabel{\dtl@thislabel}}
    }%
  \fi
  \ifx\dtlbar@ylabel\relax
  \else
    \addtolength{\dtl@yticlabelwidth}{\baselineskip}%
    \setlength{\dtl@tmplength}{0.5\DTLbarchartlength}
    \ifDTLverticalbars
      \pgftext[bottom,center,at={\pgfpointadd
          {\pgfpointxy{0}{\DTLnegextent}}%
          {\pgfpoint{-\dtl@yticlabelwidth}{\dtl@tmplength}}},
          rotate=90]{%
         \dtlbar@ylabel}
    \else
      \pgftext[bottom,center,at={\pgfpointadd
          {\pgfpointxy{\DTLnegextent}{0}}%
          {\pgfpoint{\dtl@tmplength}{-\dtl@yticlabelwidth}}}]{%
         \dtlbar@ylabel}
    \fi
  \fi
  \DTLbaratendtikz
  \end{tikzpicture}
  }%
}%
}
\newcommand*{\DTLmultibarchart}[4][\boolean{true}]{%
{\let\dtlbar@variables=\relax
\let\DTLbarmax=\relax
\let\DTLnegextent=\relax
```

```
\disable@keys{databar}{variable,upperbarlabel}%
\setkeys{databar}{#2}%
\ifx\dtlbar@variables\relax
  \PackageError{databar}{\string\DTLmultibarchart\space missing variables setting}{}%
\else
  \ifx\DTLbarmax\relax
    \@sDTLforeach[#1]{#3}{#4}{%
      \@for\DTLbarvariable:=\dtlbar@variables\do{%
        \expandafter\DTLconverttodecimal\expandafter
          {\DTLbarvariable}{\dtl@barvar}%
        \ifx\DTLbarmax\relax
          \let\DTLbarmax=\dtl@barvar
        \else
          \let\dtl@old=\DTLbarmax
          \dtlmax{\DTLbarmax}{\dtl@old}{\dtl@barvar}%
        \fi
      }%
    }%
    \ifx\dtlbar@yticgap\relax
    \else
      \let\dtl@thistick=\dtlbar@yticgap%
      \whiledo{\DTLisFPopenbetween{\dtl@thistick}{0}{\DTLbarmax}}{%
        \dtladd{\dtl@thistick}{\dtl@thistick}{\dtlbar@yticgap}%
      }%
      \let\DTLbarmax=\dtl@thistick
    \fi
  \fi
  \ifx\DTLnegextent\relax
    \def\DTLnegextent{0}%
    \@sDTLforeach[#1]{#3}{#4}{%
      \@for\DTLbarvariable:=\dtlbar@variables\do{%
        \expandafter\DTLconverttodecimal\expandafter
          {\DTLbarvariable}{\dtl@barvar}%
        \let\dtl@old=\DTLnegextent
        \DTLmin{\DTLnegextent}{\dtl@old}{\dtl@barvar}%
      }%
    }%
    \ifx\dtlbar@yticgap\relax
    \else
      \ifthenelse{\DTLisFPlt{\DTLnegextent}{0}}{%
      \edef\dtl@thistick{0}%
      \whiledo{\DTLisFPclosedbetween{\dtl@thistick}{\DTLnegextent}{0}}{%
        \dtlsub{\dtl@thistick}{\dtl@thistick}{\dtlbar@yticgap}%
      }%
      \let\DTLnegextent=\dtl@thistick
      }{}%
    \fi
  \fi
  \@dtl@tmpcount=\DTLbarchartlength
  \dtlsub{\dtl@extent}{\DTLbarmax}{\DTLnegextent}%
```

```
\dtldiv{\dtl@unit}{\number\@dtl@tmpcount}{\dtl@extent}%
 \setlength{\dtl@yticlabelwidth}{0pt}%
 \ifDTLbarytics
   \ifx\dtlbar@yticlist\relax
     \ifx\dtlbar@yticgap\relax
       \@dtl@tmpcount=\DTLmintickgap
       \divide\@dtl@tmpcount by 65536\relax
       \dtldiv{\dtl@mingap}{\number\@dtl@tmpcount}{\dtl@unit}%
       \dtl@constructticklist\DTLnegextent\DTLbarmax
         \dtl@mingap\dtlbar@yticlist
     \else
       \dtl@constructticklistwithgapz
         \DTLnegextent\DTLbarmax\dtlbar@yticlist\dtlbar@yticgap
     \fi
   \fi
   \ifx\dtlbar@ylabel\relax
   \else
     \ifx\dtlbar@yticlabels\relax
       \@for\dtl@thislabel:=\dtlbar@yticlist\do{%
         \dtlround{\dtl@thislabel}{\dtl@thislabel}
                  {\c@DTLbarroundvar}%
         \ifDTLverticalbars
           \settowidth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
         \else
           \settoheight{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \edef\@dtl@h{\the\dtl@tmplength}%
           \settodepth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \addtolength{\dtl@tmplength}{\@dtl@h}%
           \addtolength{\dtl@tmplength}{\baselineskip}%
         \fi
         \ifdim\dtl@tmplength>\dtl@yticlabelwidth
           \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
         \fi
       }%
     \else
       \@for\dtl@thislabel:=\dtlbar@yticlabels\do{%
         \ifDTLverticalbars
           \settowidth{\dtl@tmplength}{%
             \DTLbardisplayYticklabel{\dtl@thislabel}}%
         \else
           \settoheight{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \edef\@dtl@h{\the\dtl@tmplength}%
           \settodepth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \addtolength{\dtl@tmplength}{\@dtl@h}%
           \addtolength{\dtl@tmplength}{\baselineskip}%
```

```
          \fi
          \ifdim\dtl@tmplength>\dtl@yticlabelwidth
            \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
          \fi
        }%
      \fi
    \fi
  \fi
\dtl@xticlabelheight=0pt\relax
\@dtl@tmpcount=0\relax
\@for\dtl@thislabel:=\dtl@multibarlabels\do{%
  \advance\@dtl@tmpcount by 1\relax
  \settoheight{\dtl@tmplength}{\tikz\expandafter\pgftext\expandafter
    [\DTLbarXlabelalign]{\DTLdisplaylowerbarlabel{\dtl@thislabel}};}%
  \edef\@dtl@h{\the\dtl@tmplength}%
  \settodepth{\dtl@tmplength}{\tikz\expandafter\pgftext\expandafter
    [\DTLbarXlabelalign]{\DTLdisplaylowerbarlabel{\dtl@thislabel}};}%
  \addtolength{\dtl@tmplength}{\@dtl@h}%
  \addtolength{\dtl@tmplength}{\baselineskip}%
  \ifdim\dtl@tmplength>\dtl@xticlabelheight
    \setlength{\dtl@xticlabelheight}{\dtl@tmplength}%
  \fi
}
\@dtl@tmpcount=0\relax
\@for\dtl@this:=\dtlbar@variables\do{%
  \advance\@dtl@tmpcount by 1\relax
}%
\edef\DTLbargroupwidth{\number\@dtl@tmpcount}%
\edef\dtl@n{\expandafter\number\csname dtlrows@#3\endcsname}
\dtlmul{\dtl@tmpi}{\dtl@n}{\DTLbargroupwidth}
\dtlsub{\dtl@tmpii}{\dtl@n}{1}%
\dtlmul{\dtl@tmpii}{\dtl@tmpii}{\dtlbar@groupgap}%
\dtladd{\DTLbarchartwidth}{\dtl@tmpi}{\dtl@tmpii}
\begin{tikzpicture}
\ifDTLverticalbars
  \pgfsetyvec{\pgfpoint{0pt}{\dtl@unit sp}}%
  \pgfsetxvec{\pgfpoint{\DTLbarwidth}{0pt}}%
\else
  \pgfsetxvec{\pgfpoint{\dtl@unit sp}{0pt}}%
  \pgfsetyvec{\pgfpoint{0pt}{\DTLbarwidth}}%
\fi
\DTLbaratbegintikz
\def\@dtl@start{0}%
\@sDTLforeach[#1]{#3}{#4}{%
\@dtl@barcount = 1\relax
\let\dtl@multibar@labels=\dtl@multibarlabels
\let\dtl@uppermultibar@labels=\dtl@uppermultibarlabels
\dtlmul{\dtl@multimidpt}{\DTLbargroupwidth}{0.5}%
\dtladd{\dtl@multimidpt}{\dtl@multimidpt}{\@dtl@start}%
\@for\DTLbarvariable:=\dtlbar@variables\do{%
```

```
\dtladd{\@dtl@endpt}{\@dtl@start}{1}%
\expandafter\DTLconverttodecimal\expandafter
  {\DTLbarvariable}{\dtl@variable}%
\dtl@chopfirst\dtl@multibar@labels\dtl@thisbarlabel\dtl@rest
\let\dtl@multibar@labels=\dtl@rest
\dtl@chopfirst\dtl@uppermultibar@labels\dtl@thisupperbarlabel\dtl@rest
\let\dtl@uppermultibar@labels=\dtl@rest
\begin{scope}
 \expandafter\color\expandafter{\DTLgetbarcolor{\@dtl@barcount}}%
 \ifDTLverticalbars
   \fill (\@dtl@start,0) -- (\@dtl@start,\dtl@variable)
      -- (\@dtl@endpt,\dtl@variable) -- (\@dtl@endpt,0) -- cycle;
 \else
   \fill (0,\@dtl@start) -- (\dtl@variable,\@dtl@start)
      -- (\dtl@variable,\@dtl@endpt) -- (0,\@dtl@endpt) -- cycle;
 \fi
\end{scope}
\begin{scope}
\ifdim\DTLbaroutlinewidth>0pt
 \expandafter\color\expandafter{\DTLbaroutlinecolor}
 \ifDTLverticalbars
   \draw (\@dtl@start,0) -- (\@dtl@start,\dtl@variable)
      -- (\@dtl@endpt,\dtl@variable) -- (\@dtl@endpt,0) -- cycle;
 \else
   \draw (0,\@dtl@start) -- (\dtl@variable,\@dtl@start)
      -- (\dtl@variable,\@dtl@endpt) -- (0,\@dtl@endpt) -- cycle;
 \fi
\fi
\end{scope}
\dtladd{\@dtl@midpt}{\@dtl@start}{0.5}%
\ifDTLverticalbars
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\@dtl@midpt}{0}}
           {\noexpand\pgfpoint{0pt}{-\noexpand\DTLbarlabeloffset}}},
     \DTLbarXlabelalign
  }%
  \edef\DTLstartpt{\noexpand\pgfpointxy{\@dtl@midpt}{0}}%
\else
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{0}{\@dtl@midpt}}
           {\noexpand\pgfpoint{-\noexpand\DTLbarlabeloffset}{0pt}}},
     \DTLbarXlabelalign
  }%
  \edef\DTLstartpt{\noexpand\pgfpointxy{0}{\@dtl@midpt}}%
\fi
 \expandafter\pgftext\expandafter[\dtl@textopt]{%
   \DTLdisplaylowermultibarlabel{\dtl@thisbarlabel}}
\ifDTLverticalbars
```

```
  \expandafter\DTLifnumlt\expandafter{\DTLbarvariable}{0}
  {
    \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\@dtl@midpt}{\dtl@variable}}
           {\noexpand\pgfpoint{0pt}{-\noexpand\DTLbarlabeloffset}}}
    }%
  }{%
    \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\@dtl@midpt}{\dtl@variable}}
           {\noexpand\pgfpoint{0pt}{\noexpand\DTLbarlabeloffset}}}
    }%
  }
  \edef\DTLendpt{\noexpand\pgfpointxy{\@dtl@midpt}{\dtl@variable}}%
\else
  \expandafter\DTLifnumlt\expandafter{\DTLbarvariable}{0}
  {
    \edef\dtl@textopt{right,
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\dtl@variable}{\@dtl@midpt}}
           {\noexpand\pgfpoint{-\noexpand\DTLbarlabeloffset}{0pt}}}
    }%
  }{%
    \edef\dtl@textopt{left,
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\dtl@variable}{\@dtl@midpt}}
           {\noexpand\pgfpoint{\noexpand\DTLbarlabeloffset}{0pt}}}
    }%
  }
  \edef\DTLendpt{\noexpand\pgfpointxy{\dtl@variable}{\@dtl@midpt}}%
\fi
 \expandafter\pgftext\expandafter[\dtl@textopt]{%
    \DTLdisplayuppermultibarlabel{\dtl@thisupperbarlabel}}
\def\DTLmidpt{\pgfpointlineattime{0.5}{\DTLstartpt}{\DTLendpt}}%
\DTLeverybarhook
  \dtladd{\@dtl@start}{\@dtl@start}{1}%
  \advance\@dtl@barcount by 1\relax
}%
\setlength{\dtl@tmplength}{\DTLbarlabeloffset}%
\addtolength{\dtl@tmplength}{\dtl@xticlabelheight}%
\ifDTLverticalbars
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\dtl@multimidpt}{0}}
           {\noexpand\pgfpoint{0pt}{-\noexpand\dtl@tmplength}}},
     \DTLbarXlabelalign
  }%
\else
  \edef\dtl@textopt{%
```

```
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{0}{\dtl@multimidpt}}
           {\noexpand\pgfpoint{-\noexpand\dtl@tmplength}{0pt}}},
     \DTLbarXlabelalign
  }%
\fi
 \expandafter\pgftext\expandafter[\dtl@textopt]{%
   \DTLdisplaylowerbarlabel{\dtl@barlabel}}
  \dtladd{\@dtl@start}{\@dtl@start}{\dtlbar@groupgap}%
}
\ifDTLbarxaxis
  \ifDTLverticalbars
    \expandafter\draw\expandafter[\DTLBarXAxisStyle]
      (0,0) -- (\DTLbarchartwidth,0);
  \else
    \expandafter\draw\expandafter[\DTLBarXAxisStyle]
      (0,0) -- (0,\DTLbarchartwidth);
  \fi
\fi
\ifDTLbaryaxis
  \ifDTLverticalbars
    \expandafter\draw\expandafter[\DTLBarYAxisStyle]
      (0,\DTLnegextent) -- (0,\DTLbarmax);
  \else
    \expandafter\draw\expandafter[\DTLBarYAxisStyle]
      (\DTLnegextent,0) -- (\DTLbarmax,0);
  \fi
\fi
\ifx\dtlbar@yticlist\relax
\else
  \@for\dtl@thistick:=\dtlbar@yticlist\do{%
    \ifDTLverticalbars
      \pgfpathmoveto{\pgfpointxy{0}{\dtl@thistick}}
      \pgfpathlineto{
        \pgfpointadd{\pgfpointxy{0}{\dtl@thistick}}
                    {\pgfpoint{-\DTLticklength}{0pt}}}
    \else
      \pgfpathmoveto{\pgfpointxy{\dtl@thistick}{0}}
      \pgfpathlineto{
        \pgfpointadd{\pgfpointxy{\dtl@thistick}{0}}
                    {\pgfpoint{0pt}{-\DTLticklength}}}
    \fi
    \pgfusepath{stroke}
    \ifx\dtlbar@yticlabels\relax
       \dtlround{\dtl@thislabel}{\dtl@thistick}
                {\c@DTLbarroundvar}%
    \else
       \dtl@chopfirst\dtlbar@yticlabels\dtl@thislabel\dtl@rest
       \let\dtlbar@yticlabels=\dtl@rest
    \fi
```

```
    \ifDTLverticalbars
      \edef\dtl@textopt{\DTLbarYticklabelalign,%
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{0}{\dtl@thistick}}
             {\noexpand\pgfpoint{-\noexpand\DTLticklabeloffset}{0pt}},
        }}%
    \else
      \edef\dtl@textopt{\DTLbarYticklabelalign,
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{\dtl@thistick}{0}}
             {\noexpand\pgfpoint{0pt}{-\noexpand\DTLticklabeloffset}}
        }}%
    \fi
    \expandafter\pgftext\expandafter[\dtl@textopt]{%
      \DTLbardisplayYticklabel{\dtl@thislabel}}
  }%
\fi
\ifx\dtlbar@ylabel\relax
\else
  \addtolength{\dtl@yticlabelwidth}{\baselineskip}%
  \setlength{\dtl@tmplength}{0.5\DTLbarchartlength}
  \ifDTLverticalbars
    \pgftext[bottom,center,at={\pgfpointadd
        {\pgfpointxy{0}{\DTLnegextent}}%
        {\pgfpoint{-\dtl@yticlabelwidth}{\dtl@tmplength}}},
        rotate=90]{%
       \dtlbar@ylabel}
  \else
    \pgftext[bottom,center,at={\pgfpointadd
        {\pgfpointxy{\DTLnegextent}{0}}%
        {\pgfpoint{\dtl@tmplength}{-\dtl@yticlabelwidth}}}]{%
       \dtlbar@ylabel}
  \fi
\fi
\DTLbaratendtikz
\end{tikzpicture}
\fi
}}
```

30.11 Rollback v2.32 (databib-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{databib}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{datatool}[=v2.32]
\newcommand*{\dtlbib@style}{plain}
\define@choicekey{databib.sty}{style}{plain,abbrv,alpha}{%
\def\dtlbib@style{#1}}
\ProcessOptionsX
\newcommand*{\DTLloadbbl}[3][\jobname.bbl]{%
\bibliographystyle{databib}%
```

```
\if@filesw
 \immediate\write\@auxout{\string\bibdata{#3}}%
\fi
\DTLnewdb{#2}%
\edef\DTLBIBdbname{#2}%
\@input@{#1}}
\newcommand*{\DTLnewbibrow}{\@DTLnewrow{\DTLBIBdbname}}
\newcommand*{\DTLnewbibitem}[2]{%
  \@DTLnewdbentry{\DTLBIBdbname}{#1}{#2}}
\providecommand*{\andname}{and}
\providecommand*{\ofname}{of}
\providecommand*{\inname}{in}
\providecommand*{\etalname}{et al.}
\providecommand*{\editorname}{editor}
\providecommand*{\editorsname}{editors}
\providecommand*{\volumename}{volume}
\providecommand*{\numbername}{number}
\providecommand*{\pagesname}{pages}
\providecommand*{\pagename}{page}
\providecommand*{\editionname}{edition}
\providecommand*{\techreportname}{Technical report}
\providecommand*{\mscthesisname}{Master's thesis}
\providecommand*{\phdthesisname}{PhD thesis}
\newcommand*{\DTLbibliography}[2][\boolean{true}]{%
  \begin{DTLthebibliography}[#1]{#2}%
  \DTLforeachbibentry[#1]{#2}{%
    \DTLbibitem \DTLformatbibentry \DTLendbibitem
  }%
  \end{DTLthebibliography}%
}
\newcommand*{\DTLformatbibentry}{%
  \@ifundefined{DTLformat\DBIBentrytype}%
  {%
    \PackageError{databib}{Don't know how to format bibliography
    entries of type `\DBIBentrytype'}{}%
  }%
  {%
    \dtl@message{[\DBIBcitekey]}%
    \DTLstartsentencefalse\DTLmidsentencefalse\DTLperiodfalse
     \csname DTLformat\DBIBentrytype\endcsname
  }%
}
\newcommand*{\gDTLformatbibentry}{%
  \@ifundefined{DTLformat\DBIBentrytype}%
  {%
    \PackageError{databib}{Don't know how to format bibliography
    entries of type `\DBIBentrytype'}{}%
  }%
  {%
    \dtl@message{[\DBIBcitekey]}%
```

```
    \global\DTLstartsentencefalse
    \global\DTLmidsentencefalse
    \global\DTLperiodfalse
     \csname DTLformat\DBIBentrytype\endcsname
  }%
}
\newcommand*{\DTLformatthisbibentry}[2]{%
  \edef\DBIBname{#1}%
  \edef\DBIBcitekey{#2}%
  \edtlgetrowforvalue{#1}{\dtlcolumnindex{#1}{CiteKey}}{\DBIBcitekey}%
  \dtl@gathervalues{#1}{\dtlcurrentrow}%
  \letcs{\DBIBentrytype}{@dtl@key@EntryType}%
  \DTLformatbibentry
}
\newcommand*{\DTLendbibitem}{}
\newlength\dtl@widest
\newcommand*{\DTLcomputewidestbibentry}[4]{%
\dtl@widest=0pt\relax
\let#4=\@empty
\DTLforeachbibentry[#1]{#2}{%
\settowidth{\dtl@tmplength}{#3}%
\ifdim\dtl@tmplength>\dtl@widest\relax
 \dtl@widest=\dtl@tmplength
 \protected@edef#4{#3}%
\fi
}%
}
\newcommand*{\DTLforeachbibentry}{%
\@ifstar\@sDTLforeachbibentry\@DTLforeachbibentry}
\newcommand*{\@DTLforeachbibentry}[3][\boolean{true}]{%
  \edef\DBIBname{#2}%
  \setcounter{DTLbibrow}{0}%
  \@DTLforeach{#2}{\DBIBcitekey=CiteKey,\DBIBentrytype=EntryType}%
  {%
    \dtl@gathervalues{#2}{\dtlcurrentrow}%
    \ifthenelse{#1}{\refstepcounter{DTLbibrow}#3}{}%
  }%
}
\newcommand*{\@sDTLforeachbibentry}[3][\boolean{true}]{%
  \edef\DBIBname{#2}%
  \setcounter{DTLbibrow}{0}%
  \@sDTLforeach{#2}{\DBIBcitekey=CiteKey,\DBIBentrytype=EntryType}%
  {%
    \dtl@gathervalues{#2}{\dtlcurrentrow}%
    \ifthenelse{#1}{\refstepcounter{DTLbibrow}#3}{}%
  }%
}
\newcommand{\gDTLforeachbibentry}{%
\@ifstar\@sgDTLforeachbibentry\@gDTLforeachbibentry}
\newcommand*{\@gDTLforeachbibentry}[3][\boolean{true}]{%
```

```
  \xdef\DBIBname{#2}%
  \global\c@DTLbibrow = 0\relax
  \@DTLforeach{#2}{\DBIBcitekey=CiteKey,\DBIBentrytype=EntryType}%
  {%
    \dtl@g@gathervalues{#2}{\dtlcurrentrow}%
    \ifthenelse{#1}%
    {%
      \refstepcounter{DTLbibrow}%
      \global\c@DTLbibrow=\c@DTLbibrow
      #3%
    }%
    {}%
  }%
}
\newcommand*{\@sgDTLforeachbibentry}[3][\boolean{true}]{%
  \xdef\DBIBname{#2}%
  \global\c@DTLbibrow = 0\relax
  \@sDTLforeach{#2}{\DBIBcitekey=CiteKey,\DBIBentrytype=EntryType}%
  {%
    \dtl@g@gathervalues{#2}{\dtlcurrentrow}%
    \ifthenelse{#1}%
    {%
      \refstepcounter{DTLbibrow}%
      \global\c@DTLbibrow=\c@DTLbibrow
      #3%
    }%
    {}%
  }%
}
\newcounter{DTLbibrow}
\def\theHDTLbibrow{\theHDTLrow.bib.\arabic{DTLbibrow}}%
\newcommand*{\DTLbibfield}[1]{\csname @dtl@key@#1\endcsname}
\newcommand*{\DTLbibfieldlet}[2]{%
  \letcs{#1}{@dtl@key@#2}%
}
\newcommand*{\DTLifbibfieldexists}[3]{%
\@ifundefined{@dtl@key@#1}{#3}{%
\expandafter\DTLifnull\csname @dtl@key@#1\endcsname
{#3}{#2}}}
\newcommand*{\DTLifanybibfieldexists}[3]{%
\@for\dtl@thisfield:=#1\do{%
\@ifundefined{@dtl@key@\dtl@thisfield}{}{%
\expandafter\DTLifnull\csname @dtl@key@\dtl@thisfield\endcsname
{}{%
\@endfortrue}}}%
\if@endfor
 #2%
\else
 #3%
\fi
```

```
\@endforfalse
}
\newif\ifDTLperiod
\newcommand*{\DTLcheckendsperiod}[1]{%
\dtl@checkendsperiod#1\@nil\relax}
\def\dtl@checkendsperiod#1#2{%
\def\@dtl@argi{#1}\def\@dtl@argii{#2}%
\def\@dtl@period{.}%
\ifx\@dtl@argi\@nnil
  \global\DTLperiodfalse
  \let\@dtl@donext=\relax
\else
  \ifx\@dtl@argii\@nnil
    \ifx\@dtl@argi\@dtl@period
      \global\DTLperiodtrue
    \else
      \global\DTLperiodfalse
    \fi
    \let\@dtl@donext=\@gobble
  \else
    \let\@dtl@donext=\dtl@checkendsperiod
  \fi
\fi
\@dtl@donext{#2}%
}
\newcommand*{\DTLcheckbibfieldendsperiod}[1]{%
\protected@edef\@dtl@tmp{\DTLbibfield{#1}}%
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}}
\newif\ifDTLmidsentence
\newif\ifDTLstartsentence
\newcommand*{\DTLaddperiod}{\DTLmidsentencefalse\DTLperiodfalse
\DTLstartsentencetrue
\ifDTLperiod\else.\fi}
\newcommand*{\DTLaddcomma}{, \DTLmidsentencetrue
\DTLperiodfalse\DTLstartsentencefalse}
\newcommand*{\DTLstartsentencespace}{%
\ifDTLstartsentence\spacefactor=\sfcode`\.\relax\space
\fi\DTLstartsentencefalse}
\newcommand*{\DTLtwoand}{\ \andname\ }
\newcommand*{\DTLandlast}{, \andname\ }
\newcommand*{\DTLandnotlast}{, }
\newcount\@dtl@authorcount
\newcounter{DTLmaxauthors}
\setcounter{DTLmaxauthors}{10}
\newcommand*{\DTLformatauthorlist}{%
\DTLifbibfieldexists{Author}{%
\DTLstartsentencespace
\@dtl@authorcount=0\relax
\@for\@dtl@author:=\@dtl@key@Author\do{%
\advance\@dtl@authorcount by 1\relax}%
```

```
\@dtl@tmpcount=0\relax
\ifnum\@dtl@authorcount>\c@DTLmaxauthors
{%
  \@for\@dtl@author:=\@dtl@key@Author\do{%
  \advance\@dtl@tmpcount by 1\relax
  \ifnum\@dtl@tmpcount=1\relax
   \expandafter\DTLformatauthor\@dtl@author
  \else
    \ifnum\@dtl@tmpcount>\c@DTLmaxauthors
      \DTLandnotlast \etalname
      \expandafter\DTLcheckendsperiod\expandafter{\etalname}%
      \@endfortrue
    \else
      \DTLandnotlast \expandafter\DTLformatauthor\@dtl@author
    \fi
  \fi
  }%
}%
\else
  \@for\@dtl@author:=\@dtl@key@Author\do{%
  \advance\@dtl@tmpcount by 1\relax
  \ifnum\@dtl@tmpcount=1\relax
   \expandafter\DTLformatauthor\@dtl@author
  \else
    \ifnum\@dtl@tmpcount=\@dtl@authorcount
      \ifnum\@dtl@authorcount=2\relax
        \DTLtwoand
      \else
        \DTLandlast
      \fi
      \expandafter\DTLformatauthor\@dtl@author
    \else
      \DTLandnotlast \expandafter\DTLformatauthor\@dtl@author
    \fi
  \fi
  }%
\fi
}{}%
}
\newcounter{DTLmaxeditors}
\setcounter{DTLmaxeditors}{10}
\newcommand*{\DTLformateditorlist}{%
\DTLifbibfieldexists{Editor}{%
\DTLstartsentencespace
\@dtl@authorcount=0\relax
\@for\@dtl@author:=\@dtl@key@Editor\do{%
\advance\@dtl@authorcount by 1\relax}%
\@dtl@tmpcount=0\relax
\ifnum\@dtl@authorcount>\c@DTLmaxeditors
{%
```

```
  \@for\@dtl@author:=\@dtl@key@Editor\do{%
  \advance\@dtl@tmpcount by 1\relax
  \ifnum\@dtl@tmpcount=1\relax
   \expandafter\DTLformateditor\@dtl@author
  \else
    \ifnum\@dtl@tmpcount>\c@DTLmaxeditors
      \DTLandnotlast \etalname
      \expandafter\DTLcheckendsperiod\expandafter{\etalname}%
      \@endfortrue
    \else
      \DTLandnotlast \expandafter\DTLformateditor\@dtl@author
    \fi
  \fi
  }%
}%
\else
\@for\@dtl@author:=\@dtl@key@Editor\do{%
  \advance\@dtl@tmpcount by 1\relax
  \ifnum\@dtl@tmpcount=1\relax
   \expandafter\DTLformateditor\@dtl@author
  \else
    \ifnum\@dtl@tmpcount=\@dtl@authorcount
      \ifnum\@dtl@authorcount=2\relax
        \DTLtwoand
      \else
        \DTLandlast
      \fi
      \expandafter\DTLformateditor\@dtl@author
    \else
      \DTLandnotlast \expandafter\DTLformateditor\@dtl@author
    \fi
  \fi
  }%
\fi
,
\ifnum\@dtl@authorcount=1\relax
  \editorname
  \expandafter\DTLcheckendsperiod\expandafter{\editorname}%
\else
  \editorsname
  \expandafter\DTLcheckendsperiod\expandafter{\editorsname}%
\fi
}{}%
}
\newcommand*{\DTLformatsurnameonly}[4]{%
\DTLstartsentencespace
\def\@dtl@tmp{#1}%
\ifx\@dtl@tmp\@empty\else#1~\fi
#2%
\def\@dtl@tmp{#3}%
```

```
\ifx\@dtl@tmp\@empty
  \DTLcheckendsperiod{#2}%
\else
  , #3%
  \DTLcheckendsperiod{#3}%
\fi
}
\newcommand*{\DTLformatforenames}[1]{%
\DTLstartsentencespace
#1%
\DTLcheckendsperiod{#1}}
\newcommand*{\DTLformatabbrvforenames}[1]{%
\DTLstartsentencespace
\DTLstoreinitials{#1}{\@dtl@tmp}\@dtl@tmp
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}}
\newcommand*{\DTLformatvon}[1]{%
\DTLstartsentencespace
\def\@dtl@tmp{#1}%
\ifx\@dtl@tmp\@empty
\else
  #1~%
\fi
}
\newcommand*{\DTLformatsurname}[1]{%
\DTLstartsentencespace
#1\DTLcheckendsperiod{#1}}
\newcommand*{\DTLformatjr}[1]{%
\DTLstartsentencespace
\def\@dtl@tmp{#1}%
\ifx\@dtl@tmp\@empty
\else
  , #1\DTLcheckendsperiod{#1}%
\fi
}
\newcommand*{\DTLformatcrossrefeditor}{%
\DTLifbibfieldexists{Editor}{%
\DTLstartsentencespace
\@dtl@authorcount=0\relax
\@for\@dtl@author:=\@dtl@key@Editor\do{%
\advance\@dtl@authorcount by 1\relax}%
{\@dtl@tmpcount=0\relax
\@for\@dtl@author:=\@dtl@key@Editor\do{%
\ifnum\@dtl@authorcount=1\relax
 \expandafter\DTLformatsurnameonly\@dtl@author
\else
 \advance\@dtl@tmpcount by 1\relax
 \ifnum\@dtl@tmpcount=1\relax
    \expandafter\DTLformatsurnameonly\@dtl@author
  \else
    \ifnum\@dtl@authorcount=2\relax
```

```
      \ \andname\ \expandafter\DTLformatsurnameonly\@dtl@author
    \else
      \ \etalname
      \expandafter\DTLcheckendsperiod\expandafter{\etalname}
    \fi
    \@endfortrue
  \fi
\fi
}}%
}{}%
}
\newcommand*{\DTLformatvolnumpages}{%
\DTLifbibfieldexists{Volume}{%
\DTLstartsentencespace
\DTLbibfield{Volume}\DTLperiodfalse}{}%
\DTLifbibfieldexists{Number}{%
\DTLstartsentencespace
(\DTLbibfield{Number})\DTLperiodfalse}{}%
\DTLifbibfieldexists{Pages}{%
\DTLifanybibfieldexists{Volume,Number}{:}{%
\DTLstartsentencespace
\protected@edef\@dtl@pages{0\DTLbibfield{Pages}}%
\DTLifnumerical{\@dtl@pages}{\pagename}{\pagesname}~}%
\DTLbibfield{Pages}\DTLperiodfalse}{}%
}
\newcommand*{\DTLformatbvolume}{%
\DTLifbibfieldexists{Volume}{%
\ifDTLmidsentence
  \volumename
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\volumename
\fi
~\DTLbibfield{Volume}%
\DTLifbibfieldexists{Series}{\ \ofname\
{\em\DTLbibfield{Series}}\DTLcheckbibfieldendsperiod{Series}}{%
\DTLcheckbibfieldendsperiod{Volume}}%
}{}}
\newcommand*{\DTLformatchapterpages}{%
\DTLifbibfieldexists{Chapter}{%
\DTLifbibfieldexists{Type}{%
\DTLstartsentencespace
\DTLbibfield{Type}}{%
\DTLstartsentencespace
\chaptername}~\DTLbibfield{Chapter}%
\DTLifbibfieldexists{Pages}{\DTLaddcomma}{%
\DTLcheckbibfieldendsperiod{Chapter}}}{}%
\DTLstartsentencespace
\DTLformatpages}
\newcommand*{\DTLformatpages}{%
```

```
\DTLifbibfieldexists{Pages}{%
\DTLstartsentencespace
\protected@edef\@dtl@pages{0\DTLbibfield{Pages}}%
\DTLifnumerical{\@dtl@pages}{\pagename}{\pagesname}~%
\DTLbibfield{Pages}\DTLcheckbibfieldendsperiod{Pages}}{}%
}
\newcommand*{\DTLformatnumberseries}{%
\DTLifbibfieldexists{Volume}{}{%
\DTLifbibfieldexists{Number}{%
\ifDTLmidsentence
  \numbername
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\numbername
\fi~\DTLbibfield{Number}%
\DTLifbibfieldexists{Series}{\ \inname\ \DTLbibfield{Series}%
\DTLcheckbibfieldendsperiod{Series}}{%
\DTLcheckbibfieldendsperiod{Number}}%
}{%
\DTLifbibfieldexists{Series}{%
\DTLstartsentencespace
\DTLbibfield{Series}%
\DTLcheckbibfieldendsperiod{Series}}{}}%
}%
}
\newcommand*{\DTLformatbookcrossref}{%
\DTLifbibfieldexists{Volume}{%
\ifDTLmidsentence
  \volumename
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\volumename
\fi
~\DTLbibfield{Volume}\ \ofname\
}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\ }%
\DTLifbibfieldexists{Editor}{\DTLformatcrossrefeditor}{%
\DTLifbibfieldexists{Key}{%
\DTLbibfield{Key}}{%
\DTLifbibfieldexists{Series}{%
{\em\DTLbibfield{Series}}}{}%
}%
}%
~\DTLpcite{\DTLbibfield{CrossRef}}%
}
```

```
\newcommand*{\DTLformatincollproccrossref}{%
\DTLifbibfieldexists{Editor}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\
\DTLformatcrossrefeditor
}{%
\DTLifbibfieldexists{Key}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\ \DTLbibfield{Key}%
}{%
\DTLifbibfieldexists{BookTitle}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\ \DTLformatbooktitle{\DTLbibfield{BookTitle}}}{}%
}}%
~\DTLpcite{\DTLbibfield{CrossRef}}%
}
\newcommand*{\DTLformatinedbooktitle}{%
\DTLifbibfieldexists{BookTitle}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\
\DTLifbibfieldexists{Editor}{%
\DTLformateditorlist\DTLaddcomma \DTLformatbooktitle{\DTLbibfield{BookTitle}}%
\DTLcheckbibfieldendsperiod{BookTitle}%
}{\DTLformatbooktitle{\DTLbibfield{BookTitle}}%
\DTLcheckbibfieldendsperiod{BookTitle}%
}}{}}
\newcommand*{\DTLformatdate}{%
\DTLifbibfieldexists{Year}{%
\DTLifbibfieldexists{Month}{%
\protected@edef\@dtl@tmp{\DTLbibfield{Month}}%
\ifDTLmidsentence
  \@dtl@tmp
\else
  \DTLstartsentencespace
```

```
  \expandafter\MakeUppercase\@dtl@tmp
\fi\
\DTLmidsentencefalse}{}%
\DTLstartsentencespace
\DTLbibfield{Year}}{%
\DTLifbibfieldexists{Month}{%
\protected@edef\@dtl@tmp{\DTLbibfield{Month}}%
\ifDTLmidsentence
  \@dtl@tmp
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\@dtl@tmp
\fi
\DTLcheckbibfieldendsperiod{Month}%
}{}}}
\newcommand*{\DTLformatarticlecrossref}{%
\DTLifbibfieldexists{Key}{%
\ifDTLmidsentence
 \inname
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\inname
\fi
\ {\em\DTLbibfield{Key}}}{%
\DTLifbibfieldexists{Journal}{%
\ifDTLmidsentence
 \inname
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\inname
\fi
\ {\em\DTLbibfield{Journal}}}{}}%
~\DTLpcite{\DTLbibfield{CrossRef}}%
}
\newrobustcmd*{\DTLpcite}[1]{%
  \protected@edef\@dtl@tmp{#1}%
  \cite{\@dtl@tmp}%
}
\newcommand*{\DTLbibfieldexists}[1]{%
\TE@throw\noexpand\dtl@testbibfieldexists{#1}%
\noexpand\if@dtl@condition}
\newcommand*{\dtl@testbibfieldexists}[1]{%
\DTLifbibfieldexists{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
\newcommand*{\DTLbibfieldiseq}[2]{%
\TE@throw\noexpand\dtl@testbibfieldiseq{#1}{#2}%
\noexpand\if@dtl@condition}
\newcommand*{\dtl@testbibfieldiseq}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
```

```
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount=0\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
\newcommand*{\DTLbibfieldislt}[2]{%
\TE@throw\noexpand\dtl@testbibfieldislt{#1}{#2}%
\noexpand\if@dtl@condition}
\newcommand*{\dtl@testbibfieldislt}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount=-1\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
\newcommand*{\DTLbibfieldisle}[2]{%
\TE@throw\noexpand\dtl@testbibfieldisle{#1}{#2}%
\noexpand\if@dtl@condition}
\newcommand*{\dtl@testbibfieldisle}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount<1\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
```

```
}{%
\@dtl@conditionfalse}%
}
\newcommand*{\DTLbibfieldisgt}[2]{%
\TE@throw\noexpand\dtl@testbibfieldisgt{#1}{#2}%
\noexpand\if@dtl@condition}
\newcommand*{\dtl@testbibfieldisgt}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount=1\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
\newcommand*{\DTLbibfieldisge}[2]{%
\TE@throw\noexpand\dtl@testbibfieldisge{#1}{#2}%
\noexpand\if@dtl@condition}
\newcommand*{\dtl@testbibfieldisge}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount>-1\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
\newcommand*{\DTLbibfieldcontains}[2]{%
\TE@throw\noexpand\dtl@testbibfieldcontains{#1}{#2}%
\noexpand\if@dtl@condition}
\newcommand*{\dtl@testbibfieldcontains}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
  =\csname @dtl@key@#1\endcsname
```

```
\expandafter\dtl@testifsubstring\expandafter{\@dtl@tmp}{#2}%
}{\@dtl@conditionfalse}}
\newenvironment{DTLthebibliography}[2][\boolean{true}]{%
\@dtl@tmpcount=0\relax
\@sDTLforeach[#1]{#2}{}{\advance\@dtl@tmpcount by 1\relax}%
\begin{thebibliography}{\number\@dtl@tmpcount}
}{\end{thebibliography}}
\newcommand*{\DTLmonthname}[1]{%
\dtl@monthname{#1}}
\newcommand*{\dtl@monthname}[1]{%
\ifcase#1%
\or January%
\or February%
\or March%
\or April%
\or May%
\or June%
\or July%
\or August%
\or September%
\or October%
\or November%
\or December%
\fi}
\newcommand*{\dtl@abbrvmonthname}[1]{%
\ifcase#1%
\or Jan.%
\or Feb.%
\or Mar.%
\or Apr.%
\or May%
\or June%
\or July%
\or Aug.%
\or Sept.%
\or Oct.%
\or Nov.%
\or Dec.%
\fi}
\newcommand*{\DTLbibitem}{\bibitem{\DBIBcitekey}}
\newcommand*{\DTLmbibitem}[1]{\bibitem{#1@\DBIBcitekey}}
\newcommand*{\DTLcustombibitem}[3]{%
  #1%
  \if@filesw
    \immediate\write\@auxout{\string\bibcite{#3}{#2}}%
  \fi
  \ignorespaces
}
\newcommand*{\DTLformatauthor}[4]{%
\DTLformatforenames{#4}
```

```
\DTLformatvon{#1}%
\DTLformatsurname{#2}%
\DTLformatjr{#3}}
\newcommand*{\DTLformateditor}[4]{%
\DTLformatforenames{#4}
\DTLformatvon{#1}%
\DTLformatsurname{#2}%
\DTLformatjr{#3}}
\newcommand*{\DTLformatedition}[1]{#1 \editionname}
\newcommand{\DTLformatarticle}{}
\newcommand{\DTLformatbook}{}
\newcommand{\DTLformatbooklet}{}
\newcommand{\DTLformatinbook}{}
\newcommand{\DTLformatincollection}{}
\newcommand{\DTLformatinproceedings}{}
\newcommand{\DTLformatmanual}{}
\newcommand{\DTLformatmasterthesis}{}
\newcommand{\DTLformatmisc}{}
\newcommand{\DTLformatphdthesis}{}
\newcommand{\DTLformatproceedings}{}
\newcommand{\DTLformattechreport}{}
\newcommand{\DTLformatunpublished}{}
\newcommand*{\DTLacmcs}{ACM Computing Surveys}
\newcommand*{\DTLacta}{Acta Informatica}
\newcommand*{\DTLcacm}{Communications of the ACM}
\newcommand*{\DTLibmjrd}{IBM Journal of Research and Development}
\newcommand*{\DTLibmsj}{IBM Systems Journal}
\newcommand*{\DTLieeese}{IEEE Transactions on Software Engineering}
\newcommand*{\DTLieeetc}{IEEE Transactions on Computers}
\newcommand*{\DTLieeetcad}{IEEE Transactions on Computer-Aided Design
of Integrated Circuits}
\newcommand*{\DTLipl}{Information Processing Letters}
\newcommand*{\DTLjacm}{Journal of the ACM}
\newcommand*{\DTLjcss}{Journal of Computer and System Sciences}
\newcommand*{\DTLscp}{Science of Computer Programming}
\newcommand*{\DTLsicomp}{SIAM Journal on Computing}
\newcommand*{\DTLtocs}{ACM Transactions on Computer Systems}
\newcommand*{\DTLtods}{ACM Transactions on Database Systems}
\newcommand*{\DTLtog}{ACM Transactions on Graphics}
\newcommand*{\DTLtoms}{ACM Transactions on Mathematical Software}
\newcommand*{\DTLtoois}{ACM Transactions on Office Information
Systems}
\newcommand*{\DTLtoplas}{ACM Transactions on Programming Languages
and Systems}
\newcommand*{\DTLtcs}{Theoretical Computer Science}
\newcommand{\dtlbst@plain}{%
  \renewenvironment{DTLthebibliography}[2][\boolean{true}]{%
  \@dtl@tmpcount=0\relax
  \@sDTLforeach[##1]{##2}{}{\advance\@dtl@tmpcount by 1\relax}%
  \begin{thebibliography}{\number\@dtl@tmpcount}%
```

```
  }{\end{thebibliography}}%
  \renewcommand*{\DTLbibitem}{\bibitem{\DBIBcitekey}}%
  \renewcommand*{\DTLmbibitem}[1]{\bibitem{##1@\DBIBcitekey}}%
  \renewcommand*{\DTLformatauthor}[4]{%
  \DTLformatforenames{##4}
  \DTLformatvon{##1}%
  \DTLformatsurname{##2}%
  \DTLformatjr{##3}}
  \renewcommand*{\DTLformateditor}[4]{%
  \DTLformatforenames{##4}
  \DTLformatvon{##1}%
  \DTLformatsurname{##2}%
  \DTLformatjr{##3}}
  \renewcommand*{\DTLformatedition}[1]{##1 \editionname}%
  \let\DTLmonthname\dtl@monthname
  \renewcommand*{\DTLacmcs}{ACM Computing Surveys}
  \renewcommand*{\DTLacta}{Acta Informatica}
  \renewcommand*{\DTLcacm}{Communications of the ACM}
  \renewcommand*{\DTLibmjrd}{IBM Journal of Research and Development}
  \renewcommand*{\DTLibmsj}{IBM Systems Journal}
  \renewcommand*{\DTLieeese}{IEEE Transactions on Software Engineering}
  \renewcommand*{\DTLieeetc}{IEEE Transactions on Computers}
  \renewcommand*{\DTLieeetcad}{IEEE Transactions on Computer-Aided Design
of Integrated Circuits}
  \renewcommand*{\DTLipl}{Information Processing Letters}
  \renewcommand*{\DTLjacm}{Journal of the ACM}
  \renewcommand*{\DTLjcss}{Journal of Computer and System Sciences}
  \renewcommand*{\DTLscp}{Science of Computer Programming}
  \renewcommand*{\DTLsicomp}{SIAM Journal on Computing}
  \renewcommand*{\DTLtocs}{ACM Transactions on Computer Systems}
  \renewcommand*{\DTLtods}{ACM Transactions on Database Systems}
  \renewcommand*{\DTLtog}{ACM Transactions on Graphics}
  \renewcommand*{\DTLtoms}{ACM Transactions on Mathematical Software}
  \renewcommand*{\DTLtoois}{ACM Transactions on Office Information
Systems}
  \renewcommand*{\DTLtoplas}{ACM Transactions on Programming Languages
and Systems}
  \renewcommand*{\DTLtcs}{Theoretical Computer Science}
  \renewcommand*{\DTLformatarticle}{%
    \DTLformatauthorlist
    \DTLifbibfieldexists{Author}{\DTLaddperiod}{}%
    \DTLifbibfieldexists{Title}{%
    \DTLstartsentencespace\DTLbibfield{Title}%
    \DTLcheckbibfieldendsperiod{Title}%
    \DTLaddperiod}{}%
    \DTLifbibfieldexists{CrossRef}{%
    \DTLformatarticlecrossref
    \DTLifbibfieldexists{Pages}{\DTLaddcomma}{}%
    \DTLformatpages
    \DTLaddperiod
```

```
    }{% no cross ref field
    \DTLifbibfieldexists{Journal}{\DTLstartsentencespace
    {\em\DTLbibfield{Journal}}%
    \DTLcheckbibfieldendsperiod{Journal}%
    \DTLifanybibfieldexists{Number,Volume,Pages,Month,Year}{%
    \DTLaddcomma}{\DTLaddperiod}}{}%
    \DTLformatvolnumpages
    \DTLifanybibfieldexists{Volume,Number,Pages}{%
    \DTLifanybibfieldexists{Year,Month}{\DTLaddcomma}{%
    \DTLaddperiod}%
    \DTLmidsentencefalse}{}%
    \DTLformatdate
    \DTLifanybibfieldexists{Year,Month}{\DTLaddperiod}{}%
    }%
    \DTLifbibfieldexists{Note}{\DTLstartsentencespace\DTLbibfield{Note}%
    \DTLcheckbibfieldendsperiod{Note}%
    \DTLaddperiod}{}%
  }
\renewcommand*{\DTLformatbook}{%
  \DTLifbibfieldexists{Author}%
  {%
    \DTLformatauthorlist\DTLaddperiod
  }%
  {%
    \DTLformateditorlist
    \DTLifbibfieldexists{Editor}%
    {%
      \DTLaddperiod
    }%
    {}%
  }%
  \DTLifbibfieldexists{Title}%
  {%
    \DTLstartsentencespace
    \DTLformatbooktitle{\DTLbibfield{Title}}%
    \DTLcheckbibfieldendsperiod{Title}%
  }%
  {}%
  \DTLifbibfieldexists{CrossRef}%
  {%
    \DTLifbibfieldexists{Title}{\DTLaddperiod}{}%
    \DTLformatbookcrossref
    \DTLifanybibfieldexists{Edition,Month,Year}%
    {\DTLaddcomma}%
    {\DTLaddperiod}%
  }%
  {%
    \DTLifbibfieldexists{Title}%
    {%
      \DTLifbibfieldexists{Volume}{\DTLaddcomma}{\DTLaddperiod}%
```

```
  }%
  {}%
  \DTLformatbvolume
  \DTLformatnumberseries
  \DTLifanybibfieldexists{Number,Series,Volume}{\DTLaddperiod}{}%
  \DTLifbibfieldexists{Publisher}%
  {%
    \DTLstartsentencespace
    \DTLbibfield{Publisher}%
    \DTLcheckbibfieldendsperiod{Publisher}%
    \DTLifbibfieldexists{Address}%
    {\DTLaddcomma}%
    {%
      \DTLifanybibfieldexists{Month,Year}%
      {\DTLaddcomma}%
      {\DTLaddperiod}%
    }%
  }%
  {}%
  \DTLifbibfieldexists{Address}%
  {%
    \DTLstartsentencespace
    \DTLbibfield{Address}%
    \DTLcheckbibfieldendsperiod{Address}%
    \DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}%
  }%
  {}%
}%
\DTLifbibfieldexists{Edition}%
{%
  \protected@edef\@dtl@tmp{\DTLformatedition{\DTLbibfield{Edition}}}%
  \ifDTLmidsentence
   \@dtl@tmp
  \else
   \DTLstartsentencespace\expandafter\MakeUppercase\@dtl@tmp
  \fi
  \expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}%
  \DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}%
}%
{}%
\DTLformatdate
\DTLifanybibfieldexists{Year,Month}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}%
{%
  \DTLstartsentencespace
  \DTLbibfield{Note}%
  \DTLcheckbibfieldendsperiod{Note}%
  \DTLaddperiod
}%
{}%
```

```
}%
\renewcommand*{\DTLformatbooklet}{%
\DTLifbibfieldexists{Author}{%
\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{HowPublished}{%
\DTLstartsentencespace\DTLbibfield{HowPublished}%
\DTLcheckbibfieldendsperiod{HowPublished}%
\DTLifanybibfieldexists{Address,Month,Year}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Year,Month}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{\DTLstartsentencespace\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatinbook}{%
\DTLifbibfieldexists{Author}{%
\DTLformatauthorlist\DTLaddperiod}{%
\DTLifbibfieldexists{Editor}{\DTLformateditorlist\DTLaddperiod}{}}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
{\em\DTLbibfield{Title}}%
\DTLcheckbibfieldendsperiod{Title}%
}{}%
\DTLifbibfieldexists{CrossRef}{%
\DTLifbibfieldexists{Title}{%
\DTLifbibfieldexists{Chapter}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatchapterpages
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddperiod}{}%
\DTLformatbookcrossref
}{% no cross ref
\DTLifbibfieldexists{Title}{%
\DTLifanybibfieldexists{Chapter,Volume}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLformatbvolume
\DTLifanybibfieldexists{Volume,Series}{%
\DTLifanybibfieldexists{Chapter,Pages}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatchapterpages
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
```

```
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLifbibfieldexists{Address}{\DTLaddcomma}{}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}}{}%
}%
\DTLifanybibfieldexists{Edition,Month,Year}{\DTLaddcomma
}{\DTLaddperiod}%
\DTLifbibfieldexists{Edition}{%
\protected@edef\@dtl@tmp{\DTLformatedition{\DTLbibfield{Edition}}}%
\ifDTLmidsentence
 \@dtl@tmp
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\@dtl@tmp
\fi
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma
}{\DTLaddperiod}%
}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatincollection}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{CrossRef}{%
\DTLformatincollproccrossref
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddcomma}{}%
\DTLformatchapterpages\DTLaddperiod
}{% no cross ref entry
\DTLformatinedbooktitle
\DTLifbibfieldexists{BookTitle}{%
\DTLifanybibfieldexists{Volume,Series,Chapter,Pages,Number}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatbvolume
\DTLifbibfieldexists{Volume}{%
\DTLifanybibfieldexists{Number,Series,Chapter,Pages}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
```

```
\DTLformatnumberseries
\DTLifanybibfieldexists{Number,Series}{%
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLformatchapterpages
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLifanybibfieldexists{Address,Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Edition}{%
\protected@edef\@dtl@tmp{\DTLformatedition{\DTLbibfield{Edition}}}%
\ifDTLmidsentence
 \@dtl@tmp
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\@dtl@tmp
\fi
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma
}{\DTLaddperiod}%
}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatinproceedings}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist
\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{CrossRef}{%
\DTLformatincollproccrossref
\DTLifbibfieldexists{Pages}{\DTLaddcomma}{%
```

```
\DTLaddperiod}%
\DTLformatpages
\DTLifbibfieldexists{Pages}{\DTLaddperiod}{}%
}{% no cross ref
\DTLformatinedbooktitle
\DTLifbibfieldexists{BookTitle}{%
\DTLifanybibfieldexists{Volume,Series,Pages,Number,Address,%
Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatbvolume
\DTLifbibfieldexists{Volume}{%
\DTLifanybibfieldexists{Number,Series,Pages,Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatnumberseries
\DTLifanybibfieldexists{Number,Series}{%
\DTLifanybibfieldexists{Pages,Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatpages
\DTLifbibfieldexists{Pages}{%
\DTLifanybibfieldexists{Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifbibfieldexists{Publisher}{\DTLaddcomma}{%
\DTLaddperiod}}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLaddperiod}{}%
}{%
\DTLifanybibfieldexists{Publisher,Organization}{%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifanybibfieldexists{Publisher,Month,Year}{%
\DTLaddcomma}{}}{}%
\DTLifbibfieldexists{Publisher}{%
```

```
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
}%
}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatmanual}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist
\DTLaddperiod}{%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifbibfieldexists{Address}{\DTLaddcomma \DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
}{}%
\DTLaddperiod}{}%
}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
{\em\DTLbibfield{Title}}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLifbibfieldexists{Author}{%
\DTLifanybibfieldexists{Organization,Address}{%
\DTLaddperiod}{\DTLaddcomma}}{%
\DTLifanybibfieldexists{Organization,Address,Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}}{}%
\DTLifbibfieldexists{Author}{%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifanybibfieldexists{Address,Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
}{%
```

```
\DTLifbibfieldexists{Organization}{}{%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Edition,Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}}{}}%
}%
\DTLifbibfieldexists{Edition}{%
\protected@edef\@dtl@tmp{\DTLformatedition{\DTLbibfield{Edition}}}%
\ifDTLmidsentence
 \@dtl@tmp
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\@dtl@tmp
\fi
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatmastersthesis}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Type}{%
\DTLstartsentencespace
\DTLbibfield{Type}%
\DTLcheckbibfieldendsperiod{Type}%
\DTLifanybibfieldexists{School,Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{School}{%
\DTLstartsentencespace
\DTLbibfield{School}%
\DTLcheckbibfieldendsperiod{School}%
\DTLifanybibfieldexists{Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
```

```
\DTLifanybibfieldexists{Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatmisc}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLifbibfieldexists{HowPublished}{\DTLaddperiod}{%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}%
}%
\DTLmidsentencefalse}{}%
\DTLifbibfieldexists{HowPublished}{%
\DTLstartsentencespace
\DTLbibfield{HowPublished}%
\DTLcheckbibfieldendsperiod{HowPublished}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatphdthesis}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
{\em\DTLbibfield{Title}}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Type}{%
\DTLstartsentencespace
\DTLbibfield{Type}%
\DTLcheckbibfieldendsperiod{Type}%
\DTLifanybibfieldexists{School,Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{School}{%
\DTLstartsentencespace
```

```
\DTLbibfield{School}%
\DTLcheckbibfieldendsperiod{School}%
\DTLifanybibfieldexists{Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatproceedings}{%
\DTLifbibfieldexists{Editor}{%
\DTLformateditorlist\DTLaddperiod}{%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLaddperiod}{}}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
{\em\DTLbibfield{Title}}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLifanybibfieldexists{Volume,Number,Address,Editor,Publisher,%
Month,Year}{\DTLaddcomma}{\DTLaddperiod}%
}{}%
\DTLformatbvolume
\DTLifbibfieldexists{Volume}{%
\DTLifanybibfieldexists{Number,Address,Editor,Publisher,%
Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatnumberseries
\DTLifbibfieldexists{Number}{%
\DTLifanybibfieldexists{Address,Editor,Publisher,%
Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Editor}{\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
```

```
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifbibfieldexists{Publisher}{%
\DTLaddcomma}{\DTLaddperiod}}{}}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLaddperiod
}{}%
}{% no address
\DTLifbibfieldexists{Editor}{%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifanybibfieldexists{Publisher,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformattechreport}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Type}{%
\DTLstartsentencespace
\DTLbibfield{Type}%
\DTLcheckbibfieldendsperiod{Type}%
\DTLifbibfieldexists{Number}{~}{}}{}%
\DTLifbibfieldexists{Number}{%
\DTLstartsentencespace
\DTLbibfield{Number}%
\DTLcheckbibfieldendsperiod{Number}%
}{}%
```

```
\DTLifanybibfieldexists{Type,Number}{%
\DTLifanybibfieldexists{Institution,Address,Month,Year}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Institution}{%
\DTLstartsentencespace
\DTLbibfield{Institution}%
\DTLcheckbibfieldendsperiod{Institution}%
\DTLifanybibfieldexists{Address,Month,Year}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
\renewcommand*{\DTLformatunpublished}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
}%
}
\newcommand*{\DTLformatbooktitle}[1]{\emph{#1}}
\newcommand{\dtlbst@abbrv}{%
\dtlbst@plain
\renewcommand*{\DTLformatauthor}[4]{%
\DTLformatabbrvforenames{##4}
\DTLformatvon{##1}%
\DTLformatsurname{##2}%
\DTLformatjr{##3}}
\renewcommand*{\DTLformateditor}[4]{%
\DTLformatabbrvforenames{##4}
\DTLformatvon{##1}%
```

```
\DTLformatsurname{##2}%
\DTLformatjr{##3}}
\let\DTLmonthname\dtl@abbrvmonthname
\renewcommand*{\DTLacmcs}{ACM Comput.\ Surv.}
\renewcommand*{\DTLacta}{Acta Inf.}
\renewcommand*{\DTLcacm}{Commun.\ ACM}
\renewcommand*{\DTLibmjrd}{IBM J.\ Res.\ Dev.}
\renewcommand*{\DTLibmsj}{IBM Syst.~J.}
\renewcommand*{\DTLieeese}{IEEE Trans. Softw.\ Eng.}
\renewcommand*{\DTLieeetc}{IEEE Trans.\ Comput.}
\renewcommand*{\DTLieeetcad}{IEEE Trans.\ Comput.-Aided Design
Integrated Circuits}
\renewcommand*{\DTLipl}{Inf.\ Process.\ Lett.}
\renewcommand*{\DTLjacm}{J.~ACM}
\renewcommand*{\DTLjcss}{J.~Comput.\ Syst.\ Sci.}
\renewcommand*{\DTLscp}{Sci.\ Comput.\ Programming}
\renewcommand*{\DTLsicomp}{SIAM J.~Comput.}
\renewcommand*{\DTLtocs}{ACM Trans.\ Comput.\ Syst.}
\renewcommand*{\DTLtods}{ACM Trans.\ Database Syst.}
\renewcommand*{\DTLtog}{ACM Trans.\ Gr.}
\renewcommand*{\DTLtoms}{ACM Trans.\ Math. Softw.}
\renewcommand*{\DTLtoois}{ACM Trans. Office Inf.\ Syst.}
\renewcommand*{\DTLtoplas}{ACM Trans.\ Prog. Lang.\ Syst.}
\renewcommand*{\DTLtcs}{Theoretical Comput.\ Sci.}
}
\newcommand{\dtlbst@alpha}{%
\dtlbst@plain
\renewenvironment{DTLthebibliography}[2][\boolean{true}]{%
\dtl@createalphabiblabels{##1}{##2}%
\begin{thebibliography}{\@dtl@widestlabel}%
}{\end{thebibliography}}%
\renewcommand*{\DTLbibitem}{%
\expandafter\bibitem\expandafter
 [\csname dtl@biblabel@\DBIBcitekey\endcsname]{\DBIBcitekey}}%
\renewcommand*{\DTLmbibitem}[1]{%
\expandafter\bibitem\expandafter
 [\csname dtl@biblabel@\DBIBcitekey\endcsname]{##1@\DBIBcitekey}}%
}
\newcommand*{\dtl@createalphabiblabels}[2]{%
\dtl@message{Creating bib labels}%
\begingroup
\gdef\@dtl@widestlabel{}%
\dtl@widest=0pt\relax
\DTLforeachbibentry[#1]{#2}{%
\dtl@message{\DBIBcitekey}%
\DTLifbibfieldexists{Author}{%
  \dtl@listgetalphalabel{\@dtl@thislabel}{\@dtl@key@Author}%
}{%
\DTLifbibfieldexists{Editor}{%
    \dtl@listgetalphalabel{\@dtl@thislabel}{\@dtl@key@Editor}%
```

```
 }{%
   \DTLifbibfieldexists{Key}{%
     \expandafter\dtl@get@firstthree\expandafter
       {\@dtl@key@Key}{\@dtl@thislabel}%
   }{%
     \DTLifbibfieldexists{Organization}{%
       \expandafter\dtl@get@firstthree\expandafter
         {\@dtl@key@Organization}{\@dtl@thislabel}%
   }{%
       \expandafter\dtl@get@firstthree\expandafter
         {\DBIBentrytype}{\@dtl@thislabel}%
   }%
   }}}%
\DTLifbibfieldexists{Year}{}{\DTLifbibfieldexists{CrossRef}{%
\DTLgetvalueforkey{\@dtl@key@Year}{Year}{#2}{CiteKey}{%
\@dtl@key@CrossRef}}{}}%
\DTLifbibfieldexists{Year}{%
\expandafter\dtl@get@yearsuffix\expandafter{\@dtl@key@Year}%
\expandafter\toks@\expandafter{\@dtl@thislabel}%
\expandafter\@dtl@toks\expandafter{\@dtl@year}%
\edef\@dtl@thislabel{\the\toks@\the\@dtl@toks}%
}{}%
\let\@dtl@s@thislabel=\@dtl@thislabel
\@onelevel@sanitize\@dtl@s@thislabel
\@ifundefined{c@biblabel@\@dtl@s@thislabel}{%
\newcounter{biblabel@\@dtl@s@thislabel}%
\setcounter{biblabel@\@dtl@s@thislabel}{1}%
\expandafter\edef\csname @dtl@bibfirst@\@dtl@s@thislabel\endcsname{%
\DBIBcitekey}%
\expandafter\global
\expandafter\let\csname dtl@biblabel@\DBIBcitekey\endcsname=
  \@dtl@thislabel
}{%
\expandafter\ifnum\csname c@biblabel@\@dtl@s@thislabel\endcsname=1\relax
 \expandafter\let\expandafter\@dtl@tmp
   \csname @dtl@bibfirst@\@dtl@s@thislabel\endcsname
 \expandafter\protected@xdef\csname dtl@biblabel@\@dtl@tmp\endcsname{%
   \@dtl@thislabel a}%
\fi
\stepcounter{biblabel@\@dtl@s@thislabel}%
\expandafter\protected@xdef\csname dtl@biblabel@\DBIBcitekey\endcsname{%
   \@dtl@thislabel\alph{biblabel@\@dtl@s@thislabel}}%
}%
\settowidth{\dtl@tmplength}{%
 \csname dtl@biblabel@\DBIBcitekey\endcsname}%
\ifdim\dtl@tmplength>\dtl@widest
 \dtl@widest=\dtl@tmplength
 \expandafter\global\expandafter\let\expandafter\@dtl@widestlabel
  \expandafter=\csname dtl@biblabel@\DBIBcitekey\endcsname
\fi
```

```
}%
\endgroup
}
\newcommand*{\dtl@listgetalphalabel}[2]{%
\@dtl@authorcount=0\relax
\@for\@dtl@author:=#2\do{%
\advance\@dtl@authorcount by 1\relax}%
\ifnum\@dtl@authorcount=1\relax
 \expandafter\dtl@getsinglealphalabel#2{#1}\relax
\else
  {%
  \xdef#1{}%
  \@dtl@tmpcount=0\relax
   \def\DTLafterinitials{}\def\DTLbetweeninitials{}%
   \def\DTLafterinitialbeforehyphen{}\def\DTLinitialhyphen{}%
   \@for\@dtl@author:=#2\do{%
     \expandafter\dtl@getauthorinitial\@dtl@author
     \expandafter\toks@\expandafter{\@dtl@tmp}%
     \expandafter\@dtl@toks\expandafter{#1}%
     \xdef#1{\the\@dtl@toks\the\toks@}%
     \advance\@dtl@tmpcount by 1\relax
     \ifnum\@dtl@tmpcount>2\relax\@endfortrue\fi
  }}%
\fi
}
\newcommand*{\dtl@getauthorinitial}[4]{%
\def\@dtl@vonpart{#1}%
\ifx\@dtl@vonpart\@empty
 \DTLstoreinitials{#2}{\@dtl@tmp}%
\else
 \DTLstoreinitials{#1 #2}{\@dtl@tmp}%
\fi}
\newcommand*{\dtl@getsinglealphalabel}[5]{%
\def\@dtl@vonpart{#1}%
\ifx\@dtl@vonpart\@empty
 \DTLifSubString{#2}{-}{%
   {\def\DTLafterinitials{}\def\DTLbetweeninitials{}%
    \def\DTLafterinitialbeforehyphen{}%
    \def\DTLinitialhyphen{}%
    \DTLstoreinitials{#2}{\@dtl@tmp}\global\let#5=\@dtl@tmp}%
   }{%
   \dtl@getfirstthree{#5}#2{}{}{}{}\@nil
 }
\else
 {\def\DTLafterinitials{}\def\DTLbetweeninitials{}%
  \def\DTLafterinitialbeforehyphen{}%
  \def\DTLinitialhyphen{}%
  \DTLstoreinitials{#1 #2}{\@dtl@tmp}\global\let#5=\@dtl@tmp}%
\fi
}
```

```
\def\dtl@getfirstthree#1#2#3#4#5\@nil{%
  \def#1{#2#3#4}%
}
\newcommand*{\dtl@get@firstthree}[2]{%
\dtl@getfirstthree#2#1{}{}{}{}{}\@nil}
\newcommand*{\dtl@get@yearsuffix}[1]{%
\dtl@getyearsuffix#1\@nil\relax\relax}

\def\dtl@getyearsuffix#1#2#3{%
\def\@dtl@argi{#1}\def\@dtl@argii{#2}%
\def\@dtl@argiii{#3}%
\ifx\@dtl@argi\@nnil
 \def\@dtl@year{}%
 \let\@dtl@donext=\relax
\else
  \ifx\@dtl@argii\@nnil
    \dtl@ifsingle{#1}{%
      \def\@dtl@year{#1}%
      \let\@dtl@donext=\relax
    }{%
      \def\@dtl@donext{\dtl@getyearsuffix#1#2#3}%
    }%
  \else
    \ifx\@dtl@argiii\@nnil
      \dtl@ifsingle{#1}{%
        \dtl@ifsingle{#2}{%
          \def\@dtl@year{#1#2}%
          \let\@dtl@donext=\relax
        }{%
          \def\@dtl@donext{\dtl@getyearsuffix#2#3}%
        }%
      }{%
        \def\@dtl@donext{\dtl@getyearsuffix#2#3}%
      }%
    \else
      \def\@dtl@donext{\dtl@getyearsuffix{#2}{#3}}%
    \fi
  \fi
\fi
\@dtl@donext
}
\newcommand*{\DTLbibliographystyle}[1]{%
\@ifundefined{dtlbst@#1}{\PackageError{databib}{Unknown
bibliography style `#1'}{}}{\csname dtlbst@#1\endcsname}}
\DTLbibliographystyle{\dtlbib@style}
\newcommand*{\DTLmultibibs}[1]{%
\@for\@dtl@af:=#1\do{%
\@ifundefined{dtl@aux@\@dtl@af}{%
\expandafter\newwrite\csname dtl@aux@\@dtl@af\endcsname
\expandafter\immediate
```

```
\expandafter\openout\csname dtl@aux@\@dtl@af\endcsname=\@dtl@af.aux
\expandafter\def\csname b@\@dtl@af @*\endcsname{}%
}{%
\PackageError{databib}{Can't create auxiliary file `\@dtl@af.aux',
\expandafter\string\csname dtl@aux@\@dtl@af\endcsname\space
already exists}{}}}}
\@onlypreamble{\DTLmultibibs}
\newcommand*{\DTLcite}{\@ifnextchar[{\@tempswatrue \dtl@citex
}{\@tempswafalse \dtl@citex[]}}
\def\dtl@citex[#1]#2#3{%
\leavevmode\let\@citea\@empty
\@cite{\@for\@citeb:=#3\do{\@citea
  \def\@citea{,\penalty \@m \ }%
  \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
  \if@filesw
    \@ifundefined{dtl@aux@#2}{%
      \PackageError{databib}{multibib `#2' not defined}{%
      You need to define `#2' in \string\DTLmutlibibs}%
    }{%
      \expandafter\immediate
      \expandafter\write\csname dtl@aux@#2\endcsname{%
        \string\citation{\@citeb}}%
    }%
  \fi
  \@ifundefined{b@#2@\@citeb}{%
    \hbox{\reset@font\bfseries ?}%
    \G@refundefinedtrue
    \@latex@warning{Citation `\@citeb ' on page \thepage \space
      undefined}%
  }{%
    \@cite@ofmt{\csname b@#2@\@citeb \endcsname }%
  }%
}}{#1}%
}
\newcommand*{\DTLnocite}[2]{%
\@ifundefined{dtl@aux@#1}{%
  \PackageError{databib}{multibib `#1' not defined}{%
  You need to define `#1' in \string\DTLmutlibibs}%
}{%
  \@bsphack
  \ifx\@onlypreamble\document
    \@for\@citeb:=#2\do{%
      \edef\@citeb{\expandafter\@firstofone\@citeb}%
      \if@filesw
        \expandafter\immediate
        \expandafter\write\csname dtl@aux@#1\endcsname{%
          \string\citation{\@citeb}}%
      \fi
      \@ifundefined{b@#1@\@citeb}{%
        \G@refundefinedtrue
```

```
        \@latex@warning{Citation `\@citeb ' undefined}}{}%
    }%
  \else
    \@latex@error{Cannot be used in preamble}\@eha
  \fi
  \@esphack
}%
}
\newcommand*{\DTLloadmbbl}[3]{%
\@ifundefined{dtl@aux@#1}{%
  \PackageError{databib}{multibib `#1' not defined}{%
  You need to define `#1' in \string\DTLmutlibibs}%
}{%
  \if@filesw
    \expandafter\immediate\expandafter
      \write\csname dtl@aux@#1\endcsname{\string\bibstyle{databib}}%
    \expandafter\immediate\expandafter
      \write\csname dtl@aux@#1\endcsname{\string\bibdata{#3}}%
  \fi
  \DTLnewdb{#2}%
  \edef\DTLBIBdbname{#2}%
  \@input@{#1.bbl}%
}%
}
\newcommand*{\DTLmbibliography}[3][\boolean{true}]{%
\begin{DTLthebibliography}[#1]{#3}%
\DTLforeachbibentry[#1]{#3}{%
\DTLmbibitem{#2} \DTLformatbibentry \DTLendbibitem
}%
\end{DTLthebibliography}%
}
```

30.12 Rollback v2.32 (datagidx-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datagidx}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{datatool}[=v2.32]
\RequirePackage{etoolbox}
\RequirePackage{xkeyval}
\RequirePackage{mfirstuc}
\RequirePackage{xfor}
\RequirePackage{multicol}
\RequirePackage{textcase}
\RequirePackage{afterpage}
\newcommand*{\datagidx@columns}{2}
\newcommand*{\DTLgidxSetColumns}[1]{%
  \DTLifint{#1}%
  {%
    \def\datagidx@columns{#1}%
  }%
```

```
  {%
    \PackageError{datagidx}%
    {Number of columns must be an integer}%
    {%
      You have requested `#1' columns, which can't be parsed as a
      number%
    }%
  }%
}
\newcounter{DTLgidxChildCount}
\def\theHDTLgidxChildCount{\Label.\arabic{DTLgidxChildCount}}
\newcommand*{\DTLgidxChildCountLabel}{\theDTLgidxChildCount) }
\newcommand*{\DTLgidxChildStyle}[1]{#1}
\newcommand*{\datagidx@setchildstyle}[1]{%
  \ifcase#1\relax
    \renewcommand*{\DTLgidxChildStyle}[1]{##1}%
  \or
    \renewcommand*{\DTLgidxChildStyle}[1]{%
      \DTLgidxChildCountLabel
    }%
  \fi
}
\newcommand{\datagidx@foreachchild}{%
  \datagidx@sort@foreachchild
}
\newcommand*{\datagidx@setchildsort}[1]{%
  \ifcase#1\relax
    \renewcommand*{\datagidx@foreachchild}{%
       \datagidx@sort@foreachchild
    }%
  \or
    \renewcommand*{\datagidx@foreachchild}{%
       \datagidx@unsort@foreachchild
    }%
  \fi
}
\newcommand*{\DTLgidxPostName}{ }
\newcommand*{\DTLgidxPostChildName}{\DTLgidxPostName}
\newcommand*{\DTLgidxNameCase}[1]{#1}
\newcommand*{\datagidx@setnamecase}[1]{%
  \ifcase#1\relax
    \renewcommand*{\DTLgidxNameCase}[1]{##1}%
  \or
    \let\DTLgidxNameCase\MakeTextUppercase
  \or
    \let\DTLgidxNameCase\MakeTextLowercase
  \or
    \let\DTLgidxNameCase\xmakefirstuc
  \or
    \let\DTLgidxNameCase\xcapitalisewords
```

```
  \fi
}
\newcommand*{\DTLgidxNameFont}[1]{\textnormal{#1}}
\newcommand*{\DTLgidxPostDescription}{}
\newcommand*{\datagidx@setpostdesc}[1]{%
   \ifcase#1\relax
     \renewcommand*{\DTLgidxPostDescription}{}%
   \or
     \renewcommand*{\DTLgidxPostDescription}{.}%
   \fi
 }
\newcommand*{\DTLgidxPreLocation}{\enspace}
\newcommand*{\datagidx@setprelocation}[1]{%
   \ifcase#1\relax
     \renewcommand*{\DTLgidxPreLocation}{}%
   \or
     \renewcommand*{\DTLgidxPreLocation}{\enspace}%
   \or
     \renewcommand*{\DTLgidxPreLocation}{ }%
   \or
     \renewcommand*{\DTLgidxPreLocation}{\dotfill}%
   \or
     \renewcommand*{\DTLgidxPreLocation}{\hfill}%
   \fi
 }
\newcommand*{\DTLgidxLocation}{\dtldolocationlist}
\newcommand*{\datagidx@setlocation}[1]{%
  \ifcase#1\relax
    \renewcommand*{\DTLgidxLocation}{}%
  \or
    \renewcommand*{\DTLgidxLocation}{\dtldolocationlist}%
  \or
    \renewcommand*{\DTLgidxLocation}{\dtldofirstlocation}%
  \fi
}
\newcommand*{\DTLgidxSee}{%
  \DTLifnull{\See}%
  {}%
  {%
    \DTLgidxPreLocation
    \DTLgidxFormatSee{\seename}{\See}%
  }%
}
\newcommand*{\DTLgidxSeeAlso}{%
  \DTLifnull{\SeeAlso}%
  {}%
  {%
     \DTLgidxFormatSeeAlso{\seealsoname}{\SeeAlso}%
  }%
}
```

```
\newcommand*{\DTLgidxChildrenSeeAlso}{%
  \DTLgidxChildren
  \DTLgidxSeeAlso
}
\newcommand*{\datagidx@setsee}[1]{%
  \ifcase#1\relax
    \renewcommand*{\DTLgidxSee}{%
      \DTLifnull{\See}{}%
      {%
        , \DTLgidxFormatSee{\seename}{\See}%
      }%
    }%
  \or
    \renewcommand*{\DTLgidxSee}{%
      \DTLifnull{\See}{}
      {%
        \space(\DTLgidxFormatSee{\seename}{\See})%
      }%
    }%
  \or
    \renewcommand*{\DTLgidxSee}{%
      \DTLifnull{\See}{}%
      {%
        . \DTLgidxFormatSee{\xmakefirstuc{\seename}}{\See}%
      }%
    }%
  \or
    \renewcommand*{\DTLgidxSee}{%
      \DTLifnull{\See}{}
      {%
        \space\DTLgidxFormatSee{\seename}{\See}%
      }%
    }%
  \or
    \renewcommand*{\DTLgidxSee}{%
      \DTLifnull{\See}{}
      {%
        \DTLgidxFormatSee{\seename}{\See}%
      }%
    }%
  \or
    \renewcommand*{\DTLgidxSee}{%
      \DTLifnull{\See}{}
      {%
        ; \DTLgidxFormatSee{\seename}{\See}%
      }%
    }%
  \or
    \renewcommand*{\DTLgidxSee}{%
      \DTLifnull{\See}{}
```

```
      {%
        \DTLgidxPreLocation\DTLgidxFormatSee{\seename}{\See}%
      }%
    }%
  \fi
}
\newcommand*{\DTLgidxSymDescSep}{\space}
\newlength\datagidxsymbolwidth
\newlength\datagidxlocationwidth
\newcommand{\DTLgidxFormatDesc}[1]{#1}
\newcommand*{\DTLgidxSymbolDescription}{%
  \DTLgidxSymbolDescLeft
  \DTLgidxSymbolDescRight
}
\newcommand*{\DTLgidxSymbolDescLeft}{%
  \ifdefempty{\Symbol}{}{(\Symbol)\DTLgidxSymDescSep}%
}
\newcommand*{\DTLgidxSymbolDescRight}{%
  \ifdefempty{\Description}{}%
  {%
    \DTLgidxFormatDesc{\Description}\DTLgidxPostDescription
  }%
}
\newif\if@datagidxsymbolleft
\@datagidxsymbollefttrue
\newcommand*{\datagidx@formatsymdesc}[1]{%
   \ifcase#1\relax
      \renewcommand*{\DTLgidxSymbolDescLeft}{%
        \ifdefempty{\Symbol}{}{\Symbol}%
      }%
      \renewcommand*{\DTLgidxSymbolDescRight}{}%
      \@datagidxsymbollefttrue
   \or
      \renewcommand*{\DTLgidxSymbolDescLeft}{%
        \ifdefempty{\Description}{}%
        {%
           \DTLgidxFormatDesc{\Description}\DTLgidxPostDescription
        }%
      }%
      \renewcommand*{\DTLgidxSymbolDescRight}{}%
      \@datagidxsymbolleftfalse
   \or
      \renewcommand*{\DTLgidxSymbolDescLeft}{%
        \ifdefempty{\Symbol}{}{(\Symbol)\DTLgidxSymDescSep}%
      }%
      \renewcommand*{\DTLgidxSymbolDescRight}{%
        \ifdefempty{\Description}{}%
        {%
          \DTLgidxFormatDesc{\Description}\DTLgidxPostDescription
        }%
```

```
      }%
      \@datagidxsymbollefttrue
    \or
      \renewcommand*{\DTLgidxSymbolDescLeft}{%
        \ifdefempty{\Description}{}%
        {%
          \DTLgidxFormatDesc{\Description}%
          \DTLgidxPostDescription\DTLgidxSymDescSep
        }%
      }%
      \renewcommand*{\DTLgidxSymbolDescRight}{%
        \ifdefempty{\Symbol}{}{(\Symbol)}%
      }%
      \@datagidxsymbolleftfalse
    \or
      \renewcommand*{\DTLgidxSymbolDescLeft}{%
        \ifdefempty{\Symbol}{}{\Symbol\DTLgidxSymDescSep}%
      }%
      \renewcommand*{\DTLgidxSymbolDescRight}{%
        \ifdefempty{\Description}{}%
        {%
          \DTLgidxFormatDesc{\Description}%
          \DTLgidxPostDescription
        }%
      }%
      \@datagidxsymbollefttrue
    \or
      \renewcommand*{\DTLgidxSymbolDescLeft}{%
        \ifdefempty{\Description}{}%
        {%
          \DTLgidxFormatDesc{\Description}%
          \DTLgidxPostDescription\DTLgidxSymDescSep
        }%
      }%
      \renewcommand*{\DTLgidxSymbolDescRight}{%
        \ifdefempty{\Symbol}{}{\Symbol}%
      }%
      \@datagidxsymbolleftfalse
    \fi
 }
\newcommand*{\DTLgidxSetCompositor}[1]{%
 \undef\datagidx@docomplist
 \DeclareListParser{\datagidx@docomplist}{#1}%
 \def\datagidx@compositor{#1}%
}
\DTLgidxSetCompositor{.}
\newcommand*{\datagidx@do@sort}{\datagidx@sort}
\newcommand*{\datagidx@optimize@sort}{%
  \ifdef\datagidx@do@optimize@sort
  {%
```

```
    \datagidx@sort
  }%
  {%
    \protected@write\@auxout{}{%
      \string\gdef\string\datagidx@do@optimize@sort{}%
    }%
    \global\let\@datagidx@dorerun@warn@sort\@data@rerun@warn@sort
  }%
}
\newif\if@datagidx@warn
\@datagidx@warntrue
\newcommand*\@datagidx@dorerun@warn{}
\AtEndDocument{\if@datagidx@warn\@datagidx@dorerun@warn\fi}
\newcommand*\@datagidx@dorerun@warn@sort{}
\AtEndDocument{\if@datagidx@warn\@datagidx@dorerun@warn@sort\fi}
\newcommand*\@data@rerun@warn@sort{%
  \PackageWarningNoLine{datagidx}{Rerun required to sort the
   index/glossary databases}%
}
\newcommand*\@data@rerun@warn{%
  \PackageWarningNoLine{datagidx}{Rerun required to ensure the
   index/glossary location lists are up-to-date}%
}
\newcommand*{\datagidx@do@highopt@optimize}{%
  \renewcommand*{\datagidx@do@sort}{%
    \ifcsdef{datagidx@do@highopt@sort@\DTLgidxCurrentdb}%
    {%
      \csuse{datagidx@do@highopt@sort@\DTLgidxCurrentdb}%
    }%
    {%
    }%
    \bgroup
    \def\dtl@saverawdbhook{%
      \let\db@col@id@w\@datagidx@db@col@id@w
      \def\DTLgidxName{\string\DTLgidxName\space}%
      \def\DTLgidxMac{\string\DTLgidxMac\space}%
      \def\DTLgidxRank{\string\DTLgidxRank\space}%
      \def\DTLgidxParen{\string\DTLgidxParen\space}%
      \def\DTLgidxParticle{\string\DTLgidxParticle\space}%
      \def\DTLgidxOffice{\string\DTLgidxOffice\space}%
      \def\DTLgidxSaint{\string\DTLgidxSaint\space}%
      \def\DTLgidxPlace{\string\DTLgidxPlace\space}%
      \def\DTLgidxIgnore{\string\DTLgidxIgnore\space}%
      \def\DTLgidxNameNum{\string\DTLgidxNameNum\space}%
      \def\DTLgidxSubject{\string\DTLgidxSubject\space}%
    }%
    \DTLsaverawdb{\DTLgidxCurrentdb}{\datagidxhighoptfilename\DTLgidxCurrentdb}%
    \egroup
  }%
  \def\newgidx{\datagidx@highopt@newgidx}%
```

```
  \def\newterm{\datagidx@highopt@newterm}%
}
\def\@datagidx@db@col@id@w#1\db@col@id@end@\db@col@elt@w#2\db@col@elt@end@\db@col@id@w#3
  \expandafter\@gobble\string\%^^J
  \string\db@col@id@w\space #1%
  \expandafter\@gobble\string\%^^J
  \string\db@col@id@end@\space
  \expandafter\@gobble\string\%^^J
  \string\db@col@elt@w\space
   \expandafter\ifnum\csname dtl@ci@\DTLgidxCurrentdb @Used\endcsname=#1\space
     0%
   \else
     \expandafter\ifnum\csname dtl@ci@\DTLgidxCurrentdb @Location\endcsname=#1\space
     \else
       \expandafter\ifnum\csname dtl@ci@\DTLgidxCurrentdb @CurrentLocation\endcsname=#1\
       \else
         \expandafter\ifnum\csname dtl@ci@\DTLgidxCurrentdb @Sort\endcsname=#1\space
           \protect#2%
         \else
         #2%
         \fi
       \fi
     \fi
   \fi
  \expandafter\@gobble\string\%^^J
  \string\db@col@elt@end@\space
  \expandafter\@gobble\string\%^^J
  \string\db@col@id@w\space #3%
  \expandafter\@gobble\string\%^^J
  \string\db@col@id@end@\space
}
\newcommand*{\datagidx@do@highopt@update}[1]{}
\newcommand*{\datagidxhighoptfilename}[1]{\jobname-#1.gidx}
\define@choicekey{datagidx.sty}{optimize}[\val\nr]%
 {off,low,high}[high]%
 {%
   \ifcase\nr\relax
    \renewcommand*{\datagidx@do@sort}{\datagidx@sort}
   \or
    \renewcommand*{\datagidx@do@sort}{\datagidx@optimize@sort}
   \or
    \datagidx@do@highopt@optimize
   \fi
 }
\define@choicekey{datagidx.sty}{nowarn}[\val\nr]{true,false}[true]%
{%
  \ifcase\nr\relax
    \@datagidx@warnfalse
  \or
    \@datagidx@warntrue
```

```
  \fi
}
\define@choicekey{datatool.sty}{utf8}{true,false}[true]{%
  \setbool{@dtl@utf8}{#1}%
}
\define@key{datagidx.sty}{columns}%
{%
  \DTLgidxSetColumns{#1}%
}
\define@choicekey{datagidx.sty}{child}[\val\nr]%
 {named,noname}%
 {%
   \datagidx@setchildstyle\nr
 }
\define@choicekey{datagidx.sty}{namecase}[\val\nr]%
 {nochange,uc,lc,firstuc,capitalise}%
 {%
   \datagidx@setnamecase\nr
 }
\define@key{datagidx.sty}{namefont}%
{%
  \renewcommand*{\DTLgidxNameFont}[1]{{#1{##1}}}%
}
\define@key{datagidx.sty}{postname}
{%
  \renewcommand*{\DTLgidxPostName}{#1}%
}
\define@choicekey{datagidx.sty}{postdesc}[\val\nr]%
 {none,dot}%
 {%
   \datagidx@setpostdesc\nr
 }
\define@choicekey{datagidx.sty}{prelocation}[\val\nr]%
 {none,enspace,space,dotfill,hfill}%
 {%
   \datagidx@setprelocation\nr
 }
\define@choicekey{datagidx.sty}{location}[\val\nr]%
 {hide,list,first}%
 {\datagidx@setlocation\nr}
\define@choicekey{datagidx.sty}{see}[\val\nr]%
  {comma,brackets,dot,space,nosep,semicolon,location}%
  {\datagidx@setsee\nr}
\define@choicekey{datagidx.sty}{symboldesc}[\val\nr]%
 {symbol,desc,(symbol) desc,desc (symbol),symbol desc,desc symbol}%
 {\datagidx@formatsymdesc\nr}
\define@key{datagidx.sty}{compositor}%
{%
  \DTLgidxSetCompositor{#1}%
}%
```

```
\DeclareOptionX{final}{%
  \let\datagidxshowifdraft\@gobble
}
\let\datagidxshowifdraft\@gobble
\DeclareOptionX{draft}{%
  \let\datagidxshowifdraft\@firstofone
}
\define@choicekey{datagidx.sty}{verbose}[\val\nr]%
 {true,false}[true]%
 {%
   \csuse{dtlverbose\val}%
 }
\ProcessOptionsX
\DTLnewdb{datagidx}
\providecommand*{\seename}{see}
\providecommand*{\seealsoname}{see also}
\newcommand*{\DTLgidxSeeTagFont}[1]{\emph{#1}}
\newcommand*{\DTLgidxFormatSee}[2]{%
  \DTLgidxSeeTagFont{#1} \DTLgidxSeeList{#2}%
}
\newcommand*{\DTLgidxFormatSeeAlso}[2]{%
  \datagidxdoseealso
  {%
    \DTLgidxSeeTagFont{#1} \DTLgidxSeeList{#2}%
  }%
}
\newcommand*{\datagidxdoseealso}[1]{%
  \datagidxseealsostart
   #1%
  \datagidxseealsoend
}
\newcommand*{\DTLgidxSeeList}[1]{%
  \def\datagidx@sep{}%
  \@for\dtl@thislabel:=#1\do
  {%
    \ifx\@xfor@nextelement\@nnil
      \ifdefempty{\datagidx@sep}%
      {%
      }%
      {%
         \DTLidxSeeLastSep
      }%
    \else
      \datagidx@sep
      \let\datagidx@sep\DTLidxSeeSep
    \fi
    \DTLidxFormatSeeItem{\dtl@thislabel}%
  }%
}
\newcommand*{\DTLidxFormatSeeItem}[1]{%
```

```
  \DTLgidxFetchEntry{\datagidx@value}{#1}{Name}%
  \datagidxlink{#1}%
  {%
    \datagidx@value
  }%
}
\newcommand*{\DTLidxSeeSep}{, }
\newcommand*{\DTLidxSeeLastSep}{ \& }
\newcommand*{\DTLgidxDoSeeOrLocation}{%
  \DTLifnull\See
  {%
     \ifdefempty\Location
     {%
     }%
     {%
       \DTLgidxPreLocation
       \DTLgidxLocation
     }%
  }%
  {%
     \DTLgidxSee
  }%
}
\newcommand*{\datagidx@sortchildren}{%
  \def\datagidx@sortedlist{}%
  \@for\Label:=\Children\do
  {%
    \edef\do@getrow{%
      \noexpand\dtlgetrowforvalue
      {\DTLgidxCurrentdb}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Label}}%
      {\Label}%
    }%
    \do@getrow
    \ifdefempty\datagidx@sortedlist
    {%
      \edef\datagidx@newsortedlist{{\number\dtlrownum}{\Label}}%
    }%
    {%
      \def\datagidx@newsortedlist{}%
      \@for\@datagidx@thisval:=\datagidx@sortedlist\do
      {%
        \edef\datagidx@thisidx{\expandafter\@firstoftwo\@datagidx@thisval}%
        \ifnum\datagidx@thisidx>\dtlrownum\relax
          \ifdefempty\datagidx@newsortedlist
          {%
            \eappto\datagidx@newsortedlist
            {%
              {\number\dtlrownum}{\Label},\@datagidx@thisval
            }%
```

```
          }%
          {%
            \eappto\datagidx@newsortedlist
            {%
              ,{\number\dtlrownum}{\Label},\@datagidx@thisval
            }%
          }%
          \@endfortrue
        \else
          \ifdefempty\datagidx@newsortedlist
          {%
            \edef\datagidx@newsortedlist{%
              \@datagidx@thisval
            }%
          }%
          {%
            \eappto\datagidx@newsortedlist
            {%
              ,\@datagidx@thisval
            }%
          }%
        \fi
      }%
      \if@endfor
        \ifdefempty\@forremainder
        {%
        }%
        {%
          \eappto\datagidx@newsortedlist{,\@forremainder}%
        }%
      \else
        \ifdefempty\datagidx@newsortedlist
        {%
          \edef\datagidx@newsortedlist{{\number\dtlrownum}{\Label}}%
        }%
        {%
          \eappto\datagidx@newsortedlist{,{\number\dtlrownum}{\Label}}%
        }%
      \fi
    }%
    \let\datagidx@sortedlist\datagidx@newsortedlist
    \@endforfalse
  }%
}
\newcommand{\datagidx@sort@foreachchild}[1]{%
  \datagidx@sortchildren
  \@for\@datagidx@thisval:=\datagidx@sortedlist\do
  {%
    \edef\Label{\expandafter\@secondoftwo\@datagidx@thisval}%
    #1%
```

```
  }%
}
\newcommand{\datagidx@unsort@foreachchild}[1]{%
  \@for\Label:=\Children\do
  {%
    #1%
  }%
}
\newcommand*{\DTLgidxChildren}{%
  \bgroup
    \DTLifnull\Children
    {}%
    {%
      \advance\datagidx@level by 1\relax
      \datagidxchildstart
      \let\Parent\Label
      \datagidx@foreachchild
      {%
        \edef\do@getrow{%
          \noexpand\dtlgetrowforvalue
          {\DTLgidxCurrentdb}%
          {\dtlcolumnindex{\DTLgidxCurrentdb}{Label}}%
          {\Label}%
        }%
        \do@getrow
        \dtlgetentryfromcurrentrow
          {\Location}%
          {\dtlcolumnindex{\DTLgidxCurrentdb}{Location}}%
        \dtlgetentryfromcurrentrow
          {\See}%
          {\dtlcolumnindex{\DTLgidxCurrentdb}{See}}%
        \dtlgetentryfromcurrentrow
          {\SeeAlso}%
          {\dtlcolumnindex{\DTLgidxCurrentdb}{SeeAlso}}%
        \DTLifnull\Location
        {%
          \DTLifnull\See
          {%
            \DTLifnull\SeeAlso
            {}%
            {%
              \datagidx@displaychild
            }%
          }%
          {%
            \datagidx@displaychild
          }%
        }%
        {%
           \datagidx@displaychild
```

```
        }%
      }%
      \datagidxchildend
    }%
  \egroup
}
\newcommand*{\datagidxgetchildfields}{%
  \dtlgetentryfromcurrentrow
    {\Name}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Name}}%
  \dtlgetentryfromcurrentrow
    {\Description}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Description}}%
  \dtlgetentryfromcurrentrow
    {\Symbol}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Symbol}}%
  \dtlgetentryfromcurrentrow
    {\Long}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Long}}%
  \dtlgetentryfromcurrentrow
    {\Short}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Short}}%
  \dtlgetentryfromcurrentrow
    {\Text}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Text}}%
  \dtlgetentryfromcurrentrow
    {\Plural}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Plural}}%
  \dtlgetentryfromcurrentrow
    {\Short}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Used}}%
  \dtlgetentryfromcurrentrow
    {\Children}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Child}}%
}
\newcommand*{\datagidx@displaychild}{%
  \datagidxgetchildfields
  \datagidxchilditem
}
\ifdef{\chapter}
{%
  \newcommand*{\datagidx@heading}{\chapter*}
}%
{%
  \newcommand*{\datagidx@heading}{\section*}
}
\let\DTLgidxNoHeading\@gobble
\newcommand*{\datagidx@postheading}{}
\newcommand*{\datagidx@multicols}{multicols}
\newcommand*{\datagidx@sort}{%
```

```
  \dtlsort{Sort,FirstId}{\DTLgidxCurrentdb}{\dtlwordindexcompare}%
}
\providecommand{\@idxitem}{\par\hangindent 40\p@}
\newcommand*{\datagidxstart}%
{%
  \bgroup
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{0pt plus 0.3pt}%
  \let\item\@idxitem
}
\newcommand*{\datagidxend}{\egroup}
\newcommand*{\@datagidxtarget}[2]{%
  \ifdef\hypertarget
  {%
    \bgroup
      \let\glsadd\@gobble
      \settoheight\dimen@{#2}%
      \raisebox{\dimen@}{\hypertarget{#1}{}}%
    \egroup
  }%
  {%
  }%
  #2%
}
\newcommand*{\datagidxtarget}{\@datagidxtarget}
\newcommand*{\@datagidxlink}[2]{%
  \ifdef\hyperlink
  {%
    \hyperlink{#1}{#2}%
  }%
  {%
    #2%
  }%
}
\newcommand*{\datagidxlink}{\@datagidxlink}
\newcommand*{\DTLgidxEnableHyper}{%
  \let\datagidxtarget\@datagidxtarget
  \let\datagidxlink\@datagidxlink
}
\newcommand*{\DTLgidxDisableHyper}{%
  \let\datagidxtarget\@secondoftwo
  \let\datagidxlink\@secondoftwo
}
\newcommand*{\datagidxgroupsep}{}
\newcommand*{\datagidxgroupheader}{}
\newcommand*{\datagidxitem}{}%
\newcommand*{\datagidxchildstart}{}
\newcommand*{\datagidxchildend}{}
\newcommand*{\datagidxchilditem}{}%
\newcommand*{\datagidxseealsostart}{}
```

```
\newcommand*{\datagidxseealsoend}{}
\newcommand*{\datagidx@doifsymlocwidth}[3]{%
  \setlength{\dtl@tmplength}{\linewidth}%
  \addtolength{\dtl@tmplength}{-#1}%
  \settowidth{\dimen@}{#2}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \addtolength{\dtl@tmplength}{-\datagidxsymbolwidth}%
  \addtolength{\dtl@tmplength}{-\datagidxlocationwidth}%
  \settowidth{\dimen@}{\DTLgidxPreLocation}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \settowidth{\dimen@}{\DTLgidxSymDescSep}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \if@datagidxsymbolleft
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\dtl@tmplength}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescRight
    \end{minipage}%
  \else
    \begin{minipage}[t]{\dtl@tmplength}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescRight
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
    \end{minipage}%
  \fi
  \DTLgidxPreLocation
  \begin{minipage}[t]{\datagidxlocationwidth}%
    \datagidxlocalign
    \let\DTLgidxPreLocation\@empty
    #3%
  \end{minipage}%
}
\newcommand*{\datagidx@doiflocwidth}[3]{%
  \setlength{\dtl@tmplength}{\linewidth}%
  \addtolength{\dtl@tmplength}{-#1}%
  \settowidth{\dimen@}{#2}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \addtolength{\dtl@tmplength}{-\datagidxlocationwidth}%
  \settowidth{\dimen@}{\DTLgidxPreLocation}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
```

```
  \begin{minipage}[t]{\dtl@tmplength}%
    \DTLgidxSymbolDescription
  \end{minipage}%
  \DTLgidxPreLocation
  \begin{minipage}[t]{\datagidxlocationwidth}%
    \datagidxlocalign
    \let\DTLgidxPreLocation\@empty
    #3%
  \end{minipage}%
}
\newcommand*{\datagidx@doifsymwidth}[3]{%
  \setlength{\dtl@tmplength}{\linewidth}%
  \addtolength{\dtl@tmplength}{-#1}%
  \settowidth{\dimen@}{#2}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \addtolength{\dtl@tmplength}{-\datagidxsymbolwidth}%
  \settowidth{\dimen@}{\DTLgidxSymDescSep}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \if@datagidxsymbolleft
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\dtl@tmplength}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescRight
      #3%
    \end{minipage}%
  \else
    \begin{minipage}[t]{\dtl@tmplength}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescRight
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
      #3%
    \end{minipage}%
  \fi
}
\newcommand*{\datagidxlocalign}{\raggedleft}
\newcommand*{\datagidxsymalign}{\centering}
\newcommand*{\datagidxsetstyle}[1]{%
  \ifcsdef{datagidx@style@#1}%
  {%
    \csuse{datagidx@style@#1}%
```

```
  }%
  {%
    \PackageError{datagidx}{Unknown style `#1'}{}%
  }%
}
\newcommand*{\datagidx@style@index}{%
  \renewcommand*{\datagidxstart}%
  {%
    \bgroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \raggedright
    \let\item\@idxitem
    \ifdim\datagidxsymbolwidth>0pt\relax
      \ifdim\datagidxlocationwidth>0pt\relax
        \def\datagidx@item@body{%
          \datagidx@doifsymlocwidth{0pt}%
           {\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
           {%
             \DTLgidxDoSeeOrLocation
           }%
         }%
      \else
        \def\datagidx@item@body{%
          \datagidx@doiflocwidth{0pt}%
           {\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
           {%
             \DTLgidxDoSeeOrLocation
           }%
         }%
      \fi
    \else
      \ifdim\datagidxlocationwidth>0pt\relax
        \def\datagidx@item@body{%
          \datagidx@doiflocwidth{0pt}%
           {\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
           {%
             \DTLgidxDoSeeOrLocation
           }%
         }%
      \else
        \def\datagidx@item@body{%
          \DTLgidxSymbolDescription
          \DTLgidxDoSeeOrLocation
        }%
      \fi
    \fi
  }%
  \renewcommand*{\datagidxend}{\egroup}%
  \renewcommand*{\datagidxgroupsep}{\ifdatagidxshowgroups\indexspace\fi}%
```

```
\renewcommand{\datagidxgroupheader}{%
  \ifdatagidxshowgroups
    \item
     \makebox[\linewidth]%
     {%
       \textbf{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
     }%
     \DTLpar\nobreak\@afterheading
  \fi
}%
\renewcommand*{\datagidxitem}{%
  \ifdefempty\datagidxprevgroup
  {%
    \datagidxgroupheader
  }%
  {%
    \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
    {%
    }%
    {%
      \datagidxgroupsep
      \datagidxgroupheader
    }%
  }%
  \item
  \datagidxtarget{\Label}%
  {%
    \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
  }%
  \DTLgidxPostName
  \datagidx@item@body
  \DTLgidxChildrenSeeAlso
}%
\renewcommand*{\datagidxchildstart}%
{%
  \bgroup
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{0pt plus 0.3pt}%
  \let\item\@idxitem
}%
\renewcommand*{\datagidxchildend}{\egroup}%
\renewcommand*{\datagidxchilditem}{%
  \setlength{\dimen@}{\datagidxindent}%
  \multiply\dimen@ by \datagidx@level\relax
  \@idxitem\hspace*{\dimen@}%
  \refstepcounter{DTLgidxChildCount}%
  \datagidxtarget{\Label}%
  {%
    \DTLgidxChildStyle
    {%
```

```
        \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
        \DTLgidxPostChildName
      }%
    }%
    \DTLgidxSymbolDescription
    \DTLgidxDoSeeOrLocation
    \DTLgidxChildrenSeeAlso
  }%
  \renewcommand*{\datagidxseealsostart}%
  {%
    \bgroup
      \setlength{\parindent}{0pt}%
      \setlength{\parskip}{0pt plus 0.3pt}%
      \setlength{\dimen@}{\datagidxindent}%
      \advance\datagidx@level by 1\relax
      \multiply\dimen@ by \datagidx@level\relax
      \@idxitem\hspace*{\dimen@}%
  }%
  \renewcommand{\datagidxseealsoend}{\egroup}%
}
\datagidx@style@index
\newcommand*{\datagidx@style@indexalign}{%
  \renewcommand*{\datagidxstart}%
  {%
    \bgroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \setlength{\datagidxnamewidth}{0pt}%
    \DTLforeach*{\DTLgidxCurrentdb}%
      {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
       \Parent=Parent}%
    {%
       \DTLifnull{\Parent}%
       {%
         \datagidx@doifdisplayed
         {%
           \settowidth{\dimen@}{\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
           \ifdim\dimen@>\datagidxnamewidth\relax
              \datagidxnamewidth=\dimen@\relax
           \fi
         }%
       }%
       {}%
    }%
    \settowidth{\dimen@}{\DTLgidxPostName}%
    \addtolength{\datagidxnamewidth}{\dimen@}%
    \setlength{\datagidxdescwidth}{\linewidth}%
    \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
    \ifdim\datagidxsymbolwidth>0pt\relax
      \addtolength{\datagidxdescwidth}{-\datagidxsymbolwidth}%
```

```
  \settowidth{\dimen@}{\DTLgidxSymDescSep}%
  \addtolength{\datagidxdescwidth}{-\dimen@}%
\fi
\ifdim\datagidxlocationwidth>0pt\relax
  \addtolength{\datagidxdescwidth}{-\datagidxlocationwidth}%
  \settowidth{\dimen@}{\DTLgidxPreLocation}%
  \addtolength{\datagidxdescwidth}{-\dimen@}%
\fi
\ifdim\datagidxsymbolwidth>0pt\relax
  \ifdim\datagidxlocationwidth>0pt\relax
    \if@datagidxsymbolleft
      \def\datagidx@item@body{%
        \begin{minipage}[t]{\datagidxsymbolwidth}%
          \datagidxsymalign
          \let\DTLgidxSymDescSep\@empty
          \DTLgidxSymbolDescLeft
        \end{minipage}%
        \DTLgidxSymDescSep
        \begin{minipage}[t]{\datagidxdescwidth}%
          \let\DTLgidxSymDescSep\@empty
          \setlength{\parskip}{0pt plus 0.3pt}%
          \DTLgidxSymbolDescRight
        \end{minipage}%
        \DTLgidxPreLocation
        \begin{minipage}[t]{\datagidxlocationwidth}%
          \datagidxlocalign
          \let\DTLgidxPreLocation\@empty
          \DTLgidxDoSeeOrLocation
        \end{minipage}%
      }%
    \else
      \def\datagidx@item@body{%
        \begin{minipage}[t]{\datagidxdescwidth}%
          \let\DTLgidxSymDescSep\@empty
          \DTLgidxSymbolDescLeft
        \end{minipage}%
        \DTLgidxSymDescSep
        \begin{minipage}[t]{\datagidxsymbolwidth}%
          \datagidxsymalign
          \let\DTLgidxSymDescSep\@empty
          \setlength{\parskip}{0pt plus 0.3pt}%
          \DTLgidxSymbolDescRight
        \end{minipage}%
        \DTLgidxPreLocation
        \begin{minipage}[t]{\datagidxlocationwidth}%
          \datagidxlocalign
          \let\DTLgidxPreLocation\@empty
          \DTLgidxDoSeeOrLocation
        \end{minipage}%
      }%
```

```
    \fi
  \else
    \if@datagidxsymbolleft
      \def\datagidx@item@body{%
        \begin{minipage}[t]{\datagidxsymbolwidth}%
          \datagidxsymalign
          \let\DTLgidxSymDescSep\@empty
          \DTLgidxSymbolDescLeft
        \end{minipage}%
        \DTLgidxSymDescSep
        \begin{minipage}[t]{\datagidxdescwidth}%
          \let\DTLgidxSymDescSep\@empty
          \setlength{\parskip}{0pt plus 0.3pt}%
          \DTLgidxSymbolDescRight
          \DTLgidxDoSeeOrLocation
        \end{minipage}%
      }%
    \else
      \def\datagidx@item@body{%
        \begin{minipage}[t]{\datagidxdescwidth}%
          \let\DTLgidxSymDescSep\@empty
          \DTLgidxSymbolDescLeft
        \end{minipage}%
        \DTLgidxSymDescSep
        \begin{minipage}[t]{\datagidxsymbolwidth}%
          \datagidxsymalign
          \let\DTLgidxSymDescSep\@empty
          \setlength{\parskip}{0pt plus 0.3pt}%
          \DTLgidxSymbolDescRight
          \DTLgidxDoSeeOrLocation
        \end{minipage}%
      }%
    \fi
  \fi
\else
  \ifdim\datagidxlocationwidth>0pt\relax
    \def\datagidx@item@body{%
      \begin{minipage}[t]{\datagidxdescwidth}%
       \setlength{\parskip}{0pt plus 0.3pt}%
       \DTLgidxSymbolDescription
      \end{minipage}%
      \DTLgidxPreLocation
      \begin{minipage}[t]{\datagidxlocationwidth}%
      \datagidxlocalign
       \let\DTLgidxPreLocation\@empty
       \DTLgidxDoSeeOrLocation
     }%
  \else
    \def\datagidx@item@body{%
      \begin{minipage}[t]{\datagidxdescwidth}%
```

```
        \setlength{\parskip}{0pt plus 0.3pt}%
        \DTLgidxSymbolDescription
        \DTLgidxDoSeeOrLocation
       \end{minipage}%
      }%
    \fi
  \fi
}%
\renewcommand*{\datagidxend}{\egroup}%
\renewcommand*{\datagidxgroupsep}{}%
\renewcommand*{\datagidxgroupheader}{}%
\renewcommand*{\datagidxitem}{%
  \ifdefempty\datagidxprevgroup
  {%
    \datagidxgroupheader
  }%
  {%
    \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
    {%
    }%
    {%
      \datagidxgroupsep
      \datagidxgroupheader
    }%
  }%
  \hangindent0pt\relax
  \parindent0pt\relax
  \makebox[\datagidxnamewidth][l]%
  {%
    \datagidxtarget{\Label}%
    {%
      \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
      \DTLgidxPostName
    }%
  }%
  \datagidx@item@body
  \par
  \DTLgidxChildrenSeeAlso
  \par
}%
\renewcommand*{\datagidxchildstart}%
{%
  \bgroup
  \setlength{\dimen@}{\datagidxindent}%
  \multiply\dimen@ by \datagidx@level\relax
  \setlength{\dtl@tmplength}{\linewidth}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{0pt plus 0.3pt}%
  \edef\item{\noexpand\parshape=1 \the\dimen@ \the\dtl@tmplength}%
```

```
  \setlength{\datagidxnamewidth}{0pt}%
  \DTLforeach*{\DTLgidxCurrentdb}%
    {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
     \Parent=Parent}%
  {%
     \DTLifnull{\Parent}%
     {%
       \datagidx@doifdisplayed
       {%
         \settowidth{\dimen@}%
         {%
           \DTLgidxChildStyle
           {%
             \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
           }%
         }%
         \ifdim\dimen@>\datagidxnamewidth\relax
            \datagidxnamewidth=\dimen@\relax
         \fi
       }%
     }%
     {}%
  }%
  \settowidth{\dimen@}{\DTLgidxChildStyle\DTLgidxPostChildName}%
  \addtolength{\datagidxnamewidth}{\dimen@}%
  \setlength{\datagidxdescwidth}{\dtl@tmplength}%
  \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
}%
\renewcommand{\datagidxchildend}{\egroup}%
\renewcommand*{\datagidxchilditem}{%
  \item
  \refstepcounter{DTLgidxChildCount}%
  \makebox[\datagidxnamewidth][l]%
  {%
    \datagidxtarget{\Label}%
    {%
      \DTLgidxChildStyle
      {%
        \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
        \DTLgidxPostChildName
      }%
    }%
  }%
  \begin{minipage}[t]{\datagidxdescwidth}%
   \setlength{\parskip}{0pt plus 0.3pt}%
   \DTLgidxSymbolDescription
   \DTLgidxDoSeeOrLocation
   \DTLgidxChildrenSeeAlso
  \end{minipage}%
  \par
```

```
  }%
}
\newlength\datagidxindent
\setlength\datagidxindent{10\p@}
\newlength\datagidxnamewidth
\newlength\datagidxdescwidth
\newcommand*{\datagidx@style@align}{%
  \renewcommand*{\datagidxstart}%
  {%
    \bgroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \setlength{\datagidxnamewidth}{0pt}%
    \DTLforeach*{\DTLgidxCurrentdb}%
      {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
       \Parent=Parent}%
    {%
       \DTLifnull{\Parent}%
       {%
         \datagidx@doifdisplayed
         {%
           \settowidth{\dimen@}{\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
           \ifdim\dimen@>\datagidxnamewidth\relax
              \datagidxnamewidth=\dimen@\relax
           \fi
         }%
       }%
       {}%
    }%
    \settowidth{\dimen@}{\DTLgidxPostName}%
    \addtolength{\datagidxnamewidth}{\dimen@}%
    \setlength{\datagidxdescwidth}{\linewidth}%
    \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
    \ifdim\datagidxsymbolwidth>0pt\relax
      \addtolength{\datagidxdescwidth}{-\datagidxsymbolwidth}%
      \settowidth{\dimen@}{\DTLgidxSymDescSep}%
      \addtolength{\datagidxdescwidth}{-\dimen@}%
    \fi
    \ifdim\datagidxlocationwidth>0pt\relax
      \addtolength{\datagidxdescwidth}{-\datagidxlocationwidth}%
      \settowidth{\dimen@}{\DTLgidxPreLocation}%
      \addtolength{\datagidxdescwidth}{-\dimen@}%
    \fi
    \ifdim\datagidxsymbolwidth>0pt\relax
      \ifdim\datagidxlocationwidth>0pt\relax
        \if@datagidxsymbolleft
          \def\datagidx@item@body{%
            \begin{minipage}[t]{\datagidxsymbolwidth}%
              \datagidxsymalign
              \let\DTLgidxSymDescSep\@empty
```

```
      \DTLgidxSymbolDescLeft
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\datagidxdescwidth}%
      \let\DTLgidxSymDescSep\@empty
      \setlength{\parskip}{0pt plus 0.3pt}%
      \DTLgidxSymbolDescRight
    \end{minipage}%
    \DTLgidxPreLocation
    \begin{minipage}[t]{\datagidxlocationwidth}%
      \datagidxlocalign
      \let\DTLgidxPreLocation\@empty
      \DTLgidxDoSeeOrLocation
      \DTLgidxChildrenSeeAlso
    \end{minipage}%
   }%
  \else
   \def\datagidx@item@body{%
    \begin{minipage}[t]{\datagidxdescwidth}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \setlength{\parskip}{0pt plus 0.3pt}%
      \DTLgidxSymbolDescRight
    \end{minipage}%
    \DTLgidxPreLocation
    \begin{minipage}[t]{\datagidxlocationwidth}%
      \datagidxlocalign
      \let\DTLgidxPreLocation\@empty
      \DTLgidxDoSeeOrLocation
      \DTLgidxChildrenSeeAlso
    \end{minipage}%
   }%
  \fi
\else
  \if@datagidxsymbolleft
   \def\datagidx@item@body{%
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\datagidxdescwidth}%
      \let\DTLgidxSymDescSep\@empty
      \setlength{\parskip}{0pt plus 0.3pt}%
```

```
            \DTLgidxSymbolDescRight
            \DTLgidxDoSeeOrLocation
            \DTLgidxChildrenSeeAlso
          \end{minipage}%
        }%
       \else
         \def\datagidx@item@body{%
           \begin{minipage}[t]{\datagidxdescwidth}%
             \let\DTLgidxSymDescSep\@empty
             \DTLgidxSymbolDescLeft
           \end{minipage}%
           \DTLgidxSymDescSep
           \begin{minipage}[t]{\datagidxsymbolwidth}%
             \datagidxsymalign
             \let\DTLgidxSymDescSep\@empty
             \setlength{\parskip}{0pt plus 0.3pt}%
             \DTLgidxSymbolDescRight
             \DTLgidxDoSeeOrLocation
             \DTLgidxChildrenSeeAlso
          \end{minipage}%
        }%
       \fi
    \fi
  \else
    \ifdim\datagidxlocationwidth>0pt\relax
      \def\datagidx@item@body{%
        \begin{minipage}[t]{\datagidxdescwidth}%
         \setlength{\parskip}{0pt plus 0.3pt}%
         \DTLgidxSymbolDescription
        \end{minipage}%
        \DTLgidxPreLocation
        \begin{minipage}[t]{\datagidxlocationwidth}%
        \datagidxlocalign
         \let\DTLgidxPreLocation\@empty
         \DTLgidxDoSeeOrLocation
         \DTLgidxChildrenSeeAlso
        \end{minipage}%
       }%
    \else
      \def\datagidx@item@body{%
        \begin{minipage}[t]{\datagidxdescwidth}%
         \setlength{\parskip}{0pt plus 0.3pt}%
         \DTLgidxSymbolDescription
         \DTLgidxDoSeeOrLocation
         \DTLgidxChildrenSeeAlso
        \end{minipage}%
       }%
    \fi
  \fi
}%
```

```
\renewcommand*{\datagidxend}{\egroup}%
\renewcommand*{\datagidxgroupsep}{\ifdatagidxshowgroups\indexspace\fi}%
\renewcommand{\datagidxgroupheader}{%
  \ifdatagidxshowgroups
    \item
     \makebox[\linewidth]%
     {%
       \textbf{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
     }%
     \DTLpar\nobreak\@afterheading
  \fi
}%
\renewcommand*{\datagidxitem}{%
  \ifdefempty\datagidxprevgroup
  {%
    \datagidxgroupheader
  }%
  {%
    \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
    {%
    }%
    {%
      \datagidxgroupsep
      \datagidxgroupheader
    }%
  }%
  \hangindent0pt\relax
  \parindent0pt\relax
  \makebox[\datagidxnamewidth][l]%
  {%
    \datagidxtarget{\Label}%
    {%
      \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
      \DTLgidxPostName
    }%
  }%
   \datagidx@item@body
  \par
}%
\renewcommand*{\datagidxchildstart}%
{%
  \bgroup
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{0pt plus 0.3pt}%
  \setlength{\datagidxnamewidth}{0pt}%
  \DTLforeach*{\DTLgidxCurrentdb}%
    {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
     \Parent=Parent}%
  {%
     \DTLifnull{\Parent}%
```

```
      {%
        \datagidx@doifdisplayed
        {%
          \settowidth{\dimen@}%
          {%
            \DTLgidxChildStyle
            {%
              \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
            }%
          }%
          \ifdim\dimen@>\datagidxnamewidth\relax
             \datagidxnamewidth=\dimen@\relax
          \fi
        }%
      }%
      {}%
    }%
    \settowidth{\dimen@}{\DTLgidxChildStyle\DTLgidxPostChildName}%
    \addtolength{\datagidxnamewidth}{\dimen@}%
    \setlength{\datagidxdescwidth}{\linewidth}%
    \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
  }%
  \renewcommand{\datagidxchildend}{\egroup}%
  \renewcommand*{\datagidxchilditem}{%
    \hangindent0pt\relax
    \parindent0pt\relax
    \refstepcounter{DTLgidxChildCount}%
    \makebox[\datagidxnamewidth][l]%
    {%
      \datagidxtarget{\Label}%
      {%
        \DTLgidxChildStyle
        {%
          \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
          \DTLgidxPostChildName
        }%
      }%
    }%
    \begin{minipage}[t]{\datagidxdescwidth}%
     \setlength{\parskip}{0pt plus 0.3pt}%
     \DTLgidxSymbolDescription
     \DTLgidxDoSeeOrLocation
     \DTLgidxChildrenSeeAlso
    \end{minipage}%
    \par
  }%
}
\newcommand*{\datagidx@style@gloss}{%
  \renewcommand*{\datagidxstart}%
  {%
```

```
  \bgroup
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{0pt plus 0.3pt}%
  \setlength{\datagidxnamewidth}{0pt}%
  \DTLforeach*{\DTLgidxCurrentdb}%
    {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
     \Parent=Parent}%
  {%
     \DTLifnull{\Parent}%
     {%
       \datagidx@doifdisplayed
       {%
         \settowidth{\dimen@}{\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
         \ifdim\dimen@>\datagidxnamewidth\relax
            \datagidxnamewidth=\dimen@\relax
         \fi
       }%
     }%
     {}%
  }%
  \settowidth{\dimen@}{\DTLgidxPostName}%
  \addtolength{\datagidxnamewidth}{\dimen@}%
  \setlength{\datagidxdescwidth}{\linewidth}%
  \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
}%
\renewcommand*{\datagidxend}{\egroup}%
\renewcommand*{\datagidxgroupsep}{\ifdatagidxshowgroups\indexspace\fi}%
\renewcommand{\datagidxgroupheader}{%
  \ifdatagidxshowgroups
    \item
     \makebox[\linewidth]%
     {%
       \textbf{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
     }%
     \DTLpar\nobreak\@afterheading
  \fi
}%
\renewcommand*{\datagidxitem}{%
  \ifdefempty\datagidxprevgroup
  {%
    \datagidxgroupheader
  }%
  {%
    \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
    {%
    }%
    {%
      \datagidxgroupsep
      \datagidxgroupheader
    }%
```

```
  }%
  \hangindent0pt\relax
  \parindent0pt\relax
  \makebox[\datagidxnamewidth][l]%
  {%
    \datagidxtarget{\Label}%
    {%
      \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
      \DTLgidxPostName
    }%
  }%
  \begin{minipage}[t]{\datagidxdescwidth}%
   \setlength{\parskip}{0pt plus 0.3pt}%
   \@tempswatrue
   \ifdefempty{\Description}%
   {%
     \ifdefempty{\Symbol}%
     {%
       \ifdefempty{\Location}{\@tempswafalse}{}%
     }%
     {}%
   }%
   {}%
   \if@tempswa
     \DTLgidxSymbolDescription
     \DTLgidxDoSeeOrLocation
   \else
     \mbox{}%
   \fi
   \DTLgidxChildrenSeeAlso
  \end{minipage}%
  \par
}%
\renewcommand*{\datagidxchildstart}%
{%
  \bgroup
  \def\datagidx@childsep{}%
  \setcounter{DTLgidxChildCount}{0}%
}%
\renewcommand{\datagidxchildend}{\DTLgidxPostChild\egroup}%
\renewcommand*{\datagidxchilditem}{%
  \datagidx@childsep
  \refstepcounter{DTLgidxChildCount}%
  \datagidxtarget{\Label}%
  {%
    \DTLgidxChildStyle
    {%
      \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
      \DTLgidxPostChildName
    }%
```

```
    }%
    \DTLgidxSymbolDescription
    \DTLgidxDoSeeOrLocation
    \DTLgidxChildrenSeeAlso
    \let\datagidx@childsep\DTLgidxChildSep
  }%
}
\newcommand*{\DTLgidxChildSep}{ }
\newcommand*{\DTLgidxPostChild}{}
\ifdef\chapter
{%
  \newcommand\DTLgidxDictHead{%
    \chapter{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
  }%
}%
{%
  \newcommand\DTLgidxDictHead{%
    \section{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
  }%
}
\newcommand*{\DTLgidxCategoryNameFont}[1]{#1}
\newcommand*{\DTLgidxCategorySep}{\space}
\newcommand*{\DTLgidxSubCategorySep}{\space}
\newcommand*{\datagidxdictindent}{1em}
\newcommand{\DTLgidxDictPostItem}{\par}
\newcommand*{\datagidx@style@dict}{%
  \renewcommand*{\datagidxstart}%
  {%
    \bgroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \dimen@=\linewidth
    \advance\dimen@ by -\datagidxdictindent\relax
    \dtl@tmplength=\datagidxdictindent\relax
    \xdef\datagidxdictparshape{%
      \noexpand\parshape=2 0pt \the\linewidth\space
       \the\dtl@tmplength\space \the\dimen@\relax
    }%
    \datagidx@level=1\relax
    \raggedright
  }%
  \renewcommand*{\datagidxend}{\egroup}%
  \renewcommand*{\datagidxgroupsep}{}%
  \renewcommand{\datagidxgroupheader}{%
    \ifdatagidxshowgroups
      \datagidxend
      \datagidx@postend
      \DTLgidxDictHead
      \datagidx@prestart
      \datagidxstart
```

```
  \fi
}%
\renewcommand*{\datagidxitem}{%
  \ifdefempty\datagidxprevgroup
  {%
    \datagidxgroupheader
  }%
  {%
    \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
    {%
    }%
    {%
      \datagidxgroupsep
      \datagidxgroupheader
    }%
  }%
  \datagidxdictparshape
  \datagidxtarget{\Label}%
  {%
    \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
  }%
  \DTLgidxPostName
  \let\datagidx@catsep\@empty
  \let\datagidx@subcatsep\@empty
  \DTLgidxSymbolDescription
  \DTLgidxChildrenSeeAlso
  \DTLgidxDictPostItem
}%
\renewcommand*{\datagidxchildstart}%
{%
  \bgroup
}%
\renewcommand*{\datagidxchildend}{\egroup}%
\renewcommand*{\datagidxchilditem}{%
  \ifnum\datagidx@level=2\relax
    \datagidx@catsep
    \let\datagidx@catsep\DTLgidxCategorySep
    \let\datagidx@subcapsep\@empty
    \datagidxtarget{\Label}%
    {%
      \DTLgidxChildStyle
      {%
        \DTLgidxCategoryNameFont{\DTLgidxNameCase{\Name}}%
        \DTLgidxPostChildName
      }%
    }%
    \setcounter{DTLgidxChildCount}{0}%
  \else
    \datagidx@subcatsep
    \let\datagidx@subcatsep\DTLgidxSubCategorySep
```

```
      \refstepcounter{DTLgidxChildCount}%
      \DTLgidxChildCountLabel
      \DTLgidxPostChildName
    \fi
    \DTLgidxSymbolDescription
    \DTLgidxDoSeeOrLocation
    \DTLgidxChildrenSeeAlso
  }%
  \renewcommand*{\datagidxseealsostart}%
  {%
    \bgroup
      \setlength{\parindent}{0pt}%
      \setlength{\parskip}{0pt plus 0.3pt}%
      \setlength{\dimen@}{\datagidxindent}%
      \advance\datagidx@level by 1\relax
      \multiply\dimen@ by \datagidx@level\relax
      \@idxitem\hspace*{\dimen@}%
  }%
  \renewcommand{\datagidxseealsoend}{\egroup}%
}
\newcommand*{\dtldofirstlocation}{%
  \@for\dtl@thisloc:=\Location\do{%
    \ifdefempty\dtl@thisloc
    {}%
    {%
       \expandafter\datagidx@getlocation\dtl@thisloc
       \datagidxlink{\datagidx@current@target}%
       {%
         \datagidx@formatlocation
            \datagidx@current@format\datagidx@current@locationstring
       }%
      \@endfortrue
    }%
  }%
}
\newcommand*{\datagidx@formatlocation}[2]{%
  \ifdefempty{#1}%
  {#2}%
  {%
    \ifcsdef{#1}%
    {%
      \csuse{#1}{#2}%
    }%
    {%
     \PackageWarning{datagidx}{Unknown format `#1'}%
      #2%
    }%
  }%
}
\newcommand*{\dtldolocationlist}{%
```

```
 \DTLifnull{\Location}%
 {}%
 {%
    \def\datagidx@prev@location{-1}%
    \def\datagidx@prev@locationstring{}%
    \def\datagidx@prev@format{}%
    \def\datagidx@prev@locationformat{}%
    \def\datagidx@prev@prefix{}%
    \def\datagidx@prev@target{}%
    \def\datagidx@location@sep{}%
    \def\datagidx@location@start{-1}%
    \expandafter\forcsvlist\expandafter\datagidx@parse@location
      \expandafter{\Location}%
    \do@prevlocation % tidy up loose ends
  }%
}
\newif\if@dtl@sequential
\newcommand*\datagidx@getlocdo[1]{%
  \ifdefempty\datagidx@current@location
  {}%
  {%
    \eappto\datagidx@current@prefix{%
      \datagidx@current@location\datagidx@compositor
    }%
  }%
  \def\datagidx@current@location{#1}%
}
\def\datagidx@getlocation[#1]#2#3{%
  \def\datagidx@current@locationstring{#2}%
  \bgroup
    \datagidx@escapelocationformat
    \xdef\datagidx@current@locationformat{#2}%
    \datagidx@clearlocationformat
    \xdef\datagidx@current@location{#2}%
  \egroup
  \let\datagidx@list\datagidx@current@location
  \def\datagidx@current@prefix{}%
  \def\datagidx@current@location{}%
  \let\do\datagidx@getlocdo
  \expandafter\datagidx@docomplist
   \expandafter{\datagidx@list}%
  \def\datagidx@current@format{#1}%
  \def\datagidx@current@target{#3}%
}
\newcommand*{\datagidx@parse@location}[1]{%
  \datagidx@getlocation#1\relax
  \ifdefequal{\datagidx@prev@locationstring}{\datagidx@current@locationstring}%
  {%
    \ifdefequal{\datagidx@prev@format}{\datagidx@current@format}%
    {%
```

```
    }%
    {%
      \ifdefempty{\datagidx@current@format}%
      {%
      }%
      {%
        \ifdefempty{\datagidx@prev@format}%
        {%
           \let\datagidx@prev@format\datagidx@current@format
        }%
        {%
           \PackageWarning{datagidx}%
           {%
              Conflicting location formats `\datagidx@prev@format' and
              `\datagidx@current@format' for location `\datagidx@current@location'%
           }%
        }%
      }%
    }%
  }%
  {%
    \@datagidx@parse@location
  }%
}
\newcommand*{\@datagidx@parse@location}{%
  \@dtl@sequentialtrue
  \ifdefequal{\datagidx@prev@format}{\datagidx@current@format}%
  {%
    \ifdefequal{\datagidx@prev@locationformat}{\datagidx@current@locationformat}%
    {%
      \ifdefequal{\datagidx@prev@prefix}{\datagidx@current@prefix}%
      {%
      }%
      {%
        \@dtl@sequentialfalse
      }%
    }%
    {%
      \@dtl@sequentialfalse
    }%
  }%
  {%
    \@dtl@sequentialfalse
  }%
  \if@dtl@sequential
    \ifnumequal{\datagidx@prev@location+1}{\datagidx@current@location}%
    {%
       \ifdefequal
         \datagidx@current@locationformat
         \datagidx@prev@locationformat
```

```
      {%
        \@dtl@sequentialtrue
      }%
      {%
             \@dtl@sequentialfalse
      }%
    }%
    {%
       \@dtl@sequentialfalse
    }%
  \fi
  \if@dtl@sequential
     \ifnumequal{\datagidx@location@start}{-1}%
     {%
       \let\datagidx@location@start\datagidx@prev@location
       \let\datagidx@location@startval\datagidx@prev@locationstring
       \let\datagidx@location@format\datagidx@prev@format
       \let\datagidx@location@target\datagidx@prev@target
     }%
     {%
     }%
  \else
    \do@prevlocation
  \fi
  \let\datagidx@prev@location\datagidx@current@location
  \let\datagidx@prev@format\datagidx@current@format
  \let\datagidx@prev@prefix\datagidx@current@prefix
  \let\datagidx@prev@locationformat\datagidx@current@locationformat
  \let\datagidx@prev@locationstring\datagidx@current@locationstring
  \let\datagidx@prev@target\datagidx@current@target
}
\newcommand*{\DTLgidxLocationSep}{, }
\newcommand*{\DTLgidxLocationF}[2]{%
  #1\DTLgidxLocationSep#2%
}
\newcommand*{\DTLgidxLocationFF}[2]{%
  #1--#2%
}
\newcommand*{\do@prevlocation}{%
  \ifnumequal{\datagidx@location@start}{-1}%
  {%
     \ifdefempty{\datagidx@prev@locationstring}%
     {}%
     {%
       \datagidx@location@sep
       \datagidxlink{\datagidx@prev@target}%
       {%
         \datagidx@formatlocation
           \datagidx@prev@format\datagidx@prev@locationstring
       }%
```

```
      \def\datagidx@location@sep{\DTLgidxLocationSep}%
    }%
  }%
  {%
    \datagidx@location@sep
    \do@locrange
    \def\datagidx@location@sep{\DTLgidxLocationSep}%
    \def\datagidx@location@start{-1}%
  }%
}
\newcommand*{\do@locrange}{%
  \ifnumgreater{\datagidx@prev@location}{\datagidx@location@start+1}%
  {%
    \DTLgidxLocationFF
      {%
        \datagidxlink{\datagidx@location@target}%
        {%
          \datagidx@formatlocation
            \datagidx@location@format\datagidx@location@startval
        }%
      }%
      {%
        \datagidxlink{\datagidx@prev@target}%
        {%
          \datagidx@formatlocation
            \datagidx@prev@format\datagidx@prev@locationstring
        }%
      }%
  }%
  {%
    \DTLgidxLocationF
      {%
        \datagidxlink{\datagidx@location@target}%
        {%
          \datagidx@formatlocation
            \datagidx@location@format\datagidx@location@startval
        }%
      }%
      {%
        \datagidxlink{\datagidx@prev@target}%
        {%
          \datagidx@formatlocation
            \datagidx@prev@format\datagidx@prev@locationstring
        }%
      }%
  }%
}
\newcommand*{\datagidx@defaultdatabase}{}
\newcommand*{\DTLgidxSetDefaultDB}[1]{%
  \renewcommand*{\datagidx@defaultdatabase}{#1}%
```

```
}
\define@key{newgloss}{heading}{\renewcommand*{\datagidx@heading}{#1}}
\define@key{newgloss}{postheading}{%
  \renewcommand*{\datagidx@postheading}{#1}%
}
\newif\ifdatagidxbalance
\datagidxbalancetrue
\define@choicekey{newgloss}{balance}[\val\nr]{true,false}[true]{%
 \ifcase\nr\relax
   \renewcommand*{\datagidx@multicols}{multicols}%
   \datagidxbalancetrue
 \or
   \renewcommand*{\datagidx@multicols}{multicols*}%
   \datagidxbalancefalse
 \fi
}
\define@key{newgloss}{sort}{\renewcommand*{\datagidx@sort}{#1}}
\newcommand*{\datagidx@style}{index}
\define@key{newgloss}{style}{\renewcommand*{\datagidx@style}{#1}}
\newif\ifdatagidxshowgroups
\newcommand*{\datagidx@showgroups}{false}
\define@choicekey{newgloss}{showgroups}{true,false}[true]%
{%
  \renewcommand{\datagidx@showgroups}{#1}%
}%
\ifundef\newgidx
{%
  \newcommand*{\newgidx}{\datagidx@newgidx}
}%
{}
\@onlypreamble\newgidx
\newcommand*{\datagidx@highopt@newgidx}[3][]{%
  \edef\datagidx@indexfilename{\datagidxhighoptfilename{#2}}%
  \IfFileExists{\datagidx@indexfilename}%
  {%
    \input{\datagidx@indexfilename}%
    \bgroup
      \setkeys{newgloss}{#1}%
      \datagidx@newgidx@update{#2}{#3}%
    \egroup
  }%
  {%
    \datagidx@newgidx[#1]{#2}{#3}%
  }%
}
\newcommand*{\loadgidx}[3][]{%
  \input{#2}%
  \bgroup
    \setkeys{newgloss}{sort={},#1}%
    \expandafter\datagidx@newgidx@update\expandafter
```

```
      {\dtllastloadeddb}{#3}%
  \egroup
  \edef\datagidx@defaultdatabase{\dtllastloadeddb}%
  \dtlforcolumn{\Label}{\dtllastloadeddb}{Label}%
  {%
     \csxdef{datagidxentry@\Label}{\dtllastloadeddb}%
  }%
}
\@onlypreamble\loadgidx
\newcommand*{\datagidx@newgidx}[3][]{%
\bgroup
  \setkeys{newgloss}{#1}%
  \ifdefempty{\datagidx@defaultdatabase}%
  {\xdef\datagidx@defaultdatabase{#2}}%
  {}%
  \DTLgnewdb{#2}%
  \DTLaddcolumn{#2}{Label}%
  \DTLaddcolumn{#2}{Location}%
  \DTLaddcolumn{#2}{CurrentLocation}%
  \DTLaddcolumn{#2}{FirstId}%
  \DTLaddcolumn{#2}{Name}%
  \DTLaddcolumn{#2}{Text}%
  \DTLaddcolumn{#2}{Parent}%
  \DTLaddcolumn{#2}{Child}%
  \DTLaddcolumn{#2}{Description}%
  \DTLaddcolumn{#2}{Used}%
  \DTLaddcolumn{#2}{Symbol}%
  \DTLaddcolumn{#2}{Long}%
  \DTLaddcolumn{#2}{Short}%
  \DTLaddcolumn{#2}{See}%
  \DTLaddcolumn{#2}{SeeAlso}%
  \datagidx@newgidx@update{#2}{#3}%
\egroup
}
\newcommand*{\datagidx@newgidx@update}[2]{%
  \DTLnewrow{datagidx}%
  \DTLnewdbentry{datagidx}{Glossary}{#1}%
  \DTLnewdbentry{datagidx}{Title}{#2}%
  {%
    \dtlexpandnewvalue
    \DTLnewdbentry{datagidx}{Heading}{\expandonce\datagidx@heading}%
    \DTLnewdbentry{datagidx}{PostHeading}{\expandonce\datagidx@postheading}%
    \DTLnewdbentry{datagidx}{MultiCols}{\expandonce\datagidx@multicols}%
    \DTLnewdbentry{datagidx}{Sort}{\expandonce\datagidx@sort}%
    \DTLnewdbentry{datagidx}{Style}{\expandonce\datagidx@style}%
    \DTLnewdbentry{datagidx}{ShowGroups}{\expandonce\datagidx@showgroups}%
  }%
}
\newcommand*{\newterm@label}{}
\define@key{newterm}{label}{\renewcommand*{\newterm@label}{#1}}
```

```
\newcommand*{\newterm@parent}{}
\define@key{newterm}{parent}{\renewcommand*{\newterm@parent}{#1}}
\newcommand*{\newterm@text}{}
\define@key{newterm}{text}{\renewcommand*{\newterm@text}{#1}}
\newcommand*{\newterm@description}{}
\define@key{newterm}{description}{%
  \renewcommand*{\newterm@description}{#1}%
}
\define@key{newterm}{plural}{\def\newterm@plural{#1}}
\newcommand*{\newterm@sort}{}
\define@key{newterm}{sort}{\renewcommand*{\newterm@sort}{#1}}
\newcommand*{\newterm@symbol}{}
\define@key{newterm}{symbol}{\renewcommand*{\newterm@symbol}{#1}}
\newcommand*{\newterm@database}{}
\define@key{newterm}{database}{\renewcommand*{\newterm@database}{#1}}
\newcommand*{\newterm@long}{}
\define@key{newterm}{long}{%
  \renewcommand*{\newterm@long}{#1}%
  \def\newterm@longplural{#1s}%
}
\newcommand*{\newterm@short}{}
\define@key{newterm}{short}{%
  \renewcommand*{\newterm@short}{#1}%
  \def\newterm@shortplural{#1s}%
}
\define@key{newterm}{longplural}{%
  \def\newterm@longplural{#1}%
}
\define@key{newterm}{shortplural}{%
  \def\newterm@shortplural{#1}%
}
\newcommand*{\newterm@see}{}
\define@key{newterm}{see}{%
  \renewcommand*{\newterm@see}{#1}%
}
\newcommand*{\newterm@seealso}{}
\define@key{newterm}{seealso}{%
  \renewcommand*{\newterm@seealso}{#1}%
}
\newcommand*{\newterm@defaultshook}{}
\newcommand*{\newterm@extrafields}{}
\newcommand*{\DTLgidxAssignList}{%
  \Name=Name,\Description=Description,\Used=Used,\Symbol=Symbol,%
  \Long=Long,\Short=Short,\LongPlural=LongPlural,\ShortPlural=ShortPlural,%
  \Location=Location,\See=See,\SeeAlso=SeeAlso,%
  \Text=Text,\Plural=Plural,\CurrentLocation=CurrentLocation,%
  \Label=Label,\Parent=Parent,\Children=Child,\FirstId=FirstId,\Sort=Sort%
}
\newcommand*{\datagidxtermkeys}{%
  name,description,symbol,long,short,see,seealso,text,plural,%
```

```
  label,parent,sort%
}
\newcommand*\@datagidx@fieldkey@Name{name}%
\newcommand*\@datagidx@fieldkey@Description{description}%
\newcommand*\@datagidx@fieldkey@Symbol{symbol}%
\newcommand*\@datagidx@fieldkey@Long{long}%
\newcommand*\@datagidx@fieldkey@Short{short}%
\newcommand*\@datagidx@fieldkey@See{see}%
\newcommand*\@datagidx@fieldkey@SeeAlso{seealso}%
\newcommand*\@datagidx@fieldkey@Text{text}%
\newcommand*\@datagidx@fieldkey@Plural{plural}%
\newcommand*\@datagidx@fieldkey@Label{label}%
\newcommand*\@datagidx@fieldkey@Parent{parent}%
\newcommand*\@datagidx@fieldkey@Sort{sort}%
\newcommand*{\newtermaddfield}[4][]{%
  \ifstrempty{#1}%
  {%
    \dtlforcolumn{\datagidx@thisidx}{datagidx}{Glossary}%
    {%
      \DTLaddcolumn{\datagidx@thisidx}{#2}%
    }%
  }%
  {%
    \@for\datagidx@thisidx:=#1\do
    {%
      \DTLaddcolumn{\datagidx@thisidx}{#2}%
    }%
  }%
  \expandafter\gdef\csname newterm@#3\endcsname{}%
  \define@key{newterm}{#3}%
  {%
    \expandafter\def\csname newterm@#3\endcsname{##1}%
  }%
  \gappto\newterm@defaultshook
  {%
    \expandafter\protected@edef\csname newterm@#3\endcsname{#4}%
  }%
  \gappto\newterm@extrafields
  {%
     \protected@edef\datagidx@value{\csname newterm@#3\endcsname}%
     \DTLnewdbentry{\newterm@database}{#2}{\expandonce\datagidx@value}%
  }%
  \xappto\DTLgidxAssignList
  {%
    ,\expandafter\noexpand\csname#2\endcsname=#2%
  }%
  \xappto\datagidxtermkeys{,#3}%
  \expandafter\xdef\csname @datagidx@fieldkey@#2\endcsname{#3}%
  \xappto\datagidxgetchildfields
  {%
```

```
    \noexpand\dtlgetentryfromcurrentrow
     {\expandafter\noexpand\csname#2\endcsname}%
     {\noexpand\dtlcolumnindex{\noexpand\DTLgidxCurrentdb}{#2}}%
  }%
}
\newcommand*{\newtermlabelhook}{}
\newcommand*{\DTLgidxNoFormat}[1]{#1}
\newcommand*{\DTLgidxGobble}[1]{}
\newcommand*{\DTLgidxStripBackslash}[1]{%
  \expandafter\@gobble\string#1%
}
\newcommand*{\DTLgidxName}[2]{%
  #1\space #2%
}
\newcommand*{\DTLgidxNameNum}[1]{\@Roman{#1}}
\newcommand*{\datagidx@namenum}[1]{\two@digits{#1}}
\newcommand*{\DTLgidxPlace}[2]{%
  #2%
}
\newcommand*{\DTLgidxSubject}[2]{%
  #2%
}
\newcommand*{\DTLgidxOffice}[2]{%
  #2 (#1)%
}
\newcommand*{\DTLgidxIgnore}[1]{#1}
\newcommand*{\DTLgidxMac}[1]{#1}
\newcommand*{\datagidx@mac}[1]{Mac}
\newcommand*{\DTLgidxSaint}[1]{#1}
\newcommand*{\datagidx@saint}[1]{Saint}
\newcommand*{\DTLgidxRank}[2]{#1~#2}
\newcommand*{\datagidx@rank}[2]{#2.}
\newcommand*{\DTLgidxParticle}[2]{#1~#2}
\newcommand*{\datagidx@particle}[2]{#2.}
\newcommand*{\datagidx@bothoftwo}[2]{#1#2}
\newcommand*{\datagidx@person}[2]{#2\noexpand\datatoolpersoncomma #1}
\newcommand*{\datagidx@place}[2]{#2\noexpand\datatoolplacecomma #1}
\newcommand*{\datagidx@subject}[2]{#2\noexpand\datatoolsubjectcomma #1}
\newcommand*{\datagidx@paren}[1]{\noexpand\datatoolparenstart #1}
\newcommand*{\datagidx@invert}[2]{#2, #1}
\newcommand*{\DTLgidxParen}[1]{\space(#1)}
\newcommand*{\datagidxwordifygreek}{%
  \def\alpha{alpha}%
  \def\beta{beta}%
  \def\gamma{gamma}%
  \def\delta{delta}%
  \def\epsilon{epsilon}%
  \def\varepsilon{epsilon}%
  \def\zeta{zeta}%
  \def\eta{eta}%
```

```
  \def\theta{theta}%
  \def\vartheta{theta}%
  \def\iota{iota}%
  \def\kappa{kappa}%
  \def\lambda{lambda}%
  \def\mu{mu}%
  \def\nu{nu}%
  \def\xi{xi}%
  \def\pi{pi}%
  \def\varpi{pi}%
  \def\rho{rho}%
  \def\varrho{rho}%
  \def\sigma{sigma}%
  \def\varsigma{sigma}%
  \def\tau{tau}%
  \def\upsilon{upsilon}%
  \def\phi{phi}%
  \def\varphi{phi}%
  \def\chi{chi}%
  \def\psi{psi}%
  \def\omega{omega}%
  \def\Gamma{Gamma}%
  \def\Delta{Delta}%
  \def\Theta{Theta}%
  \def\Lambda{Lambda}%
  \def\Xi{Xi}%
  \def\Pi{Pi}%
  \def\Sigma{Sigma}%
  \def\Upsilon{Upsilon}%
  \def\Phi{Phi}%
  \def\Psi{Psi}%
  \def\Omega{Omega}%
}
\newcommand{\datagidxextendedtoascii}{%
  \def\AE{AE}%
  \def\ae{ae}%
  \def\OE{OE}%
  \def\oe{oe}%
  \def\AA{AA}%
  \def\aa{aa}%
  \def\L{L}%
  \def\l{l}%
  \def\O{O}%
  \def\o{o}%
  \def\SS{SS}%
  \def\ss{ss}%
  \def\th{th}%
  \def\TH{TH}%
  \def\dh{dh}%
  \def\DH{DH}%
```

```
}
\newcommand*{\datagidxconvertchars}{%
  \let~\space
  \ifdef\andname
  {%
    \let\&\andname
  }%
  {%
    \def\&{\expandafter\@gobble\string\&}%
  }%
  \def\_{\string_}%
  \def\${\string$}%
  \def\#{\expandafter\@gobble\string\#}%
  \def\%{\expandafter\@gobble\string\%}%
  \def\{{\expandafter\@gobble\string\{}%
  \def\}{\expandafter\@gobble\string\}}%
}
\@ifl@t@r\fmtversion{2019/10/01}
{%
  \newcommand*{\datagidxstripaccents}{%
   \let\add@accent@\@secondoftwo
   \let\@text@composite@x\@secondoftwo
   \let\@tabacckludge\@secondoftwo
   \expandafter\def\csname \encodingdefault-cmd\endcsname##1##2##3{##3}%
   \expandafter\def\csname OT1-cmd\endcsname##1##2##3{##3}%
   \expandafter\def\csname T1-cmd\endcsname##1##2##3{##3}%
   \expandafter\def\csname PD1-cmd\endcsname##1##2##3{##3}%
   \def\IeC##1{\@gobbletwo##1}%
   \let\UTFviii@two@octets\UTFviii@two@octets@combine
  }%
}
{%
  \newcommand*{\datagidxstripaccents}{%
   \let\add@accent@\@secondoftwo
   \let\@text@composite@x\@secondoftwo
   \let\@tabacckludge\@secondoftwo
   \expandafter\def\csname \encodingdefault-cmd\endcsname##1##2##3{##3}%
   \expandafter\def\csname OT1-cmd\endcsname##1##2##3{##3}%
   \expandafter\def\csname T1-cmd\endcsname##1##2##3{##3}%
   \expandafter\def\csname PD1-cmd\endcsname##1##2##3{##3}%
   \def\IeC##1{\@gobbletwo##1}%
  }%
}
\providecommand{\newterm}{\datagidx@newterm}
\@onlypreamble\newterm
\newcommand{\datagidx@setfieldvalues}[2]{%
 \def\newterm@name{#2}%
 \renewcommand*\newterm@label{#2}%
 \renewcommand*\newterm@text{#2}%
 \undef\newterm@plural
```

```
\renewcommand*{\newterm@description}{}%
\renewcommand*{\newterm@sort}{#2}%
\renewcommand*{\newterm@symbol}{}%
\let\newterm@database\datagidx@defaultdatabase
\renewcommand*{\newterm@short}{#2}%
\undef\newterm@shortplural
\renewcommand*{\newterm@long}{#2}%
\undef\newterm@longplural
\renewcommand*{\newterm@see}{}%
\renewcommand*{\newterm@seealso}{}%
\renewcommand*{\newterm@parent}{}%
\let\datagidx@orgfield\field
\def\field##1{\expandafter\noexpand\csname newterm@##1\endcsname}%
\newterm@defaultshook
\let\field\datagidx@orgfield
\setkeys{newterm}{#1}%
\bgroup
  \let\glsadd\@gobble
  \let\MakeUppercase\DTLgidxNoFormat
  \let\MakeTextUppercase\DTLgidxNoFormat
  \let\MakeLowercase\DTLgidxNoFormat
  \let\MakeTextLowercase\DTLgidxNoFormat
  \let\acronymfont\DTLgidxNoFormat
  \let\textrm\DTLgidxNoFormat
  \let\texttt\DTLgidxNoFormat
  \let\textsf\DTLgidxNoFormat
  \let\textsc\DTLgidxNoFormat
  \let\textbf\DTLgidxNoFormat
  \let\textmd\DTLgidxNoFormat
  \let\textit\DTLgidxNoFormat
  \let\textsl\DTLgidxNoFormat
  \let\emph\DTLgidxNoFormat
  \let\textsuperscript\DTLgidxNoFormat
 \datagidxconvertchars
  \let\ensuremath\DTLgidxNoFormat
  \let\DTLgidxParen\@gobble
  \let\DTLgidxName\@secondoftwo
  \let\DTLgidxPlace\datagidx@invert
  \let\DTLgidxSubject\datagidx@invert
  \let\DTLgidxOffice\@secondoftwo
  \let\DTLgidxParticle\datagidx@bothoftwo
  \datagidxwordifygreek
  \datagidxstripaccents
  \datagidxextendedtoascii
  \newtermlabelhook
  \protected@xdef\newterm@label{\newterm@label}%
  \let\DTLgidxName\datagidx@person
  \let\DTLgidxPlace\datagidx@place
  \let\DTLgidxSubject\datagidx@subject
  \let\DTLgidxOffice\datagidx@person
```

```
   \let\DTLgidxParen\datagidx@paren
   \let\DTLgidxMac\datagidx@mac
   \let\DTLgidxSaint\datagidx@saint
   \let\DTLgidxIgnore\@gobble
   \let\DTLgidxRank\datagidx@rank
   \let\DTLgidxParticle\datagidx@particle
   \let\DTLgidxNameNum\datagidx@namenum
   \protected@xdef\newterm@sort{\newterm@sort}%
 \egroup
}
\newcommand*{\datagidx@add@term}[1]{%
  \global\cslet{datagidxentry@\newterm@label}{\newterm@database}%
  \DTLnewrow{\newterm@database}%
  \DTLnewdbentry{\newterm@database}{Name}{#1}%
  \DTLnewdbentry{\newterm@database}{Used}{0}%
  {%
    \dtlexpandnewvalue
    \DTLnewdbentry{\newterm@database}{Text}{\expandonce\newterm@text}%
    \DTLnewdbentry{\newterm@database}{Description}{\expandonce\newterm@description}%
    \DTLnewdbentry{\newterm@database}{Label}{\expandonce\newterm@label}%
    \DTLnewdbentry{\newterm@database}{Sort}{\expandonce\newterm@sort}%
    \DTLnewdbentry{\newterm@database}{Symbol}{\expandonce\newterm@symbol}%
    \DTLnewdbentry{\newterm@database}{Short}{\expandonce\newterm@short}%
    \DTLnewdbentry{\newterm@database}{Long}{\expandonce\newterm@long}%
    \ifundef\newterm@plural
    {%
      \DTLnewdbentry{\newterm@database}{Plural}{\expandonce\newterm@text s}%
    }%
    {%
      \DTLnewdbentry{\newterm@database}{Plural}{\expandonce\newterm@plural}%
    }%
    \ifundef\newterm@shortplural
    {%
      \DTLnewdbentry{\newterm@database}{ShortPlural}{\expandonce\newterm@short s}%
    }%
    {%
      \DTLnewdbentry{\newterm@database}{ShortPlural}{\expandonce\newterm@shortplural}%
    }%
    \ifundef\newterm@longplural
    {%
      \DTLnewdbentry{\newterm@database}{LongPlural}{\expandonce\newterm@long s}%
    }%
    {%
      \DTLnewdbentry{\newterm@database}{LongPlural}{\expandonce\newterm@longplural}%
    }%
    \ifdefempty{\newterm@see}%
     {}%
     {\DTLnewdbentry{\newterm@database}{See}{\newterm@see}}%
    \ifdefempty{\newterm@seealso}%
     {}%
```

```
     {\DTLnewdbentry{\newterm@database}{SeeAlso}{\newterm@seealso}}%
    \newterm@extrafields
    \ifdefempty{\newterm@parent}%
     {}%
     {%
       \iftermexists{\newterm@parent}%
       {%
         \edef\newterm@parentdatabase{\csuse{datagidxentry@\newterm@parent}}%
         \ifthenelse{\equal{\newterm@parentdatabase}{\newterm@database}}
         {%
           \DTLnewdbentry{\newterm@database}{Parent}{\newterm@parent}%
           \datagidx@addchild{\newterm@database}{\newterm@parent}{\newterm@label}%
         }%
         {%
           \PackageError{datagidx}%
           {%
             Parent entry `\newterm@parent' must belong to the
             same database as child entry `\newterm@label'%
           }%
           {%
             Parent entry is in database
             `\newterm@parentdatabase' and child entry is in
             database `\newterm@database'%
           }%
         }%
       }%
       {%
         \PackageError{datagidx}%
         {%
            Can't assign parent to `\newterm@label':
            `\newterm@parent' doesn't exist%
         }%
         {}%
       }%
     }%
  }%
  \global\let\datagidxlastlabel\newterm@label
  \postnewtermhook
}%
\newcommand*{\postnewtermhook}{}
\newcommand*{\newtermfield}[1]{\csuse{newterm@#1}}
\newcommand{\ifnewtermfield}[3]{%
  \ifcsdef{newterm@#1}
  {%
    \ifcsempty{newterm@#1}{#3}{#2}%
  }%
  {%
    #3%
  }%
}
```

```
\newcommand{\datagidx@newterm}[2][]{%
  \datagidx@setfieldvalues{#1}{#2}%
  \DTLifdbexists{\newterm@database}%
  {%
    \iftermexists{\newterm@label}%
    {%
      \PackageError{datagidx}{Term `\newterm@label' already
        exists in database `\newterm@database'}{}%
    }%
    {%
       \datagidx@add@term{#2}%
    }%
  }%
  {%
    \PackageError{datagidx}%
    {Glossary/index data base `\newterm@database' doesn't exist}%
    {%
      You must define the glossary/index data base before you can
      add any terms to it.%
    }%
  }%
}
\newcommand{\datagidx@highopt@newterm}[2][]{%
  \datagidx@setfieldvalues{#1}{#2}%
  \DTLifdbexists{\newterm@database}%
  {%
    \edef\dtl@dogetrow{%
      \noexpand\dtlgetrowindex
      {\noexpand\dtl@rowidx}%
      {\newterm@database}%
      {%
        \dtlcolumnindex{\newterm@database}{Label}%
      }%
      {\newterm@label}}%
    \dtl@dogetrow
    \ifx\dtl@rowidx\dtlnovalue
       \datagidx@add@term{#2}%
       \csdef{datagidx@do@highopt@sort@\newterm@database}{\datagidx@sort}%
    \else
      \global\cslet{datagidxentry@\newterm@label}{\newterm@database}%
      \global\let\datagidxlastlabel\newterm@label
    \fi
  }%
  {%
    \PackageError{datagidx}%
    {Glossary/index data base `\newterm@database' doesn't exist}%
    {%
      You must define the glossary/index data base before you can
      add any terms to it.%
    }%
```

```
  }%
}
\newcommand*{\datagidx@addchild}[3]{%
  \edef\dtl@dogetrow{%
    \noexpand\dtlgetrowforvalue
    {#1}%
    {%
      \dtlcolumnindex{\newterm@database}{Label}%
    }%
    {#2}}%
  \dtl@dogetrow
  \dtlgetentryfromcurrentrow
    {\datagidx@child}%
    {\dtlcolumnindex{#1}{Child}}%
  \ifx\datagidx@child\dtlnovalue
    \edef\datagidx@child{#3}%
  \else
    \edef\datagidx@child{\datagidx@child,#3}%
  \fi
  \edef\do@update{\noexpand\dtlupdateentryincurrentrow
    {Child}{\datagidx@child}}%
  \do@update
  \dtlrecombine
}
\newcommand{\newacro}[3][]{%
  \newterm
    [%
      description={\capitalisewords{#3}},%
      short={\acronymfont{#2}},%
      long={#3},%
      text={\DTLgidxAcrStyle{#3}{\acronymfont{#2}}},%
      plural={\DTLgidxAcrStyle{#3s}{\acronymfont{#2s}}},%
      sort={#2},%
      #1%
    ]%
    {\MakeTextUppercase{#2}}%
}
\newcommand*{\acronymfont}[1]{#1}
\newcommand*{\DTLgidxAcrStyle}[2]{#1 (#2)}
\newcommand{\iftermexists}[3]{%
  \ifcsdef{datagidxentry@#1}{#2}{#3}%
}
\newcommand*{\datagidxdb}[1]{%
  \csuse{datagidxentry@#1}%
}
\newcommand*{\ifentryused}[3]{%
  \letcs{\newterm@database}{datagidxentry@#1}%
  \edef\dtl@dogetrow{%
    \noexpand\dtlgetrowforvalue
    {\newterm@database}%
```

```
    {%
      \dtlcolumnindex{\newterm@database}{Label}%
    }%
    {#1}}%
  \dtl@dogetrow
  \dtlgetentryfromcurrentrow
    {\datagidx@value}%
    {\dtlcolumnindex{\newterm@database}{Used}}%
  \ifnum\datagidx@value=1\relax
    #2%
  \else
    #3%
  \fi
}
\newcommand*{\glsreset}[1]{%
  \letcs{\newterm@database}{datagidxentry@#1}%
  \edef\do@getrow{%
    \noexpand\dtlgetrowforvalue
    {\newterm@database}%
    {\dtlcolumnindex{\newterm@database}{Label}}%
    {#1}%
  }%
  \do@getrow
  \dtlreplaceentryincurrentrow
    {0}{\dtlcolumnindex{\newterm@database}{Used}}%
  \dtlrecombine
}
\newcommand*{\glsunset}[1]{%
  \letcs{\newterm@database}{datagidxentry@#1}%
  \edef\do@getrow{%
    \noexpand\dtlgetrowforvalue
    {\newterm@database}%
    {\dtlcolumnindex{\newterm@database}{Label}}%
    {#1}%
  }%
  \do@getrow
  \dtlreplaceentryincurrentrow
    {1}{\dtlcolumnindex{\newterm@database}{Used}}%
  \dtlrecombine
}
\newcommand*{\glsresetall}[1]{%
  \def\datagidx@list{}%
  \dtlforcolumn{\datagidx@label}{#1}{Label}%
  {%
    \ifdefempty\datagidx@list
    {%
      \let\datagidx@list\datagidx@label
    }%
    {%
      \eappto\datagidx@list{,\datagidx@label}%
```

```
    }%
  }%
  \@for\datagidx@thislabel:=\datagidx@list\do
  {%
    \glsreset{\datagidx@thislabel}%
  }%
}
\newcommand*{\glsunsetall}[1]{%
  \def\datagidx@list{}%
  \dtlforcolumn{\datagidx@label}{#1}{Label}%
  {%
    \ifdefempty\datagidx@list
    {%
      \let\datagidx@list\datagidx@label
    }%
    {%
      \eappto\datagidx@list{,\datagidx@label}%
    }%
  }%
  \@for\datagidx@thislabel:=\datagidx@list\do
  {%
    \glsunset{\datagidx@thislabel}%
  }%
}
\newcount\datagidx@anchorcount
\newcommand*{\datagidx@formatanchor}[1]{%
  \ifnum#1<10000
   0%
    \ifnum#1<1000
     0%
      \ifnum#1<100
       0%
        \ifnum#1<10
         0%
        \fi
      \fi
    \fi
  \fi
  \number#1%
}
\newcommand*{\@datagidx@escloc}[2]{%
  \expandafter\string\csname#1\endcsname{\noexpand\number#2}%
}
\newcommand*{\datagidx@escapelocation}{%
  \def\@arabic{\@datagidx@escloc{@arabic}}%
  \def\@roman{\@datagidx@escloc{@roman}}%
  \def\@Roman{\@datagidx@escloc{@Roman}}%
  \def\@alph{\@datagidx@escloc{@alph}}%
  \def\@Alph{\@datagidx@escloc{@Alph}}%
}
```

```
\newcommand*{\datagidx@escapelocationformat}{%
  \def\@arabic##1{arabic}%
  \def\@roman##1{roman}%
  \def\@Roman##1{Roman}%
  \def\@alph##1{alph}%
  \def\@Alph##1{Alph}%
}
\newcommand*{\datagidx@clearlocationformat}{%
  \let\@arabic\@firstofone
  \let\@roman\@firstofone
  \let\@Roman\@firstofone
  \let\@alph\@firstofone
  \let\@Alph\@firstofone
}
\newcommand*{\DTLgidxAddLocationType}[1]{%
  \gappto\datagidx@escapelocation{%
    \expandafter\def\csname#1\endcsname{\@datagidx@escloc{#1}}%
  }%
  \gappto\datagidx@escapelocationformat{%
    \expandafter\def\csname#1\endcsname##1{#1}%
  }%
  \gappto\datagidx@clearlocationformat{%
    \expandafter\let\csname#1\endcsname\@firstofone
  }%
}
\@onlypreamble\DTLgidxAddLocationType
\newcommand*{\datagidx@target}[4]{%
  \global\advance\datagidx@anchorcount by 1\relax
  \edef\@datagidx@target{datagidx.\datagidx@formatanchor\datagidx@anchorcount}%
  \ifstrempty{#3}
  {%
    \datagidx@write@usedentry{#1}{}%
  }%
  {%
    \bgroup
      \datagidx@escapelocation
      \def\@arabic{\noexpand\@arabic}%
      \def\@roman{\noexpand\@roman}%
      \def\@Roman{\noexpand\@Roman}%
      \def\@alph{\noexpand\@alph}%
      \def\@Alph{\noexpand\@Alph}%
      \protected@edef\@datagidx@dowriteaux{%
         \noexpand\datagidx@write@usedentry{#1}%
              {[#2]{#3}{\@datagidx@target}}%
      }%
      \@datagidx@dowriteaux
    \egroup
  }%
  \ifdef\hypertarget
  {%
```

```
    \datagidxshowifdraft
    {%
      [\@datagidx@target]%
      \discretionary{}{}{}%
    }%
    \bgroup
      \let\glsadd\@gobble
      \settoheight\dimen@{#4}%
      \raisebox{\dimen@}%
      {%
        \datagidxtarget{\@datagidx@target}{}%
      }%
    \egroup
  }%
  {%
  }%
  \datagidxshowifdraft{[#1]\discretionary{}{}{}}%
  #4%
}
\DeclareRobustCommand*{\glsdispentry}[2]{%
  \DTLgidxFetchEntry{\datagidx@dispentryval}{#1}{#2}%
  \datagidx@dispentryval
}
\DeclareRobustCommand*{\Glsdispentry}[2]{%
  \DTLgidxFetchEntry{\datagidx@dispentryval}{#1}{#2}%
  \xmakefirstuc\datagidx@dispentryval
}
\newcommand*{\DTLgidxFetchEntry}[3]{%
  \ifcsdef{datagidxentry@#2}%
  {%
  \letcs{\newterm@database}{datagidxentry@#2}%
  \edef\do@getrow{%
    \noexpand\dtlgetrowforvalue
    {\newterm@database}%
    {\dtlcolumnindex{\newterm@database}{Label}}%
    {#2}%
  }%
  \do@getrow
  \dtlgetentryfromcurrentrow
    {#1}%
    {\dtlcolumnindex{\newterm@database}{#3}}%
  }%
  {%
    \PackageError{datagidx}{No term `#2' defined}{}%
  }%
}
\newcommand*{\datagidx@parse@formatlabel}[1]{%
  \datagidx@parse@format@label@#1\@endparse@formatlabel@
}
\newcommand*\datagidx@parse@format@label@{%
```

```
  \@ifnextchar[{\datagidx@parse@formatlabel@}{\datagidx@parse@formatlabel@[]}%
}
\def\datagidx@parse@formatlabel@[#1]#2\@endparse@formatlabel@{%
  \def\datagidx@format{#1}%
  \def\datagidx@label{#2}%
}
\newcommand*{\@datagidx@use@entry}[1]{%
  \ifcsundef{datagidxentry@\datagidx@label}
  {%
    \PackageError{datagidx}{Entry `\datagidx@label' doesn't exist}{}%
  }%
  {%
    \letcs{\newterm@database}{datagidxentry@\datagidx@label}%
    \edef\do@getrow{%
      \noexpand\dtlgetrowforvalue
      {\newterm@database}%
      {\dtlcolumnindex{\newterm@database}{Label}}%
      {\datagidx@label}%
    }%
    \do@getrow
    \dtlgetentryfromcurrentrow
      {\datagidx@id}%
      {\dtlcolumnindex{\newterm@database}{FirstId}}%
    \DTLifnull\datagidx@id
    {%
      \count@=\datagidx@anchorcount\relax
      \advance\count@ by 1\relax
      \dtlappendentrytocurrentrow{FirstId}{\datagidx@formatanchor\count@}%
    }%
    {}%
    \dtlreplaceentryincurrentrow
      {1}{\dtlcolumnindex{\newterm@database}{Used}}%
    \dtlgetentryfromcurrentrow
      {\datagidx@parent}%
      {\dtlcolumnindex{\newterm@database}{Parent}}%
    \dtlrecombine
    \datagidx@markparent{\newterm@database}{\datagidx@parent}%
    \datagidx@target{\datagidx@label}{\datagidx@format}%
      {\csuse{the\DTLgidxCounter}}{#1}%
  }%
}
\newcommand*{\DTLgidxCounter}{page}
\newcommand*{\datagidx@markparent}[2]{%
  \ifx#2\dtlnovalue
  \else
    \datagidx@target{#2}{}{}{}%
    \edef\do@getrow{%
      \noexpand\dtlgetrowforvalue
      {#1}%
      {\dtlcolumnindex{#1}{Label}}%
```

```
      {#2}}%
    \do@getrow
  \dtlgetentryfromcurrentrow
    {\datagidx@id}%
    {\dtlcolumnindex{\newterm@database}{FirstId}}%
  \DTLifnull\datagidx@id
  {%
    \dtlappendentrytocurrentrow{FirstId}{\datagidx@formatanchor\datagidx@anchorcount}%
  }%
  {}%
    \dtlgetentryfromcurrentrow
      {\datagidx@parent}%
      {\dtlcolumnindex{#1}{Parent}}%
  \dtlrecombine
    \datagidx@markparent{#1}{\datagidx@parent}%
  \fi
}
\newcommand*{\datagidx@write@usedentry}[2]{%
  \datagidx@do@highopt@update{#1}%
  \protected@write{\@auxout}{}%
    {%
      \string\datagidx@usedentry{#1}{#2}%
    }%
  \protected@edef\datagidx@do@usedentry{%
    \noexpand\datagidx@xusedentry{CurrentLocation}{#1}{#2}%
  }%
  \expandafter\ifstrequal\expandafter{\DTLgidxCounter}{page}%
  {%
     \expandafter\afterpage\expandafter{\datagidx@do@usedentry}%
  }%
  {
    \datagidx@do@usedentry
  }%
}
\newcommand*{\datagidx@xusedentry}[3]{%
  \protected@edef\@datagidx@do@xusedentry{%
     \noexpand\datagidx@usedentry[#1]{#2}{#3}%
  }%
  \@datagidx@do@xusedentry
}
\newcommand*{\datagidx@usedentry}[3][Location]{%
  \ifcsundef{datagidxentry@#2}%
  {%
    \PackageWarning{datagidx}{No term `#2' defined. Ignoring}%
  }%
  {%
    \letcs{\newterm@database}{datagidxentry@#2}%
    \edef\do@getrow{%
      \noexpand\dtlgetrowforvalue
      {\newterm@database}%
```

```
      {\dtlcolumnindex{\newterm@database}{Label}}%
      {#2}%
    }%
    \do@getrow
    \dtlgetentryfromcurrentrow
      {\datagidx@loc}%
      {\dtlcolumnindex{\newterm@database}{#1}}%
    \ifx\datagidx@loc\dtlnovalue
      \def\datagidx@loc{#3}%
      \dtlappendentrytocurrentrow{#1}{\expandonce\datagidx@loc}%
    \else
      \ifdefempty{\datagidx@loc}%
      {%
        \def\datagidx@loc{#3}%
      }%
      {%
        \ifstrempty{#3}%
        {}%
        {%
          \appto\datagidx@loc{,#3}%
        }%
      }%
      \expandafter\dtlreplaceentryincurrentrow\expandafter
        {\datagidx@loc}%
        {\dtlcolumnindex{\newterm@database}{#1}}%
    \fi
    \dtlrecombine
  }%
}
\newcommand*{\datagidx@save@loc}[2]{%
 \bgroup
  \datagidx@escapelocation
  \xdef\datagidx@tmp{#2}%
 \egroup
 \expandafter\xdef\csname datagidx@prev@loc@#1\endcsname{\datagidx@tmp}%
}
\newcommand*{\glsadd}[1]{%
  \NoCaseChange{\@glsadd{#1}}%
}
\DeclareRobustCommand*{\@glsadd}[1]{%
  \ifcsundef{datagidxentry@\datagidx@label}%
  {%
    \PackageError{datagidx}{Term `\datagidx@label' doesn't exist}{}%
  }%
  {%
    \datagidx@parse@formatlabel{#1}%
    \datagidx@target{\datagidx@label}{\datagidx@format}%
      {\csuse{the\DTLgidxCounter}}{}%
    \letcs{\newterm@database}{datagidxentry@\datagidx@label}%
    \edef\do@getrow{%
```

```
      \noexpand\dtlgetrowforvalue
      {\newterm@database}%
      {\dtlcolumnindex{\newterm@database}{Label}}%
      {\datagidx@label}%
    }%
    \do@getrow
    \dtlreplaceentryincurrentrow
      {1}{\dtlcolumnindex{\newterm@database}{Used}}%
    \dtlgetentryfromcurrentrow
      {\datagidx@id}%
      {\dtlcolumnindex{\newterm@database}{FirstId}}%
    \DTLifnull\datagidx@id
    {%
      \dtlappendentrytocurrentrow{FirstId}{\datagidx@formatanchor\datagidx@anchorcount}%
    }%
    {}%
    \dtlrecombine
  }%
}
\newcount\datagidx@count
\newcommand*{\glsaddall}[1]{%
  \DTLifdbexists{#1}%
  {%
     \edef\datagidx@rowcount{\number\DTLrowcount{#1}}%
     \datagidx@count=0\relax
     \loop
       \advance\datagidx@count by 1\relax
       \dtlgetrow{#1}{\datagidx@count}%
      \dtlgetentryfromcurrentrow
        {\datagidx@label}%
        {\dtlcolumnindex{#1}{Label}}%
    \bgroup
      \undef\hypertarget
      \datagidx@target{\datagidx@label}{}{}{}%
    \egroup
       \dtlreplaceentryincurrentrow
        {1}{\dtlcolumnindex{#1}{Used}}%
      \dtlgetentryfromcurrentrow
        {\datagidx@id}%
        {\dtlcolumnindex{#1}{FirstId}}%
      \DTLifnull\datagidx@id
      {%
        \dtlappendentrytocurrentrow{FirstId}{\datagidx@formatanchor\datagidx@anchorcount
      }%
      {}%
      \dtlrecombine
     \ifnum\datagidx@count<\datagidx@rowcount
     \repeat
  }%
  {%
```

```
    \PackageError{datagidx}{Database `#1' doesn't exist}{}%
  }%
}
\DeclareRobustCommand*{\glslink}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \datagidxlink{\datagidx@label}%
  {%
    \@datagidx@use@entry{#2}%
  }%
}
\DeclareRobustCommand*{\useentry}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
  \datagidxlink{\datagidx@label}%
  {%
    \@datagidx@use@entry{\datagidx@value}%
  }%
}
\DeclareRobustCommand*{\Useentry}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
  \datagidxlink{\datagidx@label}%
  {%
    \@datagidx@use@entry{\xmakefirstuc{\datagidx@value}}%
  }%
}
\DeclareRobustCommand*{\USEentry}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
  \datagidxlink{\datagidx@label}%
  {%
    \@datagidx@use@entry{\MakeTextUppercase{\datagidx@value}}%
  }%
}
\DeclareRobustCommand*{\useentrynl}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
  \@datagidx@use@entry{\datagidx@value}%
}
\DeclareRobustCommand*{\Useentrynl}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
  \@datagidx@use@entry{\xmakefirstuc{\datagidx@value}}%
}
\DeclareRobustCommand*{\USEentrynl}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
  \@datagidx@use@entry{\MakeTextUppercase{\datagidx@value}}%
}
\DeclareRobustCommand*{\gls}[1]{\useentry{#1}{Text}}
```

```
\DeclareRobustCommand*{\glspl}[1]{\useentry{#1}{Plural}}
\DeclareRobustCommand*{\Gls}[1]{\Useentry{#1}{Text}}
\DeclareRobustCommand*{\Glspl}[1]{\Useentry{#1}{Plural}}
\DeclareRobustCommand*{\glsnl}[1]{\useentrynl{#1}{Text}}
\DeclareRobustCommand*{\glsplnl}[1]{\useentrynl{#1}{Plural}}
\DeclareRobustCommand*{\Glsnl}[1]{\Useentrynl{#1}{Text}}
\DeclareRobustCommand*{\Glsplnl}[1]{\Useentrynl{#1}{Plural}}
\DeclareRobustCommand*{\glssym}[1]{\useentry{#1}{Symbol}}
\DeclareRobustCommand*{\Glssym}[1]{\Useentry{#1}{Symbol}}
\newcommand*{\DTLgidxFormatAcr}[3]{%
  \DTLgidxAcrStyle{\glsdispentry{#1}{#2}}{\useentry{#1}{#3}}%
}
\newcommand*{\DTLgidxFormatAcrUC}[3]{%
  \DTLgidxAcrStyle{\Glsdispentry{#1}{#2}}{\useentry{#1}{#3}}%
}
\DeclareRobustCommand*{\acr}[1]{%
  \ifentryused{#1}%
  {\useentry{#1}{Short}}%
  {\DTLgidxFormatAcr{#1}{Long}{Short}}%
}
\DeclareRobustCommand*{\acrpl}[1]{%
  \ifentryused{#1}%
  {\useentry{#1}{ShortPlural}}%
  {\DTLgidxFormatAcr{#1}{LongPlural}{ShortPlural}}%
}
\DeclareRobustCommand*{\Acr}[1]{%
  \ifentryused{#1}%
  {\Useentry{#1}{Short}}%
  {\DTLgidxFormatAcrUC{#1}{Long}{Short}}%
}
\DeclareRobustCommand*{\Acrpl}[1]{%
  \ifentryused{#1}%
  {\Useentry{#1}{ShortPlural}}%
  {\DTLgidxFormatAcrUC{#1}{LongPlural}{ShortPlural}}%
}
\define@key{printterms}{database}{\renewcommand*{\newterm@database}{#1}}
\define@choicekey{printterms}{postdesc}[\val\nr]%
 {none,dot}%
 {%
   \datagidx@setpostdesc\nr
 }
\define@choicekey{printterms}{prelocation}[\val\nr]%
 {none,enspace,space,dotfill,hfill}%
 {%
   \datagidx@setprelocation\nr
 }
\define@choicekey{printterms}{location}[\val\nr]%
 {hide,list,first}%
 {\datagidx@setlocation\nr}
\define@choicekey{printterms}{symboldesc}[\val\nr]%
```

```
 {symbol,desc,(symbol) desc,desc (symbol),symbol desc,desc symbol}%
 {\datagidx@formatsymdesc\nr}
\define@key{printterms}{columns}%
{%
  \DTLgidxSetColumns{#1}%
}
\define@choicekey{printterms}{namecase}[\val\nr]%
 {nochange,uc,lc,firstuc,capitalise}%
 {%
   \datagidx@setnamecase\nr
 }
\define@key{printterms}{namefont}%
{%
  \renewcommand*{\DTLgidxNameFont}[1]{{#1{##1}}}%
}
\define@key{printterms}{postname}
{%
  \renewcommand*{\DTLgidxPostName}{#1}%
}
\define@choicekey{printterms}{see}[\val\nr]%
  {comma,brackets,dot,space,nosep,semicolon,location}%
  {\datagidx@setsee\nr}
\define@choicekey{printterms}{child}[\val\nr]%
 {named,noname}%
 {%
   \datagidx@setchildstyle\nr
 }
\define@key{printterms}{symbolwidth}%
{%
  \setlength{\datagidxsymbolwidth}{#1}%
}
\define@key{printterms}{locationwidth}%
{%
  \setlength{\datagidxlocationwidth}{#1}%
}
\define@choicekey{printterms}{childsort}[\val\nr]%
 {true,false}[true]%
 {%
   \datagidx@setchildsort\nr
 }
\define@choicekey{printterms}{showgroups}{true,false}[true]{%
 \appto\newterm@styles{showgroups={#1},}%
}
\define@key{printterms}{style}{\appto\newterm@styles{style={#1},}}
\define@key{printterms}{heading}{\appto\newterm@styles{heading={#1},}}
\define@key{printterms}{postheading}{%
  \appto\newterm@styles{postheading={#1},}%
}
\define@key{printterms}{sort}{\appto\newterm@styles{sort={#1},}}
\define@choicekey{printterms}{balance}[\val\nr]{true,false}[true]{%
```

```
  \ifcase\nr\relax
    \appto\newterm@styles{balance=true,}%
  \or
    \appto\newterm@styles{balance=false,}%
  \fi
}
\newcommand*{\printterms@condition}{\boolean{true}}
\define@key{printterms}{condition}{\renewcommand*{\printterms@condition}{#1}}
\newcommand{\printtermsstartpar}{\par}
\newcommand*{\printterms@setupmulticol}{%
 \ifdefempty\datagidx@postheading
 {%
   \edef\datagidx@prestart{%
    \noexpand\datagidx@heading{\noexpand\datagidx@title}%
    \noexpand\begin{\datagidx@multicols}{\datagidx@columns}%
   }%
 }%
 {%
   \edef\datagidx@prestart{%
    \noexpand\datagidx@heading{\noexpand\datagidx@title}%
    \noexpand\begin{\datagidx@multicols}{\datagidx@columns}%
    [\noexpand\datagidx@postheading]%
   }%
 }%
 \edef\datagidx@postend{%
  \noexpand\end{\datagidx@multicols}%
 }%
}
\newcommand*{\printterms@setuptwocol}{%
 \def\datagidx@prestart{%
   \twocolumn[\datagidx@heading{\datagidx@title}%
   \datagidx@postheading]}%
 \if@twocolumn
   \def\datagidx@postend{}%
 \else
   \def\datagidx@postend{\printtermsrestoreonecolumn}%
 \fi
}
\newcommand{\printtermsrestoreonecolumn}{\onecolumn}
\newcommand{\printterms}[1][]{%
\bgroup
  \let\newterm@database\datagidx@defaultdatabase
  \let\newterm@styles\@empty
  \setkeys{printterms}{#1}%
  \DTLifdbexists{\newterm@database}%
  {%
    \edef\DTLgidxCurrentdb{\newterm@database}%
    \edef\do@getrow{\noexpand\dtlgetrowforvalue
      {datagidx}%
      {\dtlcolumnindex{datagidx}{Glossary}}%
```

```
  {\newterm@database}%
}%
\do@getrow
\dtlgetentryfromcurrentrow
  {\datagidx@title}%
  {\dtlcolumnindex{datagidx}{Title}}%
\dtlgetentryfromcurrentrow
  {\datagidx@heading}%
  {\dtlcolumnindex{datagidx}{Heading}}%
\dtlgetentryfromcurrentrow
  {\datagidx@postheading}%
  {\dtlcolumnindex{datagidx}{PostHeading}}%
\dtlgetentryfromcurrentrow
  {\datagidx@multicols}%
  {\dtlcolumnindex{datagidx}{MultiCols}}%
\dtlgetentryfromcurrentrow
  {\datagidx@sort}%
  {\dtlcolumnindex{datagidx}{Sort}}%
\dtlgetentryfromcurrentrow
  {\datagidx@style}%
  {\dtlcolumnindex{datagidx}{Style}}%
\dtlgetentryfromcurrentrow
  {\datagidx@showgroups}%
  {\dtlcolumnindex{datagidx}{ShowGroups}}%
\edef\dtl@do@setkeys{\noexpand\setkeys{newgloss}{\expandonce\newterm@styles}}%
\dtl@do@setkeys
\ifnum\datagidx@columns>1\relax
   \ifnum\datagidx@columns=2\relax
    \ifdatagidxbalance
      \printterms@setupmulticol
    \else
      \printterms@setuptwocol
    \fi
   \else
      \printterms@setupmulticol
   \fi
\else
   \def\datagidx@prestart{}%
   \def\datagidx@postend{}%
\fi
\let\@dtl@dbname\DTLgidxCurrentdb
\csuse{datagidxshowgroups\datagidx@showgroups}%
\datagidxsetstyle{\datagidx@style}%
\def\datagidx@labellist{}%
\ifnum\datagidx@columns=1\relax
  \datagidx@heading{\datagidx@title}%
  \datagidx@postheading
\fi
\datagidx@do@sort
\datagidx@prestart
```

```
    \printtermsstartpar
    \datagidxstart
    \let\DTLgidxName\datagidx@invert
    \let\DTLgidxPlace\datagidx@invert
    \let\DTLgidxSubject\datagidx@invert
    \let\DTLgidxOffice\datagidx@invert
      \DTLgidxForeachEntry
      {%
         \datagidxitem
      }%
    \datagidxend
    \datagidx@postend
  }%
  {%
    \PackageError{datagidx}%
    {Glossary/index data base `\newterm@database' doesn't exist}%
    {%
      You must define the glossary/index data base before you can
      use it.%
    }%
  }%
\egroup
}
\def\datagidx@getgroup#1#2\datagidx@endgetgroup{%
  \dtl@setcharcode{#1}{\count@}%
  \dtlifintclosedbetween{\count@}{48}{57}%
  {%
    \gdef\datagidxcurrentgroup{Numbers}%
  }%
  {%
    \dtlifintclosedbetween{\count@}{97}{122}%
    {%
      \advance\count@ by -96\relax
      \xdef\datagidxcurrentgroup{\@Alph\count@}%
    }%
    {%
      \dtlifintclosedbetween{\count@}{65}{90}%
      {%
        \gdef\datagidxcurrentgroup{#1}%
      }%
      {%
        \gdef\datagidxcurrentgroup{Symbols}%
      }%
    }%
  }%
}
\newcommand*{\DTLgidxGroupHeaderTitle}[1]{%
  \ifcsdef{datagidx#1name}
  {%
    \csuse{datagidx#1name}%
```

```
  }%
  {%
    #1%
  }%
}
\newcommand{\DTLgidxForeachEntry}[1]{%
  \def\datagidxprevgroup{}%
  \edef\datagidx@doforeachentry{%
    \noexpand\DTLforeach*[\expandonce\printterms@condition]{\DTLgidxCurrentdb}%
     {\expandonce\DTLgidxAssignList}
  }%
  \datagidx@doforeachentry
  {%
    \DTLifnull{\Parent}%
    {%
      \DTLifnull\Location
      {%
         \DTLifnull\CurrentLocation
         {%
         }%
         {%
           \global\let\@datagidx@dorerun@warn\@data@rerun@warn
         }%
      }%
      {%
        \ifcsdef{datagidx@prev@loc@\Label}%
        {%
          \dtlgidx@checklocationchange
        }%
        {%
           \global\let\@datagidx@dorerun@warn\@data@rerun@warn
        }%
      }%
      \datagidx@doifdisplayed
      {%
        \edef\datagidx@dowrite{%
          \noexpand\protected@write\noexpand\@auxout{}%
          {%
            \string\datagidx@save@loc{\Label}{\CurrentLocation}%
          }%
        }%
        \datagidx@dowrite
        \datagidx@level=1\relax
        \expandafter\datagidx@getgroup\Sort{}\datagidx@endgetgroup
        #1%
        \global\let\datagidxprevgroup\datagidxcurrentgroup
      }%
    }%
    {}%
  }%
```

```
}
\newcommand*{\dtlgidx@checklocationchange}{%
  \protected@edef\@prev@location{%
     \csname datagidx@prev@loc@\Label\endcsname}%
  \@onelevel@sanitize\@prev@location
  \protected@edef\@cur@location{\CurrentLocation}%
  \@onelevel@sanitize\@cur@location
  \ifdefequal{\@prev@location}{\@cur@location}%
  {}%
  {%
    \global\let\@datagidx@dorerun@warn\@data@rerun@warn
  }%
}
\newcommand{\datagidx@doifdisplayed}[1]{%
  \DTLifnull{\Location}%
  {%
     \DTLifnull{\See}
     {%
       \DTLifnull{\SeeAlso}{}%
       {%
         #1%
       }%
     }%
     {%
       \@for\dtl@thislabel:=\See\do
       {%
           \iftermexists{\dtl@thislabel}%
           {%
             \ifentryused{\dtl@thislabel}%
             {%
               #1%
               \@endfortrue
             }%
             {}%
           }%
           {%
           }%
       }%
     }%
  }%
  {%
    #1%
  }%
}%
\newcount\datagidx@level
```

30.13 Rollback v2.32 (datapie-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datapie}[2019/09/27 v2.32 (NLCT)]
```

```
\RequirePackage{xkeyval}
\newif\ifDTLcolorpiechart
\DTLcolorpiecharttrue
\DeclareOption{color}{\DTLcolorpiecharttrue}
\DeclareOption{gray}{\DTLcolorpiechartfalse}
\define@boolkey{datapie}[DTL]{rotateinner}[true]{}
\define@boolkey{datapie}[DTL]{rotateouter}[true]{}
\DTLrotateinnerfalse
\DTLrotateouterfalse
\DeclareOption{rotateinner}{\DTLrotateinnertrue}
\DeclareOption{norotateinner}{\DTLrotateinnerfalse}
\DeclareOption{rotateouter}{\DTLrotateoutertrue}
\DeclareOption{norotateouter}{\DTLrotateouterfalse}
\ProcessOptions
\RequirePackage{datatool}[=v2.32]
\RequirePackage{tikz}
\newlength\DTLradius
\DTLradius=2cm
\newcommand*{\DTLinnerratio}{0.5}
\newcommand*{\DTLouterratio}{1.25}
\newcommand*\DTLcutawayratio{0.2}
\newcommand*{\DTLstartangle}{0}
\newlength\dtl@inneroffset
\dtl@inneroffset=\DTLinnerratio\DTLradius
\newlength\dtl@outeroffset
\dtl@outeroffset=\DTLouterratio\DTLradius
\newlength\dtl@cutawayoffset
\dtl@cutawayoffset=\DTLcutawayratio\DTLradius
\newcommand*{\dtl@piecutaways}{}
\def\dtl@innerlabel{\DTLpievariable}%
\def\dtl@outerlabel{}%
\newcounter{DTLpieroundvar}
\setcounter{DTLpieroundvar}{1}
\newcommand*{\DTLdisplayinnerlabel}[1]{#1}
\newcommand*{\DTLdisplayouterlabel}[1]{#1}
\newcommand*{\DTLpiepercent}{%
\ifnum\dtlforeachlevel=0\relax
  \PackageError{datapie}{Can't use
  \string\DTLpiepercent\space outside
  \string\DTLpiechart}{}%
\else
  \csname dtl@piepercent@\romannumeral\@dtl@seg\endcsname
\fi}
\newcommand*{\DTLpieatbegintikz}{}
\newcommand*{\DTLpieatendtikz}{}
\newcommand*{\DTLsetpiesegmentcolor}[2]{%
\expandafter\def\csname dtlpie@segcol\romannumeral#1\endcsname{#2}%
}
\newcommand*{\DTLgetpiesegmentcolor}[1]{%
\csname dtlpie@segcol\romannumeral#1\endcsname}
```

```
\newcommand*{\DTLdopiesegmentcolor}[1]{%
\expandafter\color\expandafter
{\csname dtlpie@segcol\romannumeral#1\endcsname}}
\newcommand*{\DTLdocurrentpiesegmentcolor}{%
\ifnum\dtlforeachlevel=0\relax
  \PackageError{datapie}{Can't use
  \string\DTLdocurrentpiesegmentcolor\space outside
  \string\DTLpiechart}{}%
\else
  \expandafter\DTLdopiesegmentcolor\expandafter{%
  \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname}%
\fi}
\newcommand*{\DTLpieoutlinecolor}{black}
\newlength\DTLpieoutlinewidth
\DTLpieoutlinewidth=0pt
\ifDTLcolorpiechart
 \DTLsetpiesegmentcolor{1}{red}
 \DTLsetpiesegmentcolor{2}{green}
 \DTLsetpiesegmentcolor{3}{blue}
 \DTLsetpiesegmentcolor{4}{yellow}
 \DTLsetpiesegmentcolor{5}{magenta}
 \DTLsetpiesegmentcolor{6}{cyan}
 \DTLsetpiesegmentcolor{7}{orange}
 \DTLsetpiesegmentcolor{8}{white}
\else
 \DTLsetpiesegmentcolor{1}{black!15}
 \DTLsetpiesegmentcolor{2}{black!25}
 \DTLsetpiesegmentcolor{3}{black!35}
 \DTLsetpiesegmentcolor{4}{black!45}
 \DTLsetpiesegmentcolor{5}{black!55}
 \DTLsetpiesegmentcolor{6}{black!65}
 \DTLsetpiesegmentcolor{7}{black!75}
 \DTLsetpiesegmentcolor{8}{black!85}
\fi
\define@key{datapie}{start}{\def\DTLstartangle{#1}}
\define@key{datapie}{radius}{\DTLradius=#1\relax
\dtl@inneroffset=\DTLinnerratio\DTLradius
\dtl@outeroffset=\DTLouterratio\DTLradius
\dtl@cutawayoffset=\DTLcutawayratio\DTLradius}
\define@key{datapie}{innerratio}{%
\def\DTLinnerratio{#1}%
\dtl@inneroffset=\DTLinnerratio\DTLradius}
\define@key{datapie}{outerratio}{%
\def\DTLouterratio{#1}%
\dtl@outeroffset=\DTLouterratio\DTLradius}
\define@key{datapie}{cutawayratio}{%
\def\DTLcutawayratio{#1}%
\dtl@cutawayoffset=\DTLcutawayratio\DTLradius}
\define@key{datapie}{inneroffset}{%
\dtl@inneroffset=#1}
```

```
\define@key{datapie}{outeroffset}{%
\dtl@outeroffset=#1}
\define@key{datapie}{cutawayoffset}{%
\dtl@cutawayoffset=#1}
\define@key{datapie}{cutaway}{%
\renewcommand*{\dtl@piecutaways}{#1}}
\define@key{datapie}{variable}{%
\def\DTLpievariable{#1}}
\define@key{datapie}{innerlabel}{%
\def\dtl@innerlabel{#1}}
\define@key{datapie}{outerlabel}{%
\def\dtl@outerlabel{#1}}
\newcommand*{\DTLpiechart}[4][\boolean{true}]{%
\bgroup
  \let\DTLpievariable=\relax
  \setkeys{datapie}{#2}%
  \ifx\DTLpievariable\relax
    \PackageError{datapie}%
    {\string\DTLpiechart\space missing variable}{}%
  \else
    \def\dtl@total{0}%
    \@sDTLforeach[#1]{#3}{#4}{%
      \let\dtl@oldtotal=\dtl@total
      \expandafter\DTLconverttodecimal\expandafter
        {\DTLpievariable}{\dtl@variable}%
      \dtladd{\dtl@total}{\dtl@variable}{\dtl@total}%
    }%
    \expandafter\DTLconverttodecimal\expandafter
      {\DTLstartangle}{\@dtl@start}%
    \@sDTLforeach[#1]{#3}{#4}{%
      \expandafter\DTLconverttodecimal\expandafter
        {\DTLpievariable}{\dtl@variable}%
      \dtl@computeangles
        {\csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname}%
        {\dtl@variable}%
      \expandafter\@dtl@seg\expandafter=
        \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname%
      \dtlmul{\dtl@tmp}{\dtl@variable}{100}%
      \let\dtl@old=\dtl@tmp
      \dtldiv{\dtl@tmp}{\dtl@old}{\dtl@total}%
      \expandafter\dtlround
        \csname dtl@piepercent@\romannumeral\@dtl@seg\endcsname\dtl@tmp
        \c@DTLpieroundvar
    }%
    \@for\dtl@row:=\dtl@piecutaways\do{%
      \expandafter\@dtl@set@off\dtl@row-\relax
    }%
    \let\dtl@start=\DTLstartangle
    \begin{tikzpicture}
      \DTLpieatbegintikz
```

```
\@sDTLforeach[#1]{#3}{#4}%
{%
  \expandafter\@dtl@seg\expandafter=
  \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname%
  \edef\dtl@start{%
    \csname dtl@sang@\romannumeral\@dtl@seg\endcsname}%
  \edef\dtl@extent{%
    \csname dtl@angle@\romannumeral\@dtl@seg\endcsname}%
  \dtladd{\dtl@endangle}{\dtl@start}{\dtl@extent}%
  \edef\dtl@angle{%
     \csname dtl@cut@angle@\romannumeral\@dtl@seg\endcsname}%
  \let\dtl@old=\dtl@angle
  \dtl@truncatedecimal\dtl@angle
  \ifnum\dtl@angle>180\relax
    \dtlsub{\dtl@angle}{\dtl@old}{360}%
    \dtl@truncatedecimal\dtl@angle
  \fi
  \edef\dtl@cutlen{%
    \csname dtl@cut@len@\romannumeral\@dtl@seg\endcsname
  }%
  \edef\@dtl@shift{(\dtl@angle:\dtl@cutlen)}%
  \dtlmul{\dtl@angle}{\dtl@extent}{0.5}%
  \dtladd{\dtl@midangle}{\dtl@angle}{\dtl@start}%
  \begin{scope}[shift={\@dtl@shift}]%
  \fill[color=\DTLgetpiesegmentcolor\@dtl@seg] (0,0) --
  (\dtl@start:\DTLradius)
  arc (\dtl@start:\dtl@endangle:\DTLradius) -- cycle;
  \ifdim\DTLpieoutlinewidth>0pt\relax
    \draw[color=\DTLpieoutlinecolor,%
          line width=\DTLpieoutlinewidth]
    (0,0) -- (\dtl@start:\DTLradius)
    arc (\dtl@start:\dtl@endangle:\DTLradius) -- cycle;
  \fi
  \dtl@truncatedecimal\dtl@midangle
  \ifDTLrotateinner
    \dtlifnumopenbetween{\dtl@midangle}{90}{270}%
    {%
      \let\@dtl@next\@firstoftwo
    }%
    {%
       \dtlifnumlt{\dtl@midangle}{-90}%
       {\let\@dtl@next\@firstoftwo}%
       {\let\@dtl@next\@secondoftwo}%
    }%
    \@dtl@next
    {%
      \dtlsub{\dtl@labelangle}{\dtl@midangle}{180}%
      \dtl@truncatedecimal\dtl@labelangle
      \edef\dtl@innernodeopt{anchor=east,rotate=\dtl@labelangle}%
    }%
```

```
  {%
    \edef\dtl@innernodeopt{anchor=west,rotate=\dtl@midangle}%
  }%
\else
  \edef\dtl@innernodeopt{anchor=center}%
\fi
\ifDTLrotateouter
   \dtlifnumopenbetween{\dtl@midangle}{90}{270}%
   {%
     \let\@dtl@next\@firstoftwo
   }%
   {%
      \dtlifnumlt{\dtl@midangle}{-90}%
      {\let\@dtl@next\@firstoftwo}%
      {\let\@dtl@next\@secondoftwo}%
   }%
   \@dtl@next
   {%
     \dtlsub{\dtl@labelangle}{\dtl@midangle}{180}%
     \dtl@truncatedecimal\dtl@labelangle
     \edef\dtl@outernodeopt{anchor=east,rotate=\dtl@labelangle}%
   }%
   {%
     \edef\dtl@outernodeopt{anchor=west,rotate=\dtl@midangle}%
   }%
\else
  \dtlifnumeq{\dtl@midangle}{45}
  {%
    \let\@dtl@next\@firstoftwo
  }%
  {%
    \dtlifnumgt{\dtl@midangle}{315}
    {%
      \let\@dtl@next\@firstoftwo
    }%
    {%
      \dtlifnumopenbetween{\dtl@midangle}{-45}{45}%
      {%
        \let\@dtl@next\@firstoftwo
      }%
      {%
        \let\@dtl@next\@secondoftwo
      }%
    }%
  }%
  \@dtl@next
  {%
    \edef\dtl@outernodeopt{anchor=west}%
  }%
  {%
```

```
\dtlifnumopenbetween{\dtl@midangle}{45}{135}%
{%
  \let\@dtl@next\@firstoftwo
}%
{%
  \dtlifnumeq{\dtl@midangle}{135}%
  {%
    \let\@dtl@next\@firstoftwo
  }%
  {%
    \let\@dtl@next\@secondoftwo
  }%
}%
\@dtl@next
{%
  \edef\dtl@outernodeopt{anchor=south}%
}%
{%
  \dtlifnumopenbetween{\dtl@midangle}{135}{225}%
  {%
    \let\@dtl@next\@firstoftwo
  }%
  {%
    \dtlifnumeq{\dtl@midangle}{225}%
    {%
      \let\@dtl@next\@firstoftwo
    }%
    {%
      \dtlifnumeq{\dtl@midangle}{-135}%
      {%
        \let\@dtl@next\@firstoftwo
      }%
      {%
        \dtlifnumlt{\dtl@midangle}{-135}%
        {%
          \let\@dtl@next\@firstoftwo
        }%
        {%
          \let\@dtl@next\@secondoftwo
        }%
      }%
    }%
  }%
  \@dtl@next
  {%
    \edef\dtl@outernodeopt{anchor=east}%
  }%
  {%
    \edef\dtl@outernodeopt{anchor=north}%
  }%
```

```
            }%
          }%
        \fi
        \edef\@dtl@dolabel{%
          \noexpand\draw (\dtl@midangle:\the\dtl@inneroffset)
             node[\dtl@innernodeopt]{%
           \noexpand\DTLdisplayinnerlabel{\noexpand\dtl@innerlabel}};
        }%
        \@dtl@dolabel
        \edef\@dtl@dolabel{%
          \noexpand\draw (\dtl@midangle:\the\dtl@outeroffset)
            node[\dtl@outernodeopt]{%
          \noexpand\DTLdisplayouterlabel{\noexpand\dtl@outerlabel}};
        }%
        \@dtl@dolabel
        \end{scope}%
      }%
      \DTLpieatendtikz
    \end{tikzpicture}%
  \fi
\egroup
}
\newcommand*{\dtl@computeangles}[2]{%
  \dtlifnumgt{\@dtl@start}{180}%
  {%
    \let\dtl@old=\@dtl@start
    \dtlsub{\@dtl@start}{\dtl@old}{360}%
  }%
  {}%
  \dtlifnumlt{\@dtl@start}{-180}%
  {%
    \let\dtl@old=\@dtl@start
    \dtladd{\@dtl@start}{\dtl@old}{360}%
  }%
  {}%
  \expandafter\edef\csname dtl@sang@\romannumeral#1\endcsname{%
     \@dtl@start}%
  \dtlmul{\dtl@angle}{360}{#2}%
  \let\dtl@old=\dtl@angle
  \dtldiv{\dtl@angle}{\dtl@old}{\dtl@total}%
  \expandafter\let\csname dtl@angle@\romannumeral#1\endcsname=\dtl@angle
  \let\dtl@old=\@dtl@start
  \dtladd{\@dtl@start}{\dtl@old}{\dtl@angle}%
  \expandafter\def\csname dtl@cut@angle@\romannumeral#1\endcsname{0}%
  \expandafter\def\csname dtl@cut@len@\romannumeral#1\endcsname{0cm}%
}
\def\@dtl@set@off#1-#2\relax{%
  \ifstrempty{#2}%
  {%
    \@@dtl@set@off{#1}%
```

```
  }%
  {%
    \@@dtl@set@offr#1-#2\relax
  }%
}
\newcommand*{\@@dtl@set@off}[1]{%
  \edef\dtl@old{\csname dtl@angle@\romannumeral#1\endcsname}%
  \dtlmul{\dtl@angle}{\dtl@old}{0.5}%
  \let\dtl@old=\dtl@angle
  \edef\dtl@sang{\csname dtl@sang@\romannumeral#1\endcsname}%
  \dtladd{\dtl@angle}{\dtl@old}{\dtl@sang}%
  \expandafter\edef\csname dtl@cut@angle@\romannumeral#1\endcsname{%
  \dtl@angle}%
  \expandafter\edef\csname dtl@cut@len@\romannumeral#1\endcsname{%
   \the\dtl@cutawayoffset}%
}
\newcount\@dtl@seg
\def\@@dtl@set@offr#1-#2-\relax{%
 \ifnum#1>#2\relax
   \PackageError{datapie}{Segment ranges must go in ascending order}{%
   Try #2-#1 instead of #1-#2}%
 \else
   \def\dtl@angle{0}%
   \@dtl@seg=#1\relax
   \whiledo{\not\(\@dtl@seg > #2\)}{%
    \let\dtl@old=\dtl@angle
    \edef\dtl@segang{\csname dtl@angle@\romannumeral\@dtl@seg\endcsname}%
    \dtladd{\dtl@angle}{\dtl@old}{\dtl@segang}%
    \advance\@dtl@seg by 1\relax
   }%
   \let\dtl@old=\dtl@angle
   \dtlmul{\dtl@angle}{\dtl@old}{0.5}%
   \edef\dtl@sang{\csname dtl@sang@\romannumeral#1\endcsname}%
   \let\dtl@old=\dtl@angle
   \dtladd{\dtl@angle}{\dtl@old}{\dtl@sang}%
   \@dtl@seg=#1\relax
   \whiledo{\not\(\@dtl@seg > #2\)}{%
   \expandafter
    \let\csname dtl@cut@angle@\romannumeral\@dtl@seg\endcsname
    =\dtl@angle
   \expandafter
    \edef\csname dtl@cut@len@\romannumeral\@dtl@seg\endcsname{%
    \the\dtl@cutawayoffset}
    \advance\@dtl@seg by 1\relax
   }%
 \fi
}
```

30.14 Rollback v2.32 (dataplot-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{dataplot}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{xkeyval}
\RequirePackage{tikz}
\RequirePackage{datatool}[=v2.32]
\usetikzlibrary{plotmarks}
\usetikzlibrary{plothandlers}
\usetikzlibrary{calc}
\newcommand*{\DTLplotstream}[4][\boolean{true}]{%
  \@sDTLforeach[#1]{#2}{\dtl@x=#3,\dtl@y=#4}{%
    \DTLconverttodecimal{\dtl@x}{\dtl@decx}%
    \DTLconverttodecimal{\dtl@y}{\dtl@decy}%
    \pgfplotstreampoint{\pgfpointxy{\dtl@decx}{\dtl@decy}}%
  }%
}
\newcommand*{\DTLplotmarks}{%
  \pgfuseplotmark{o},%
  \pgfuseplotmark{x},%
  \pgfuseplotmark{+},%
  \pgfuseplotmark{square},%
  \pgfuseplotmark{triangle},%
  \pgfuseplotmark{diamond},%
  \pgfuseplotmark{pentagon},%
  \pgfuseplotmark{asterisk},%
  \pgfuseplotmark{star}%
}
\newcommand*{\DTLplotmarkcolors}{%
  red,%
  green,%
  blue,%
  yellow,%
  magenta,%
  cyan,%
  orange,%
  black,%
  gray}
\newcommand*{\DTLplotlines}{%
  \pgfsetdash{}{0pt},% solid line
  \pgfsetdash{{10pt}{5pt}}{0pt},%
  \pgfsetdash{{5pt}{5pt}}{0pt},%
  \pgfsetdash{{1pt}{5pt}}{0pt},%
  \pgfsetdash{{5pt}{5pt}{1pt}{5pt}}{0pt},%
  \pgfsetdash{{1pt}{3pt}}{0pt},%
}
\newcommand*{\DTLplotlinecolors}{%
  red,%
  green,%
  blue,%
  yellow,%
  magenta,%
```

```
  cyan,%
  orange,%
  black,%
  gray}
\newlength\DTLplotwidth
\setlength\DTLplotwidth{4in}
\newlength\DTLplotheight
\setlength\DTLplotheight{4in}
\newlength\DTLticklength
\setlength\DTLticklength{5pt}
\newlength\DTLminorticklength
\setlength\DTLminorticklength{2pt}
\newlength\DTLticklabeloffset
\setlength\DTLticklabeloffset{8pt}
\newlength\dtl@xticlabelheight
\newlength\dtl@yticlabelwidth
\newlength\DTLmintickgap
\setlength\DTLmintickgap{20pt}
\newlength\DTLminminorticklength
\setlength\DTLminminorticklength{5pt}
\newcounter{DTLplotroundXvar}
\setcounter{DTLplotroundXvar}{2}
\newcounter{DTLplotroundYvar}
\setcounter{DTLplotroundYvar}{2}
\newif\ifDTLxaxis
\DTLxaxistrue
\newcommand*{\DTLXAxisStyle}{-}
\newif\ifDTLyaxis
\DTLyaxistrue
\newcommand*{\DTLYAxisStyle}{-}
\newcommand*{\DTLmajorgridstyle}{color=gray,-}
\newcommand*{\DTLminorgridstyle}{color=gray,loosely dotted}
\newif\ifDTLxticsin
\DTLxticsintrue
\newif\ifDTLyticsin
\DTLyticsintrue
\newcount\dtl@legendsetting
\newlength\DTLlegendxoffset
\setlength\DTLlegendxoffset{10pt}
\newlength\DTLlegendyoffset
\setlength\DTLlegendyoffset{10pt}
\newcommand*{\DTLformatlegend}[1]{%
\setlength{\fboxrule}{1.1pt}%
\fcolorbox{black}{white}{#1}}
\newif\ifDTLshowmarkers
\DTLshowmarkerstrue
\newif\ifDTLshowlines
\DTLshowlinesfalse
\newcommand*{\DTLplotatbegintikz}{}
\newcommand*{\@dtlplothandlermark}[1]{%
```

```
  \pgfplothandlermark
  {%
    \pgfmathparse{1/\dtl@scale@x}%
    \pgftransformxscale{\pgfmathresult}%
    \pgfmathparse{1/\dtl@scale@y}%
    \pgftransformyscale{\pgfmathresult}%
   #1%
   }%
}
\newcommand*{\dtlplothandlermark}[1]{%
   \PackageWarning{dataplot}{\string\dtlplothandlermark\space
     found outside \string\DTLplot}%
  \pgfplothandlermark{#1}%
}
\newcommand*{\DTLplotatendtikz}{}
\define@key{dataplot}{x}{%
\def\dtl@xkey{#1}}
\define@key{dataplot}{y}{%
\def\dtl@ykey{#1}}
\define@key{dataplot}{markcolors}{%
\def\DTLplotmarkcolors{#1}}
\define@key{dataplot}{linecolors}{%
\def\DTLplotlinecolors{#1}}
\define@key{dataplot}{colors}{%
\def\DTLplotmarkcolors{#1}%
\def\DTLplotlinecolors{#1}}
\define@key{dataplot}{marks}{%
\def\DTLplotmarks{#1}}
\define@key{dataplot}{lines}{%
\def\DTLplotlines{#1}}
\define@key{dataplot}{width}{%
\setlength\DTLplotwidth{#1}}
\define@key{dataplot}{height}{%
\setlength\DTLplotheight{#1}}
\define@choicekey{dataplot}{style}[\val\nr]{both,lines,markers}{%
\ifcase\nr\relax
 \DTLshowlinestrue
 \DTLshowmarkerstrue
\or
 \DTLshowlinestrue
 \DTLshowmarkersfalse
\or
 \DTLshowmarkerstrue
 \DTLshowlinesfalse
\fi}
\define@choicekey{dataplot}{axes}[\val\nr]{both,x,y,none}[both]{%
\ifcase\nr\relax
 % both
 \DTLxaxistrue
 \DTLxticstrue
```

```
 \DTLyaxistrue
 \DTLyticstrue
\or % x
 \DTLxaxistrue
 \DTLxticstrue
 \DTLyaxisfalse
 \DTLyticsfalse
\or % y
 \DTLxaxisfalse
 \DTLxticsfalse
 \DTLyaxistrue
 \DTLyticstrue
\or % none
 \DTLxaxisfalse
 \DTLxticsfalse
 \DTLyaxisfalse
 \DTLyticsfalse
\fi
}
\define@boolkey{dataplot}[DTL]{box}[true]{}
\DTLboxfalse
\define@boolkey{dataplot}[DTL]{xtics}[true]{}
\DTLxticstrue
\define@boolkey{dataplot}[DTL]{ytics}[true]{}
\DTLyticstrue
\define@boolkey{dataplot}[DTL]{xminortics}[true]{%
\ifDTLxminortics \DTLxticstrue\fi}
\DTLxminorticsfalse
\define@boolkey{dataplot}[DTL]{yminortics}[true]{%
\ifDTLyminortics \DTLyticstrue\fi}
\DTLyminorticsfalse
\define@boolkey{dataplot}[DTL]{grid}[true]{}
\define@choicekey{dataplot}{xticdir}[\val\nr]{in,out}{%
\ifcase\nr\relax
 \DTLxticsintrue
\or
 \DTLxticsinfalse
\fi
}
\define@choicekey{dataplot}{yticdir}[\val\nr]{in,out}{%
\ifcase\nr\relax
 \DTLyticsintrue
\or
 \DTLyticsinfalse
\fi
}
\define@choicekey{dataplot}{ticdir}[\val\nr]{in,out}{%
\ifcase\nr\relax
 \DTLxticsintrue
 \DTLyticsintrue
```

```
\or
 \DTLxticsinfalse
 \DTLyticsinfalse
\fi
}
\define@key{dataplot}{bounds}{%
\def\dtl@bounds{#1}}
\let\dtl@bounds=\relax
\define@key{dataplot}{minx}{%
\def\dtl@minx{#1}}
\let\dtl@minx=\relax
\define@key{dataplot}{maxx}{%
\def\dtl@maxx{#1}}
\let\dtl@maxx=\relax
\define@key{dataplot}{miny}{%
\def\dtl@miny{#1}}
\let\dtl@miny=\relax
\define@key{dataplot}{maxy}{%
\def\dtl@maxy{#1}}
\let\dtl@maxy=\relax
\define@key{dataplot}{xticpoints}{%
\def\dtl@xticlist{#1}\DTLxticstrue\DTLxaxistrue}
\let\dtl@xticlist=\relax
\define@key{dataplot}{yticpoints}{%
\def\dtl@yticlist{#1}\DTLyticstrue\DTLyaxistrue}
\let\dtl@yticlist=\relax
\define@key{dataplot}{xticgap}{\def\dtl@xticgap{#1}%
\DTLxticstrue\DTLxaxistrue}
\let\dtl@xticgap=\relax
\define@key{dataplot}{yticgap}{\def\dtl@yticgap{#1}%
\DTLyticstrue\DTLyaxistrue}
\let\dtl@yticgap=\relax
\define@key{dataplot}{xticlabels}{%
\def\dtl@xticlabels{#1}\DTLxticstrue\DTLxaxistrue}
\let\dtl@xticlabels=\relax
\define@key{dataplot}{yticlabels}{%
\def\dtl@yticlabels{#1}\DTLyticstrue\DTLyaxistrue}
\let\dtl@yticlabels=\relax
\define@key{dataplot}{xlabel}{%
\def\dtl@xlabel{#1}}
\let\dtl@xlabel=\relax
\define@key{dataplot}{ylabel}{%
\def\dtl@ylabel{#1}}
\let\dtl@ylabel=\relax
\define@choicekey{dataplot}{legend}[\val\nr]{none,north,northeast,%
east,southeast,south,southwest,west,northwest}[northeast]{%
\dtl@legendsetting=\nr\relax
}
\define@key{dataplot}{legendlabels}{\def\dtl@legendlabels{#1}}
\newcommand*{\DTLplot}[3][\boolean{true}]{%
```

```
\bgroup
  \let\dtl@xkey=\relax
  \let\dtl@ykey=\relax
  \let\dtl@legendlabels=\relax
  \setkeys{dataplot}{#3}%
  \let\dtl@plotmarklist=\DTLplotmarks
  \let\dtl@plotlinelist=\DTLplotlines
  \let\dtl@plotmarkcolorlist=\DTLplotmarkcolors
  \let\dtl@plotlinecolorlist=\DTLplotlinecolors
  \def\dtl@legend{}%
  \ifx\dtl@legendlabels\relax
   \edef\dtl@legendlabels{#2}%
  \fi
  \ifx\dtl@xkey\relax
    \PackageError{dataplot}{Missing x setting for
      \string\DTLplot}{}%
  \else
    \ifx\dtl@ykey\relax
      \PackageError{dataplot}{Missing y setting for
      \string\DTLplot}{}%
    \else
      \ifx\dtl@bounds\relax
        \DTLcomputebounds[#1]{#2}{\dtl@xkey}{\dtl@ykey}
            {\DTLminX}{\DTLminY}{\DTLmaxX}{\DTLmaxY}%
        \ifx\dtl@minx\relax
        \else
           \let\DTLminX=\dtl@minx
        \fi
        \ifx\dtl@maxx\relax
        \else
           \let\DTLmaxX=\dtl@maxx
        \fi
        \ifx\dtl@miny\relax
        \else
           \let\DTLminY=\dtl@miny
        \fi
        \ifx\dtl@maxy\relax
        \else
           \let\DTLmaxY=\dtl@maxy
        \fi
      \else
        \expandafter\dtl@getbounds\dtl@bounds\@nil
      \fi
      \@dtl@tmpcount=\DTLplotwidth
      \divide\@dtl@tmpcount by 65536\relax
      \dtlsub{\dtl@dx}{\DTLmaxX}{\DTLminX}%
      \dtldiv{\dtl@scale@x}{\number\@dtl@tmpcount}{\dtl@dx}%
      \dtlmul{\dtl@offset@x}{-\dtl@scale@x}{\DTLminX}%
      \@dtl@tmpcount=\DTLplotheight
      \divide\@dtl@tmpcount by 65536\relax
```

```
 \dtlsub{\dtl@dy}{\DTLmaxY}{\DTLminY}%
 \dtldiv{\dtl@scale@y}{\number\@dtl@tmpcount}{\dtl@dy}%
 \dtlmul{\dtl@offset@y}{-\dtl@scale@y}{\DTLminY}%
\ifDTLxtics
  \ifx\dtl@xticlist\relax
    \ifx\dtl@xticgap\relax
    \dtlsub{\dtl@mingap}{\number\DTLmintickgap}{\dtl@offset@x}%
    \dtldiv{\dtl@mingap}{\dtl@mingap}{\dtl@scale@x}%
    \dtldiv{\dtl@mingap}{\dtl@mingap}{65536}%
      \dtl@constructticklist\DTLminX\DTLmaxX
        \dtl@mingap\dtl@xticlist
    \else
      \DTLifFPopenbetween{0}{\DTLminX}{\DTLmaxX}{%
        \dtl@constructticklistwithgapz
          \DTLminX\DTLmaxX\dtl@xticlist\dtl@xticgap}{%
        \dtl@constructticklistwithgap
          \DTLminX\DTLmaxX\dtl@xticlist\dtl@xticgap}%
    \fi
  \fi
  \let\dtl@xminorticlist\@empty
  \ifDTLxminortics
    \let\dtl@prevtick=\relax
    \@for\dtl@nexttick:=\dtl@xticlist\do{%
      \ifx\dtl@prevtick\relax
      \else
        \dtl@constructminorticklist
           \dtl@prevtick\dtl@nexttick\dtl@scale@x\dtl@xminorticlist
      \fi
      \let\dtl@prevtick=\dtl@nexttick
    }%
  \fi
  \ifx\dtl@xticlabels\relax
    \settoheight{\dtl@xticlabelheight}{\dtl@xticlist}%
  \else
    \settoheight{\dtl@xticlabelheight}{\dtl@xticlabels}%
   \fi
\else
  \setlength{\dtl@xticlabelheight}{0pt}%
\fi
\setlength{\dtl@yticlabelwidth}{0pt}%
\ifDTLytics
  \ifx\dtl@yticlist\relax
    \ifx\dtl@yticgap\relax
    \dtlsub{\dtl@mingap}{\number\DTLmintickgap}{\dtl@offset@y}%
    \dtldiv{\dtl@mingap}{\dtl@mingap}{\dtl@scale@y}%
    \dtldiv{\dtl@mingap}{\dtl@mingap}{65536}%
    \dtl@constructticklist\DTLminY\DTLmaxY
      \dtl@mingap\dtl@yticlist
    \else
      \DTLifFPopenbetween{0}{\DTLminY}{\DTLmaxY}{%
```

```
          \dtl@constructticklistwithgapz
            \DTLminY\DTLmaxY\dtl@yticlist\dtl@yticgap}{%
          \dtl@constructticklistwithgap
            \DTLminY\DTLmaxY\dtl@yticlist\dtl@yticgap}%
      \fi
    \fi
    \let\dtl@yminorticlist\@empty
    \ifDTLyminortics
      \let\dtl@prevtick=\relax
      \@for\dtl@nexttick:=\dtl@yticlist\do{%
        \ifx\dtl@prevtick\relax
        \else
          \dtl@constructminorticklist
             \dtl@prevtick\dtl@nexttick\dtl@scale@y\dtl@yminorticlist
        \fi
        \let\dtl@prevtick=\dtl@nexttick
      }%
    \fi
    \ifx\dtl@ylabel\relax
    \else
      \ifx\dtl@yticlabels\relax
        \@for\dtl@thislabel:=\dtl@yticlist\do{%
          \dtlround{\dtl@thislabel}{\dtl@thislabel}
                   {\c@DTLplotroundYvar}%
          \settowidth{\dtl@tmplength}{\dtl@thislabel}%
          \ifdim\dtl@tmplength>\dtl@yticlabelwidth
            \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
          \fi
        }%
      \else
        \@for\dtl@thislabel:=\dtl@yticlabels\do{%
          \settowidth{\dtl@tmplength}{\dtl@thislabel}%
          \ifdim\dtl@tmplength>\dtl@yticlabelwidth
            \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
          \fi
        }%
      \fi
    \fi
  \fi
    \begin{tikzpicture}
    \pgfsetxvec{\pgfpoint{1pt}{0pt}}%
    \pgfsetyvec{\pgfpoint{0pt}{1pt}}%
\begin{scope}
\pgftransformcm{\dtl@scale@x}{0}{0}{\dtl@scale@y}%
  {\pgfpoint{\dtl@offset@x pt}{\dtl@offset@y pt}}%
    \let\dtlplothandlermark\@dtlplothandlermark
    \DTLplotatbegintikz
    \ifDTLbox
      \draw (\DTLminX,\DTLminY) -- (\DTLmaxX,\DTLminY) --
          (\DTLmaxX,\DTLmaxY) -- (\DTLminX,\DTLmaxY) --
```

```
        cycle;
    \else
      \ifDTLxaxis
        \expandafter\draw\expandafter[\DTLXAxisStyle]
        (\DTLminX,\DTLminY) -- (\DTLmaxX,\DTLminY);
      \fi
      \ifDTLyaxis
        \expandafter\draw\expandafter[\DTLYAxisStyle]
        (\DTLminX,\DTLminY) -- (\DTLminX,\DTLmaxY);
      \fi
    \fi
    \ifDTLgrid
      \ifDTLxminortics
        \@for\dtl@thistick:=\dtl@xminorticlist\do{%
          \expandafter\draw\expandafter[\DTLminorgridstyle]
         (\dtl@thistick,\DTLminY) -- (\dtl@thistick,\DTLmaxY);
        }%
      \fi
      \ifDTLyminortics
        \@for\dtl@thistick:=\dtl@yminorticlist\do{%
          \expandafter\draw\expandafter[\DTLminorgridstyle]
         (\DTLminX,\dtl@thistick) -- (\DTLmaxX,\dtl@thistick);
        }%
      \fi
      \@for\dtl@thistick:=\dtl@xticlist\do{%
        \expandafter\draw\expandafter[\DTLmajorgridstyle]
         (\dtl@thistick,\DTLminY) -- (\dtl@thistick,\DTLmaxY);
      }%
      \@for\dtl@thistick:=\dtl@yticlist\do{%
        \expandafter\draw\expandafter[\DTLmajorgridstyle]
         (\DTLminX,\dtl@thistick) -- (\DTLmaxX,\dtl@thistick);
      }%
    \fi
    \ifDTLxtics
      \dtlsub{\dtl@ticklength}{\number\DTLticklength}{-\dtl@offset@y}%
      \dtldiv{\dtl@ticklength}{\dtl@ticklength}{\dtl@scale@y}%
      \dtldiv{\dtl@ticklength}{\dtl@ticklength}{65536}%
      \addtolength\dtl@xticlabelheight{\DTLticklabeloffset}%
      \dtlsub{\dtl@ticlabeloffset}{\number\dtl@xticlabelheight}{-
\dtl@offset@y}%
      \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{\dtl@scale@y}%
      \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{65536}%
      \@for\dtl@thistick:=\dtl@xticlist\do{%
        \let\dtl@thisticklabel\dtl@thistick
        \ifx\dtl@xticlabels\relax
          \dtlround{\dtl@thislabel}{\dtl@thistick}
                   {\c@DTLplotroundXvar}%
        \else
          \dtl@chopfirst\dtl@xticlabels\dtl@thislabel\dtl@rest
          \let\dtl@xticlabels=\dtl@rest
```

```
            \fi
            \ifDTLxticsin
              \draw (\dtl@thistick,\DTLminY) -- ++(0,\dtl@ticklength);
              \draw (\dtl@thistick,\DTLminY)
                  ++ (0,-\dtl@ticlabeloffset) node {\dtl@thislabel};
            \else
              \draw (\dtl@thistick,\DTLminY) -- ++(0,-\dtl@ticklength)
                  ++ (0,-\dtl@ticlabeloffset) node {\dtl@thislabel};
            \fi
            \ifDTLbox
              \ifDTLxticsin
                \draw (\dtl@thistick,\DTLmaxY) -- ++(0,-\dtl@ticklength);
              \else
              \draw (\dtl@thistick,\DTLmaxY) -- ++(0,\dtl@ticklength);
              \fi
            \fi
          }%
       \fi
      \ifx\dtl@xlabel\relax
      \else
         \dtladd{\dtl@x}{\number\baselineskip}{\dtl@offset@y}%
         \dtldiv{\dtl@x}{\dtl@x}{\dtl@scale@y}%
         \dtldiv{\dtl@x}{\dtl@x}{65536}%
         \dtladd{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{\dtl@x}%
         \dtlmul{\dtl@x}{\dtl@dx}{0.5}%
         \draw (\DTLminX,\DTLminY) ++(\dtl@x,-\dtl@ticlabeloffset)
            node[anchor=north] {\dtl@xlabel};
      \fi
      \ifDTLxminortics
         \dtlsub{\dtl@ticklength}{\number\DTLminorticklength}{-
\dtl@offset@y}%
         \dtldiv{\dtl@ticklength}{\dtl@ticklength}{\dtl@scale@y}%
         \dtldiv{\dtl@ticklength}{\dtl@ticklength}{65536}%
         \@for\dtl@thistick:=\dtl@xminorticlist\do{%
           \ifDTLxticsin
             \draw (\dtl@thistick,\DTLminY) -- ++(0,\dtl@ticklength);
             \draw (\dtl@thistick,\DTLminY)
                 ++ (0,-\dtl@ticlabeloffset) node[anchor=north] {\dtl@thislabel};
           \else
             \draw (\dtl@thistick,\DTLminY) -- ++(0,-\dtl@ticklength)
                 ++ (0,-\dtl@ticlabeloffset) node[anchor=north] {\dtl@thislabel};
           \fi
           \ifDTLbox
             \ifDTLxticsin
               \draw (\dtl@thistick,\DTLmaxY) -- ++(0,-\dtl@ticklength);
             \else
             \draw (\dtl@thistick,\DTLmaxY) -- ++(0,\dtl@ticklength);
             \fi
           \fi
         }%
```

```
      \fi
      \ifDTLytics
         \dtlsub{\dtl@ticklength}{\number\DTLticklength}{-\dtl@offset@x}%
         \dtldiv{\dtl@ticklength}{\dtl@ticklength}{\dtl@scale@x}%
         \dtldiv{\dtl@ticklength}{\dtl@ticklength}{65536}%
         \dtladd{\dtl@ticlabeloffset}{\number\DTLticklabeloffset}{0}%
         \dtlsub{\dtl@ticlabeloffset}{\number\DTLticklabeloffset}{-
\dtl@offset@x}%
         \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{\dtl@scale@x}%
         \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{65536}%
         \@for\dtl@thistick:=\dtl@yticlist\do{%
           \let\dtl@thisticklabel\dtl@thistick
           \ifx\dtl@yticlabels\relax
             \dtlround{\dtl@thislabel}{\dtl@thistick}
                      {\c@DTLplotroundXvar}%
           \else
             \dtl@chopfirst\dtl@yticlabels\dtl@thislabel\dtl@rest
             \let\dtl@yticlabels=\dtl@rest
           \fi
           \ifDTLyticsin
             \draw (\DTLminX,\dtl@thistick) -- ++(\dtl@ticklength,0);
             \draw (\DTLminX,\dtl@thistick)
                 ++ (-\dtl@ticlabeloffset,0) node[anchor=east] {\dtl@thislabel};
           \else
             \draw (\DTLminX,\dtl@thistick) -- ++(-\dtl@ticklength,0)
                 ++ (-\dtl@ticlabeloffset,0) node[anchor=east] {\dtl@thislabel};
           \fi
           \ifDTLbox
             \ifDTLyticsin
               \draw (\DTLmaxX,\dtl@thistick) -- ++(-\dtl@ticklength,0);
             \else
             \draw (\DTLmaxX,\dtl@thistick) -- ++(\dtl@ticklength,0);
             \fi
           \fi
         }%
      \fi
      \ifx\dtl@ylabel\relax
      \else
         \setlength{\dtl@tmplength}{\baselineskip}%
         \addtolength{\dtl@tmplength}{\dtl@yticlabelwidth}%
         \addtolength{\dtl@tmplength}{\DTLticklabeloffset}%
         \dtlsub{\dtl@ticlabeloffset}{\number\dtl@tmplength}{-\dtl@offset@x}%
         \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{\dtl@scale@x}%
         \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{65536}%
         \dtlmul{\dtl@y}{\dtl@dy}{0.5}%
         \draw (\DTLminX,\DTLminY) ++(-\dtl@ticlabeloffset,\dtl@y)
            node[rotate=90,anchor=south] {\dtl@ylabel};
      \fi
      \ifDTLyminortics
         \dtlsub{\dtl@ticklength}{\number\DTLminorticklength}{-
```

```
\dtl@offset@x}%
         \dtldiv{\dtl@ticklength}{\dtl@ticklength}{\dtl@scale@x}%
         \dtldiv{\dtl@ticklength}{\dtl@ticklength}{65536}%
         \@for\dtl@thistick:=\dtl@yminorticlist\do{%
           \ifDTLyticsin
             \draw (\DTLminX,\dtl@thistick) -- ++(\dtl@ticklength,0);
           \else
             \draw (\DTLminX,\dtl@thistick) -- ++(-\dtl@ticklength,0);
           \fi
           \ifDTLbox
             \ifDTLyticsin
               \draw (\DTLmaxX,\dtl@thistick) -- ++(-\dtl@ticklength,0);
             \else
             \draw (\DTLmaxX,\dtl@thistick) -- ++(\dtl@ticklength,0);
             \fi
           \fi
         }%
       \fi
   \end{scope}
        \@for\dtl@thisdb:=#2\do{%
          \ifx\dtl@plotmarkcolorlist\@empty
            \let\dtl@plotmarkcolorlist=\DTLplotmarkcolors
          \fi
          \dtl@chopfirst\dtl@plotmarkcolorlist\dtl@thisplotmarkcolor
             \dtl@remainder
          \let\dtl@plotmarkcolorlist=\dtl@remainder
          \ifDTLshowmarkers
            \ifx\dtl@plotmarklist\@empty
               \let\dtl@plotmarklist=\DTLplotmarks
            \fi
            \dtl@chopfirst\dtl@plotmarklist\dtl@thisplotmark
               \dtl@remainder
            \let\dtl@plotmarklist=\dtl@remainder
            \ifx\dtl@thisplotmark\relax
              \let\dtl@mark=\relax
            \else
              \expandafter\toks@\expandafter{\dtl@thisplotmark}%
              \ifx\dtl@thisplotmarkcolor\@empty
                \edef\dtl@mark{\the\toks@}%
              \else
                \edef\dtl@mark{%
                    \noexpand\color{\dtl@thisplotmarkcolor}%
                  \the\toks@}%
              \fi
            \fi
          \else
            \let\dtl@mark=\relax
          \fi
          \ifx\dtl@plotlinecolorlist\@empty
            \let\dtl@plotlinecolorlist=\DTLplotlinecolors
```

```
  \fi
  \dtl@chopfirst\dtl@plotlinecolorlist\dtl@thisplotlinecolor
     \dtl@remainder
  \let\dtl@plotlinecolorlist=\dtl@remainder
  \ifDTLshowlines
    \ifx\dtl@plotlinelist\@empty
       \let\dtl@plotlinelist=\DTLplotlines
    \fi
    \dtl@chopfirst\dtl@plotlinelist\dtl@thisplotline
       \dtl@remainder
    \let\dtl@plotlinelist=\dtl@remainder
    \expandafter\ifx\dtl@thisplotline\relax
    \let\dtl@linestyle=\relax
  \else
    \expandafter\toks@\expandafter{\dtl@thisplotline}%
    \ifx\dtl@thisplotlinecolor\@empty
      \edef\dtl@linestyle{\the\toks@}%
    \else
      \edef\dtl@linestyle{%
          \noexpand\color{\dtl@thisplotlinecolor}%
        \the\toks@}%
    \fi
  \fi
\else
  \let\dtl@linestyle=\relax
\fi
\ifnum\dtl@legendsetting>0\relax
  \dtl@chopfirst\dtl@legendlabels\dtl@thislabel\dtl@rest
  \let\dtl@legendlabels=\dtl@rest
  \expandafter\toks@\expandafter{\dtl@mark}%
  \expandafter\@dtl@toks\expandafter{\dtl@linestyle}%
  \edef\dtl@addtolegend{\noexpand\DTLaddtoplotlegend
    {\the\toks@}{\the\@dtl@toks}{\dtl@thislabel}}%
  \dtl@addtolegend
\fi
\def\dtl@stream{\pgfplotstreamstart}%
\@sDTLforeach[#1]{\dtl@thisdb}{\dtl@x=\dtl@xkey,%
    \dtl@y=\dtl@ykey}{%
  \DTLconverttodecimal{\dtl@x}{\dtl@decx}%
  \DTLconverttodecimal{\dtl@y}{\dtl@decy}%
  \ifthenelse{%
    \DTLisclosedbetween{\dtl@x}{\DTLminX}{\DTLmaxX}%
    \and
    \DTLisclosedbetween{\dtl@y}{\DTLminY}{\DTLmaxY}%
    }%
    {%
      \expandafter\toks@\expandafter{\dtl@stream}%
      \dtlmul{\dtl@decx}{\dtl@decx}{\dtl@scale@x}%
      \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
      \dtlround{\dtl@decx}{\dtl@decx}{1}%
```

```
        \dtlmul{\dtl@decy}{\dtl@decy}{\dtl@scale@y}%
        \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
        \dtlround{\dtl@decy}{\dtl@decy}{1}%
        \edef\dtl@stream{\the\toks@
          \noexpand\pgfplotstreampoint
           {\noexpand\pgfpointxy{\dtl@decx}{\dtl@decy}}}%
      }{}%
    }%
    \expandafter\toks@\expandafter{\dtl@stream}%
    \edef\dtl@stream{\the\toks@\noexpand\pgfplotstreamend}%
    \ifx\dtl@linestyle\relax
    \else
      \begin{scope}
      \dtl@linestyle
      \pgfplothandlerlineto
      \dtl@stream
      \pgfusepath{stroke}
      \end{scope}
    \fi
    \ifx\dtl@mark\relax
    \else
      \begin{scope}
      \pgfplothandlermark{\dtl@mark}%
      \dtl@stream
      \pgfusepath{stroke}
      \end{scope}
    \fi
    }%
    \ifcase\dtl@legendsetting
  % none
    \or % north
     \dtlmul{\dtl@decx}{\dtl@dx}{0.5}%
     \dtladd{\dtl@decx}{\DTLminX}{\dtl@decx}%
     \dtlmul{\dtl@decx}{\dtl@decx}{\dtl@scale@x}%
     \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
     \dtlmul{\dtl@decy}{\DTLmaxY}{\dtl@scale@y}%
     \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
     \draw (\dtl@decx,\dtl@decy) ++(0,-\DTLlegendyoffset)
        node[anchor=north]
        {\DTLformatlegend
          {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
        };
    \or % north east
     \dtlmul{\dtl@decx}{\DTLmaxX}{\dtl@scale@x}%
     \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
     \dtlmul{\dtl@decy}{\DTLmaxY}{\dtl@scale@y}%
     \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
     \draw (\dtl@decx,\dtl@decy) ++(-\DTLlegendxoffset,-\DTLlegendyoffset)
        node[anchor=north east]
        {\DTLformatlegend
```

```
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % east
 \dtlmul{\dtl@decy}{\dtl@dy}{0.5}%
 \dtladd{\dtl@decy}{\DTLminY}{\dtl@decy}%
 \dtlmul{\dtl@decy}{\dtl@decy}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
 \dtlmul{\dtl@decx}{\DTLmaxX}{\dtl@scale@x}%
 \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
 \draw (\dtl@decx,\dtl@decy) ++(-\DTLlegendxoffset,0)
    node[anchor=east]
    {\DTLformatlegend
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % south east
 \dtlmul{\dtl@decx}{\DTLmaxX}{\dtl@scale@x}%
 \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
 \dtlmul{\dtl@decy}{\DTLminY}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
 \draw (\dtl@decx,\dtl@decy) ++(-\DTLlegendxoffset,\DTLlegendyoffset)
    node[anchor=south east]
    {\DTLformatlegend
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % south
 \dtlmul{\dtl@decx}{\dtl@dx}{0.5}%
 \dtladd{\dtl@decx}{\DTLminX}{\dtl@decx}%
 \dtlmul{\dtl@decx}{\dtl@decx}{\dtl@scale@x}%
 \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
 \dtlmul{\dtl@decy}{\DTLminY}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
 \draw (\dtl@decx,\dtl@decy) ++(0,\DTLlegendyoffset)
    node[anchor=south]
    {\DTLformatlegend
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % south west
 \dtlmul{\dtl@decx}{\DTLminX}{\dtl@scale@x}%
 \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
 \dtlmul{\dtl@decy}{\DTLminY}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
 \draw (\dtl@decx,\dtl@decy) ++(\DTLlegendxoffset,\DTLlegendyoffset)
    node[anchor=south west]
    {\DTLformatlegend
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % west
 \dtlmul{\dtl@decy}{\dtl@dy}{0.5}%
 \dtladd{\dtl@decy}{\DTLminY}{\dtl@decy}%
 \dtlmul{\dtl@decy}{\dtl@decy}{\dtl@scale@y}%
```

```
          \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
          \dtlmul{\dtl@decx}{\DTLminX}{\dtl@scale@x}%
          \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
          \draw (\dtl@decx,\dtl@decy) ++(\DTLlegendxoffset,0)
             node[anchor=west]
             {\DTLformatlegend
               {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
             };
         \or % north west
          \dtlmul{\dtl@decx}{\DTLminX}{\dtl@scale@x}%
          \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
          \dtlmul{\dtl@decy}{\DTLmaxY}{\dtl@scale@y}%
          \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
          \draw (\dtl@decx,\dtl@decy) ++(\DTLlegendxoffset,-\DTLlegendyoffset)
             node[anchor=north west]
             {\DTLformatlegend
               {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
             };
         \fi
        \pgftransformcm{\dtl@scale@x}{0}{0}{\dtl@scale@y}%
          {\pgfpoint{\dtl@offset@x pt}{\dtl@offset@y pt}}%
        \let\dtlplothandlermark\@dtlplothandlermark
         \DTLplotatendtikz
        \end{tikzpicture}
    \fi
  \fi
\egroup
}
\def\dtl@getbounds#1,#2,#3,#4\@nil{%
\def\DTLminX{#1}%
\def\DTLminY{#2}%
\def\DTLmaxX{#3}%
\def\DTLmaxY{#4}%
\dtlifnumgt{\DTLminX}{\DTLmaxX}
{%
 \PackageError{dataplot}{Min X > Max X in bounds #1,#2,#3,#4}{%
 The bounds must be specified as minX,minY,maxX,maxY}%
}{}%
\dtlifnumgt{\DTLminY}{\DTLmaxY}
{%
 \PackageError{dataplot}{Min Y > Max Y in bounds #1,#2,#3,#4}{%
 The bounds must be specified as minX,minY,maxX,maxY}%
}{}%
}
\newcommand*{\dtl@constructticklist}[4]{%
  \DTLifFPopenbetween{0}{#1}{#2}%
  {%
    \dtlsub{\@dtl@width}{0}{#1}%
    \dtldiv{\@dtl@neggap}{\@dtl@width}{10}%
    \dtlifnumlt{\@dtl@neggap}{#3}%
```

```
    {%
      \edef\@dtl@neggap{#3}%
    }%
    {}%
    \dtldiv{\@dtl@posgap}{#2}{10}%
    \dtlifnumlt{\@dtl@posgap}{#3}%
    {%
      \edef\@dtl@posgap{#3}%
    }%
    {}%
    \dtlmax{\@dtl@gap}{\@dtl@neggap}{\@dtl@posgap}%
    \dtlifnumgt{\@dtl@gap}{\@dtl@width}%
    {}%
    {%
      \dtl@constructticklistwithgapz{#1}{#2}{#4}{\@dtl@gap}%
    }%
  }%
  {%
    \dtlsub{\@dtl@width}{#2}{#1}%
    \dtldiv{\@dtl@gap}{\@dtl@width}{10}%
    \dtlifnumlt{\@dtl@gap}{#3}%
    {%
      \dtlifnumgt{#3}{\@dtl@width}%
      {%
         \def#4{#1,#2}%
      }%
      {%
        \dtl@constructticklistwithgap{#1}{#2}{#4}{#3}%
      }
    }%
    {%
      \dtl@constructticklistwithgap{#1}{#2}{#4}{\@dtl@gap}%
    }%
  }%
}
\newcommand*{\dtl@constructticklistwithgap}[4]{%
\edef\@dtl@thistick{#1}%
\edef#3{#1}%
\dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
\whiledo{\DTLisFPopenbetween{\@dtl@thistick}{#1}{#2}}{%
  \expandafter\toks@\expandafter{\@dtl@thistick}%
  \edef#3{#3,\the\toks@}%
  \dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
}%
\expandafter\toks@\expandafter{#2}%
\edef#3{#3,\the\toks@}%
}
\newcommand*{\dtl@constructticklistwithgapz}[4]{%
  \edef\@dtl@thistick{0}%
  \edef#3{0}%
```

```
  \dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
  \whiledo{\DTLisFPopenbetween{\@dtl@thistick}{0}{#2}}%
  {%
    \expandafter\toks@\expandafter{\@dtl@thistick}%
    \edef#3{#3,\the\toks@}%
    \dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
  }%
  \expandafter\toks@\expandafter{#2}%
  \edef#3{#3,\the\toks@}%
  \dtlifnumeq{#1}{0}%
  {}%
  {%
    \edef\@dtl@thistick{0}%
    \dtlsub{\@dtl@thistick}{\@dtl@thistick}{#4}%
    \whiledo{\DTLisFPopenbetween{\@dtl@thistick}{#1}{0}}%
    {%
      \expandafter\toks@\expandafter{\@dtl@thistick}%
      \edef#3{\the\toks@,#3}%
      \dtlsub{\@dtl@thistick}{\@dtl@thistick}{#4}%
    }%
    \expandafter\toks@\expandafter{#1}%
    \edef#3{\the\toks@,#3}%
  }%
}
\newcommand*{\dtl@constructminorticklist}[4]{%
  \dtlsub{\@dtl@width}{#2}{#1}%
  \dtlmul{\@dtl@width}{\@dtl@width}{#3}%
  \dtldiv{\@dtl@gap}{\@dtl@width}{10}%
  \setlength\dtl@tmplength{\@dtl@gap sp}%
  \ifdim\dtl@tmplength<\DTLminminorticksgap
    \dtldiv{\@dtl@gap}{\@dtl@width}{4}%
    \setlength\dtl@tmplength{\@dtl@gap sp}%
    \ifdim\dtl@tmplength<\DTLminminorticksgap
      \dtldiv{\@dtl@gap}{\@dtl@width}{2}%
      \setlength\dtl@tmplength{\@dtl@gap sp}%
      \ifdim\dtl@tmplength<\DTLminminorticksgap
        \let\@dtl@gap=\@dtl@width
      \fi
    \fi
  \fi
  \dtldiv{\@dtl@gap}{\@dtl@gap}{#3}%
  \dtl@constructticklistwithgapex{#1}{#2}{\dtl@tmp}{\@dtl@gap}%
  \ifx#4\@empty
    \let#4=\dtl@tmp
  \else
    \expandafter\toks@\expandafter{#4}%
    \edef#4{#4,\dtl@tmp}%
  \fi
}
\newcommand*{\dtl@constructticklistwithgapex}[4]{%
```

```
\edef\@dtl@thistick{#1}%
\let#3=\@empty
\dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
\whiledo{\DTLisFPopenbetween{\@dtl@thistick}{#1}{#2}}{%
  \expandafter\toks@\expandafter{\@dtl@thistick}%
  \ifx#3\@empty
    \edef#3{\the\toks@}%
  \else
    \edef#3{#3,\the\toks@}%
  \fi
  \dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
}%
}
\newcommand*{\DTLaddtoplotlegend}[3]{%
\def\dtl@legendline{}%
\ifx\relax#2\relax
\else
  \toks@{#2%
  \pgfpathmoveto{\pgfpoint{-10pt}{0pt}}%
  \pgfpathlineto{\pgfpoint{10pt}{0pt}}%
  \pgfusepath{stroke}}%
  \edef\dtl@legendline{\the\toks@}%
\fi
\ifx\relax#1\relax
\else
  \toks@{#1}%
  \expandafter\@dtl@toks\expandafter{\dtl@legendline}%
  \edef\dtl@legendline{\the\@dtl@toks\the\toks@}%
\fi
\expandafter\toks@\expandafter{\dtl@legendline}%
\ifx\dtl@legend\@empty
  \xdef\dtl@legend{\noexpand\tikz\the\toks@; \noexpand& #3}%
\else
  \expandafter\@dtl@toks\expandafter{\dtl@legend}%
  \xdef\dtl@legend{\the\@dtl@toks\noexpand\\%
    \noexpand\tikz\the\toks@; \noexpand& #3}%
\fi
}
```

30.15 Rollback v2.32 (datatool-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datatool}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{xkeyval}
\RequirePackage{ifthen}
\RequirePackage{xfor}
\RequirePackage{substr}
\RequirePackage{etoolbox}
\newcommand*{\@dtl@separator}{,}
\newcommand*{\DTLsetseparator}[1]{%
```

```
  \renewcommand*{\@dtl@separator}{#1}%
  \@dtl@construct@lopoffs
}
\begingroup
\catcode`\^^I12
\gdef\DTLsettabseparator{%
  \catcode`\^^I12
  \DTLsetseparator{^^I}%
}
\gdef\DTLmaketabspace{%
  \catcode`\^^I10\relax
}
\endgroup
\begingroup
\catcode`\"12\relax
\gdef\@dtl@delimiter{"}
\endgroup
\newcommand*\DTLsetdelimiter[1]{%
  \renewcommand*{\@dtl@delimiter}{#1}%
  \@dtl@construct@lopoffs
}
\edef\@dtl@construct@lopoff#1#2{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@lopoff#1##1##2\noexpand\to##3##4{%
      \noexpand\ifx#2##1\noexpand\relax
        \noexpand\ifstrempty{##1}%
        {\noexpand\@dtl@qlopoff#1{}##2\noexpand\to##3##4\relax}%
        {%
          \noexpand\dtl@ifsingle{##1}%
          {\noexpand\@dtl@qlopoff#1##1##2\noexpand\to##3##4\relax}%
          {\noexpand\@dtl@qlopoff#1{##1}##2\noexpand\to##3##4\relax}%
        }%
      \noexpand\else
        \noexpand\ifstrempty{##1}%
        {\noexpand\@dtl@lop@ff#1{}##2\noexpand\to##3##4\relax}%
        {%
          \noexpand\dtl@ifsingle{##1}%
          {\noexpand\@dtl@lop@ff#1##1##2\noexpand\to##3##4\relax}%
          {\noexpand\@dtl@lop@ff#1{##1}##2\noexpand\to##3##4\relax}%
        }%
      \noexpand\fi
  }%
}
\edef\@dtl@construct@qlopoff#1#2{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@qlopoff#1#2##1#2#1##2\noexpand\to##3##4{%
      \noexpand\def##4{##1}%
      \noexpand\DTLsubstituteall{##4}{#2#2}{#2}%
      \noexpand\edef\noexpand\@dtl@dosubs{%
        \noexpand\noexpand\noexpand\DTLsubstituteall{\noexpand\noexpand##4}%
```

```
        {\noexpand\expandafter\noexpand\noexpand\noexpand\csname#2\noexpand\endcsname#2}%
        {\noexpand\expandafter\noexpand\noexpand\noexpand\csname#2\noexpand\endcsname}%
      }%
      \noexpand\@dtl@dosubs
      \noexpand\def##3{#1##2}%
  }%
}
\edef\@dtl@construct@lop@ff#1{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@lop@ff#1##1#1#1##2\noexpand\to##3##4{%
      \noexpand\def##4{##1}%
      \noexpand\def##3{#1##2}%
  }%
}
\newcommand{\@dtl@construct@lopoffs}{%
  \edef\@dtl@chars{{\@dtl@separator}{\@dtl@delimiter}}%
  \expandafter\@dtl@construct@lopoff\@dtl@chars
  \expandafter\@dtl@construct@qlopoff\@dtl@chars
  \expandafter\@dtl@construct@lop@ff\expandafter{\@dtl@separator}%
}
\define@key{datatool.sty}{separator}{%
  \DTLsetseparator{#1}%
}
\define@key{datatool.sty}{delimiter}{%
  \DTLsetdelimiter{#1}%
}
\define@boolkey{datatool.sty}[dtl]{verbose}[true]{}
\define@choicekey{datatool.sty}{math}[\val\nr]{fp,pgfmath}{%
  \renewcommand*\@dtl@mathprocessor{#1}%
}
\providecommand*{\@dtl@mathprocessor}{fp}
\newcommand*{\@dtl@set@options}{}
\define@choicekey{datatool.sty}{utf8}{true,false}[true]{%
  \renewcommand*{\@dtl@set@options}{\setbool{@dtl@utf8}{#1}}%
}
\ProcessOptionsX
\@dtl@construct@lopoffs
\RequirePackage{datatool-base}[=v2.32]
\@dtl@set@options
\DeclareRobustCommand{\DTLpar}{\par}
\newcommand*{\DTLnewdb}[1]{%
  \DTLifdbexists{#1}%
  {%
     \PackageError{datatool}{Database `#1' already exists}{}%
  }%
  {%
    \dtl@message{Creating database `#1'}%
    \expandafter\newtoks\csname dtldb@#1\endcsname
    \expandafter\newtoks\csname dtlkeys@#1\endcsname{}%
    \expandafter\newcount\csname dtlrows@#1\endcsname
```

```
    \expandafter\newcount\csname dtlcols@#1\endcsname
  }%
}
\newcommand*{\DTLcleardb}[1]{%
  \DTLifdbexists{#1}%
  {%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \expandafter\let\csname dtl@ci@#1@\@dtl@key\endcsname\undefined
    }%
    \csname dtldb@#1\endcsname{}%
    \csname dtlkeys@#1\endcsname{}%
    \csname dtlrows@#1\endcsname=0\relax
    \csname dtlcols@#1\endcsname=0\relax
  }%
  {%
    \PackageError{Can't clear database `#1':
       database doesn't exist}{}{}%
  }%
}
\newcommand*{\DTLdeletedb}[1]{%
  \DTLifdbexists{#1}%
  {%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \expandafter\let\csname dtl@ci@#1@\@dtl@key\endcsname\undefined
    }%
    \expandafter\let\csname dtldb@#1\endcsname\undefined
    \expandafter\let\csname dtlkeys@#1\endcsname\undefined
    \expandafter\let\csname dtlrows@#1\endcsname\undefined
    \expandafter\let\csname dtlcols@#1\endcsname\undefined
  }%
  {%
    \PackageError{Can't delete database `#1':
       database doesn't exist}{}{}%
  }%
}
\newcommand*{\DTLgnewdb}[1]{%
  \DTLifdbexists{#1}%
  {%
     \PackageError{datatool}{Database `#1' already exists}{}%
  }%
  {%
    \dtl@message{Creating database `#1'}%
    \expandafter\global\expandafter\newtoks\csname dtldb@#1\endcsname
    \expandafter\global\expandafter\newtoks\csname dtlkeys@#1\endcsname{}%
    \expandafter\global\expandafter\newcount\csname dtlrows@#1\endcsname
    \expandafter\global\expandafter\newcount\csname dtlcols@#1\endcsname
  }%
}
```

```
\newcommand*{\DTLgdeletedb}[1]{%
  \DTLifdbexists{#1}%
  {%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \expandafter\global\expandafter\let\csname dtl@ci@#1@\@dtl@key\endcsname\undefined
    }%
    \expandafter\global\expandafter\let\csname dtldb@#1\endcsname\undefined
    \expandafter\global\expandafter\let\csname dtlkeys@#1\endcsname\undefined
    \expandafter\global\expandafter\let\csname dtlrows@#1\endcsname\undefined
    \expandafter\global\expandafter\let\csname dtlcols@#1\endcsname\undefined
  }%
  {%
    \PackageError{Can't delete database `#1':
       database doesn't exist}{}{}%
  }%
}
\newcommand*{\DTLgcleardb}[1]{%
  \DTLifdbexists{#1}%
  {%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \expandafter\global\expandafter\let\csname dtl@ci@#1@\@dtl@key\endcsname\undefined
    }%
    \expandafter\global\csname dtldb@#1\endcsname{}%
    \expandafter\global\csname dtlkeys@#1\endcsname{}%
    \expandafter\global\csname dtlrows@#1\endcsname=0\relax
    \expandafter\global\csname dtlcols@#1\endcsname=0\relax
  }%
  {%
    \PackageError{Can't clear database `#1':
       database doesn't exist}{}{}%
  }%
}
\newcommand*{\DTLrowcount}[1]{%
  \expandafter\number\csname dtlrows@#1\endcsname
}
\newcommand*{\DTLcolumncount}[1]{%
  \expandafter\number\csname dtlcols@#1\endcsname
}
\newcommand{\DTLifdbempty}[3]{%
  \DTLifdbexists{#1}%
    {\@DTLifdbempty{#1}{#2}{#3}}%
    {\PackageError{Can't check if database `#1' is empty:
       database doesn't exist}{}{}}%
}
\newcommand{\@DTLifdbempty}[3]{%
  \expandafter\ifnum\csname dtlrows@#1\endcsname=0\relax
    #2%
  \else
```

```
    #3%
  \fi
}
\newcommand*{\DTLnewrow}{%
  \@ifstar\@sDTLnewrow\@DTLnewrow
}
\newcommand*{\@DTLnewrow}[1]{%
\DTLifdbexists{#1}%
  {\@sDTLnewrow{#1}}%
  {\PackageError{datatool}{Can't add new row to database `#1':
     database doesn't exist}{}}%
}
\newcommand*{\@sDTLnewrow}[1]{%
  \global\advance\csname dtlrows@#1\endcsname by 1\relax
  \@dtl@toks@gput@right@cx{dtldb@#1}{%
     \noexpand\db@row@elt@w%
       \noexpand\db@row@id@w \number\csname dtlrows@#1\endcsname
       \noexpand\db@row@id@end@%
       \noexpand\db@row@id@w \number\csname dtlrows@#1\endcsname
       \noexpand\db@row@id@end@%
     \noexpand\db@row@elt@end@%
  }%
  \dtl@message{New row added to database `#1'}%
}
\newcount\dtlcolumnnum
\newcount\dtlrownum
\newcommand*{\DTLifhaskey}{\@ifstar\@sDTLifhaskey\@DTLifhaskey}
\newcommand{\@DTLifhaskey}[4]{%
   \DTLifdbexists{#1}%
   {%
     \@sDTLifhaskey{#1}{#2}{#3}{#4}%
   }%
   {%
     \PackageError{datatool}{Database `#1' doesn't exist}{}%
   }%
}
\newcommand{\@sDTLifhaskey}[4]{%
  \@ifundefined{dtl@ci@#1@#2}%
  {%
    #4%
  }%
  {%
    #3%
  }%
}
\newcommand*{\DTLgetcolumnindex}{%
  \@ifstar\@sdtl@getcolumnindex\@dtl@getcolumnindex
}
\newcommand*{\@dtl@getcolumnindex}[3]{%
  \DTLifdbexists{#2}%
```

```
  {%
    \@sDTLifhaskey{#2}{#3}%
    {%
      \@sdtl@getcolumnindex{#1}{#2}{#3}%
    }%
    {%
      \PackageError{datatool}{Database `#2' doesn't contain
       key `#3'}{}%
    }%
  }%
  {%
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
\newcommand*{\@sdtl@getcolumnindex}[3]{%
  \expandafter\let\expandafter#1\csname dtl@ci@#2@#3\endcsname
}
\newcommand*{\dtlcolumnindex}[2]{%
  \csname dtl@ci@#1@#2\endcsname
}
\newcommand*{\DTLgetkeyforcolumn}{%
  \@ifstar\@sdtlgetkeyforcolumn\@dtlgetkeyforcolumn}
\newcommand*{\@dtlgetkeyforcolumn}[3]{%
  \DTLifdbexists{#2}%
  {%
    \ifnum#3<1\relax
      \PackageError{datatool}{Invalid column index \number#3}{%
      Column indices start at 1}%
    \else
      \expandafter\ifnum\csname dtlcols@#2\endcsname<#3\relax
        \PackageError{datatool}{Index \number#3\space out of
        range for database `#2'}{Database `#2' only has
        \expandafter\number\csname dtlcols@#2\endcsname\space
        columns}%
      \else
        \@sdtlgetkeyforcolumn{#1}{#2}{#3}%
      \fi
    \fi
  }%
  {%
    \PackageError{datatool}{Database `#2' doesn't exists}{}%
  }%
}
\newcommand*{\@sdtlgetkeyforcolumn}[3]{%
  \edef\@dtl@dogetkeyforcolumn{\noexpand\@dtl@getkeyforcolumn
    {\noexpand#1}{#2}{\number#3}}%
  \@dtl@dogetkeyforcolumn
}
\newcommand*{\@dtl@getkeyforcolumn}[3]{%
  \def\@dtl@get@keyforcolumn##1% before stuff
```

```
    \db@plist@elt@w% start of block
    \db@col@id@w #3\db@col@id@end@% index
    \db@key@id@w ##2\db@key@id@end@% key
    \db@type@id@w ##3\db@type@id@end@% data type
    \db@header@id@w ##4\db@header@id@end@% header
    \db@col@id@w #3\db@col@id@end@% index
    \db@plist@elt@end@% end of block
    ##5\q@nil{\def#1{##2}}%
  \edef\@dtl@tmp{\expandafter\the\csname dtlkeys@#2\endcsname}%
  \expandafter\@dtl@get@keyforcolumn\@dtl@tmp
    \db@plist@elt@w% start of block
    \db@col@id@w #3\db@col@id@end@ %index
    \db@key@id@w \@nil\db@key@id@end@% key
    \db@type@id@w \db@type@id@end@% data type
    \db@header@id@w \db@header@id@end@% header
    \db@col@id@w #3\db@col@id@end@% index
    \db@plist@elt@end@% end of block
    \q@nil
}
\def\DTLunsettype{}
\def\DTLstringtype{0}
\def\DTLinttype{1}
\def\DTLrealtype{2}
\def\DTLcurrencytype{3}
\newcommand*{\DTLgetdatatype}{%
  \@ifstar\@sdtlgetdatatype\@dtlgetdatatype
}
\newcommand*{\@dtlgetdatatype}[3]{%
  \DTLifdbexists{#2}%
  {%
    \@sDTLifhaskey{#2}{#3}%
    {%
      \@sdtlgetdatatype{#1}{#2}{#3}%
    }%
    {%
      \PackageError{datatool}{Key `#3' undefined in database `#2'}{}%
    }%
  }%
  {%
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
\newcommand*{\@sdtlgetdatatype}[3]{%
  \edef\@dtl@dogetdata{\noexpand\@dtl@getdatatype{\noexpand#1}%
   {\expandafter\the\csname dtlkeys@#2\endcsname}%
   {\dtlcolumnindex{#2}{#3}}}%
  \@dtl@dogetdata
}
\newcommand*{\@dtl@getdatatype}[3]{%
  \def\@dtl@get@keydata##1% stuff before
```

```
    \db@plist@elt@w% start of key block
     \db@col@id@w #3\db@col@id@end@% column index
      \db@key@id@w ##2\db@key@id@end@% key id
      \db@type@id@w ##3\db@type@id@end@% data type
      \db@header@id@w ##4\db@header@id@end@% header
     \db@col@id@w #3\db@col@id@end@% column index
    \db@plist@elt@end@% end of key block
    ##5% stuff afterwards
    \q@nil{\def#1{##3}}%
  \@dtl@get@keydata#2\q@nil
}
\newcommand*{\@dtl@getprops}[7]{%
  \def\@dtl@get@keydata##1% stuff before
    \db@plist@elt@w% start of key block
     \db@col@id@w #7\db@col@id@end@% column index
      \db@key@id@w ##2\db@key@id@end@% key id
      \db@type@id@w ##3\db@type@id@end@% data type
      \db@header@id@w ##4\db@header@id@end@% header
     \db@col@id@w #7\db@col@id@end@% column index
    \db@plist@elt@end@% end of key block
    ##5% stuff afterwards
    \q@nil{%
      \def#1{##2}% key
      \def#2{##3}% data type
      #3={##4}% header
      #4={##1}% before stuff
      #5={##5}% after stuff
    }%
  \@dtl@get@keydata#6\q@nil
}
\newtoks\@dtl@before
\newtoks\@dtl@after
\newtoks\@dtl@colhead
\newcommand*{\DTLaddcolumn}{%
   \@ifstar\@sDTLaddcolumn\@DTLaddcolumn
}
\newcommand{\@DTLaddcolumn}[2]{%
  \DTLifdbexists{#1}%
    {\@dtl@updatekeys{#1}{#2}{}}%
    {\PackageError{datatool}{Can't add new column to database `#1':
      database doesn't exist}{}}%
}
\newcommand{\s@DTLaddcolumn}[2]{%
  \@dtl@updatekeys{#1}{#2}{}%
}
\newcommand*{\@dtl@updatekeys}[3]{%
  \@sDTLifhaskey{#1}{#2}%
  {%
    \expandafter\dtlcolumnnum\expandafter
      =\dtlcolumnindex{#1}{#2}\relax
```

```
\edef\@dtl@dogetprops{\noexpand\@dtl@getprops
  {\noexpand\@dtl@key}{\noexpand\@dtl@type}%
  {\noexpand\@dtl@colhead}{\noexpand\@dtl@before}%
  {\noexpand\@dtl@after}{\the\csname dtlkeys@#1\endcsname}%
  {\number\dtlcolumnnum}}%
\@dtl@dogetprops
\ifstrempty{#3}%
{%
}%
{%
  \let\@dtl@oldtype\@dtl@type
  \@dtl@checknumerical{#3}%
  \ifdefempty{\@dtl@type}%
  {%
    \edef\@dtl@type{\number\@dtl@datatype}%
  }%
  {%
    \ifcase\@dtl@datatype % string
      \def\@dtl@type{0}%
    \or % int
    \or % real
      \ifnum\@dtl@type=1\relax
        \def\@dtl@type{2}%
      \fi
    \or % currency
      \ifnum\@dtl@type>0\relax
        \def\@dtl@type{3}%
      \fi
    \fi
  }%
  \ifx\@dtl@oldtype\@dtl@type
  \else
    \@dtl@toks@gconcat@middle@cx{dtlkeys@#1}%
    {\@dtl@before}%
    {%
      \noexpand\db@plist@elt@w% start of key block
        \noexpand\db@col@id@w \the\dtlcolumnnum
          \noexpand\db@col@id@end@% column index
        \noexpand\db@key@id@w #2\noexpand\db@key@id@end@% key id
        \noexpand\db@type@id@w \@dtl@type
          \noexpand\db@type@id@end@% data type
        \noexpand\db@header@id@w \the\@dtl@colhead
          \noexpand\db@header@id@end@% header
        \noexpand\db@col@id@w \the\dtlcolumnnum
          \noexpand\db@col@id@end@% column index
        \noexpand\db@plist@elt@end@% end of key block
    }%
    {\@dtl@after}%
  \fi
}%
```

```
  }%
  {%
    \expandafter\global\expandafter\advance
      \csname dtlcols@#1\endcsname by 1\relax
    \dtlcolumnnum=\csname dtlcols@#1\endcsname\relax
    \expandafter\xdef\csname dtl@ci@#1@#2\endcsname{%
      \number\dtlcolumnnum}%
    \ifstrempty{#2}%
    {%
      \edef\@dtl@type{}% don't know data type yet
    }%
    {%
      \@dtl@checknumerical{#3}%
      \edef\@dtl@type{\number\@dtl@datatype}%
    }%
    \@dtl@toks@gput@right@cx{dtlkeys@#1}%
    {%
      \noexpand\db@plist@elt@w
      \noexpand\db@col@id@w \the\dtlcolumnnum
        \noexpand\db@col@id@end@
      \noexpand\db@key@id@w #2\noexpand\db@key@id@end@
      \noexpand\db@type@id@w \@dtl@type
        \noexpand\db@type@id@end@
      \noexpand\db@header@id@w #2\noexpand\db@header@id@end@
      \noexpand\db@col@id@w \the\dtlcolumnnum
        \noexpand\db@col@id@end@
      \noexpand\db@plist@elt@end@
    }%
  }%
}
\newcommand*{\DTLsetheader}{\@ifstar\@sDTLsetheader\@DTLsetheader}
\newcommand*{\@DTLsetheader}[3]{%
  \DTLifdbexists{#1}%
  {%
    \@sDTLifhaskey{#1}{#2}%
    {%
      \@sDTLsetheader{#1}{#2}{#3}%
    }%
    {%
      \PackageError{datatool}{Database `#1' doesn't contain key
      `#2'}{}%
    }%
  }%
  {%
    \PackageError{datatool}{Database `#1' doesn't exist}{}%
  }%
}
\newcommand*{\@sDTLsetheader}[3]{%
  \expandafter\dtlcolumnnum\expandafter
    =\dtlcolumnindex{#1}{#2}\relax
```

```
  \@dtl@setheaderforindex{#1}{\dtlcolumnnum}{#3}%
}
\newcommand*{\@dtl@setheaderforindex}[3]{%
  \edef\@dtl@dogetprops{\noexpand\@dtl@getprops
    {\noexpand\@dtl@key}{\noexpand\@dtl@type}%
    {\noexpand\@dtl@colhead}{\noexpand\@dtl@before}%
    {\noexpand\@dtl@after}{\the\csname dtlkeys@#1\endcsname}%
    {\number#2}}%
  \@dtl@dogetprops
  \@dtl@colhead={#3}%
  \edef\@dtl@colnum{\number#2}\relax
  \@dtl@toks@gconcat@middle@cx{dtlkeys@#1}%
  {\@dtl@before}%
  {%
    \noexpand\db@plist@elt@w% start of block
      \noexpand\db@col@id@w \@dtl@colnum
        \noexpand\db@col@id@end@% index
      \noexpand\db@key@id@w \@dtl@key\noexpand\db@key@id@end@% key
      \noexpand\db@type@id@w \@dtl@type
        \noexpand\db@type@id@end@% data type
      \noexpand\db@header@id@w \the\@dtl@colhead
        \noexpand\db@header@id@end@% header
      \noexpand\db@col@id@w \@dtl@colnum
        \noexpand\db@col@id@end@% index
    \noexpand\db@plist@elt@end@% end of block
  }%
  {\@dtl@after}%
}
\newcommand*{\dtlexpandnewvalue}{%
  \def\@dtl@setnewvalue##1{\protected@edef\@dtl@tmp{##1}%
  \expandafter\@dtl@toks\expandafter{\@dtl@tmp}}%
}
\newcommand*{\dtlnoexpandnewvalue}{%
  \def\@dtl@setnewvalue##1{\@dtl@toks{##1}}%
}
\dtlnoexpandnewvalue
\newcommand{\DTLnewdbentry}{%
   \@ifstar\@sDTLnewdbentry\@DTLnewdbentry
}
\newcommand{\@DTLnewdbentry}[3]{%
  \DTLifdbexists{#1}%
    {\@sDTLnewdbentry{#1}{#2}{#3}}%
    {\PackageError{datatool}{Can't add new entry to database `#1':
      database doesn't exist}{}}%
}
\newcommand*{\@sDTLnewdbentry}[3]{%
  \@dtl@updatekeys{#1}{#2}{#3}%
  \expandafter\dtlcolumnnum\expandafter
    =\dtlcolumnindex{#1}{#2}\relax
  \edef\dtl@dogetrow{\noexpand\dtlgetrow{#1}%
```

```
    {\number\csname dtlrows@#1\endcsname}}%
  \dtl@dogetrow
  \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
     {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
  }%
  \dtl@dogetentry
  \ifx\dtl@entry\dtlnovalue
    \@dtl@setnewvalue{#3}%
    \@dtl@toks@gconcat@middle@cx{dtldb@#1}%
    {\dtlbeforerow}%
    {%
      \noexpand\db@row@elt@w%
      \noexpand\db@row@id@w \number\csname dtlrows@#1\endcsname
        \noexpand\db@row@id@end@%
      \the\dtlcurrentrow
      \noexpand\db@col@id@w \number\dtlcolumnnum
        \noexpand\db@col@id@end@%
          \noexpand\db@col@elt@w \the\@dtl@toks
            \noexpand\db@col@elt@end@%
      \noexpand\db@col@id@w \number\dtlcolumnnum
        \noexpand\db@col@id@end@%
      \noexpand\db@row@id@w \number\csname dtlrows@#1\endcsname
        \noexpand\db@row@id@end@%
      \noexpand\db@row@elt@end@%
    }%
    {\dtlafterrow}%
    \dtl@message{Added #2\space -> #3\space to database `#1'}%
  \else
    \PackageError{datatool}{Can't add entry with ID `#2' to
      current row of database `#1'}{There is already an entry with
      this ID in the current row}%
  \fi
}
\newcommand{\DTLifdbexists}[3]{%
  \@ifundefined{dtldb@#1}{#3}{#2}}
\newcommand*{\DTLassign}[3]{%
  \DTLifdbexists{#1}
  {%
    {%
      \dtlgetrow{#1}{#2}%
      \@dtl@assign{#3}{#1}%
    }%
  }%
  {%
    \PackageError{datatool}{Database `#1' doesn't exist}{}%
  }%
}
\newcommand*{\DTLassignfirstmatch}[4]{%
  \dtl@assignfirstmatch{#3}{#1}{#2}{#4}%
}
```

```
\newcommand*{\xDTLassignfirstmatch}[4]{%
  \protected@edef\@dtl@asg@value{\expandonce{#3}}%
  \expandafter\dtl@assignfirstmatch\expandafter
   {\@dtl@asg@value}{#1}{#2}{#4}%
}
\newcommand*{\dtl@assignfirstmatch}[4]{%
  \DTLifdbexists{#2}%
  {%
    {%
      \dtlgetrowindex{\dtl@asg@rowidx}{#2}{\dtlcolumnindex{#2}{#3}}{#1}%
      \ifx\dtl@asg@rowidx\dtlnovalue
        \PackageError{datatool}{No match found for
          \string\DTLassignfirstmatch{#2}{#3}{#1}{#4}}{}%
      \else
        \dtlgetrow{#2}{\dtl@asg@rowidx}%
        \@dtl@assign{#4}{#2}%
      \fi
    }%
  }%
  {%
    \PackageError{datatool}{Data base `#2' doesn't exist}{}%
  }%
}
\newcommand*{\@dtl@assign}[2]{%
  \ifstrempty{#1}{}%
  {%
    \@dtl@assigncmd#1,\@nil\@@{#2}%
  }%
}
\def\@dtl@assigncmd#1#2=#3,#4\@@#5{%
  \edef\@dtl@dbname{#5}%
  \@sDTLifhaskey{#5}{#3}%
  {%
    \edef\@dtl@dogetentry{%
      \noexpand\dtlgetentryfromcurrentrow
        {\noexpand#1}{\dtlcolumnindex{#5}{#3}}}%
    \@dtl@dogetentry
    \ifdefequal{#1}{\dtlnovalue}%
    {%
      \@@dtl@setnull{#1}{#3}%
    }%
    {}%
    \global\let#1=#1\relax
  }%
  {%
    \PackageError{datatool}{Can't assign \string#1\space: there
     is no key `#3' in data base `#5'}{}%
    \global\let#1\DTLstringnull
  }%
  \def\dtl@tmp{#4}%
```

```
  \ifx\@nnil\dtl@tmp
    \let\@dtl@next\@dtl@assigncmdnoop
  \else
    \let\@dtl@next\@dtl@assigncmd
  \fi
  \@dtl@next#4\@@{#5}%
}
\def\@dtl@assigncmdnoop#1\@@#2{}
\newcommand*{\@dtl@setnull}[2]{%
  \@sDTLifhaskey{\@dtl@dbname}{#2}%
  {%
    \@@dtl@setnull{#1}{#2}%
  }%
  {%
    \global\let#1=\DTLstringnull
  }%
}
\newcommand*{\@@dtl@setnull}[2]{%
  \@sdtlgetdatatype{\@dtl@type}{\@dtl@dbname}{#2}%
  \ifnum0\@dtl@type=0\relax
    \global\let#1=\DTLstringnull
  \else
    \global\let#1=\DTLnumbernull
  \fi
}
\newcommand*{\DTLstringnull}{\@dtlstringnull}
\newcommand*{\@dtlstringnull}{NULL}
\newcommand*{\DTLnumbernull}{\@dtlnumbernull}
\newcommand*{\@dtlnumbernull}{0}
\newcommand*{\DTLifnull}[3]{%
  \ifx#1\dtlnovalue
    #2%
  \else
    \ifx#1\DTLstringnull
      #2%
    \else
      \ifx#1\DTLnumbernull
        #2%
      \else
        #3%
      \fi
    \fi
  \fi
}
\newcommand*{\DTLifnullorempty}[3]{%
  \ifdefempty{#1}{#2}{\DTLifnull{#1}{#2}{#3}}%
}
\def\@dtlnovalue{Undefined Value}
\def\dtlnovalue{\@dtlnovalue}
\newcommand*{\DTLgetkeydata}{%
```

```
  \@ifstar\@sdtlgetkeydata\@dtlgetkeydata
}
\newcommand*{\@dtlgetkeydata}[5]{%
  \DTLifdbexists{#2}%
  {%
    \@sDTLifhaskey{#2}{#1}%
    {%
      \@sdtlgetkeydata{#1}{#2}{#3}{#4}{#5}%
    }%
    {%
      \PackageError{datatool}{Key `#1' not defined in database
        `#2'}{}%
    }%
  }%
  {%
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
\newcommand*{\@sdtlgetkeydata}[5]{%
  \@sdtl@getcolumnindex{#3}{#2}{#1}%
  \edef\@dtl@dogetkeydata{\noexpand\@dtl@getprops
    {\noexpand\@dtl@key}{\noexpand#4}{\noexpand\@dtl@colhead}%
    {\noexpand\@dtl@before}{\noexpand\@dtl@after}%
    {\expandafter\the\csname dtlkeys@#2\endcsname}%
    {#3}}%
  \@dtl@dogetkeydata
  \edef#5{\the\@dtl@toks}%
}
\newcommand{\dtl@gathervalues}[3][key]{%
  \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#2}\do
  {%
    \dtlgetentryfromrow{\@dtl@tmp}{\@dtl@col}{#3}%
    \ifx\@dtl@tmp\dtlnovalue
      \@dtl@setnull{\@dtl@tmp}{\@dtl@key}%
    \fi
    \expandafter\let\csname @dtl@#1@\@dtl@key\endcsname\@dtl@tmp
  }%
}
\newcommand{\dtl@g@gathervalues}[3][key]{%
  \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#2}\do
  {%
    \dtlgetentryfromrow{\@dtl@tmp}{\@dtl@col}{#3}%
    \ifx\@dtl@tmp\dtlnovalue
      \@dtl@setnull{\@dtl@tmp}{\@dtl@key}%
    \fi
    \expandafter\global
      \expandafter\let\csname @dtl@#1@\@dtl@key\endcsname\@dtl@tmp
  }%
}
\newtoks\dtlcurrentrow
```

```
\newtoks\dtlbeforerow
\newtoks\dtlafterrow

\newcommand*{\dtlgetrow}[2]{%
  \dtlrownum=#2\relax
  \edef\dtldbname{#1}%
  \expandafter\toks@\expandafter=\csname dtldb@#1\endcsname
  \edef\@dtl@dogetrow{\noexpand\@dtlgetrow{\the\toks@}{\number#2}}%
  \@dtl@dogetrow
}
\newcommand{\edtlgetrowforvalue}[3]{%
  \protected@edef\@dtl@dogetrowforvalue{%
    \noexpand\dtlgetrowforvalue{#1}{#2}{#3}}%
  \@dtl@dogetrowforvalue
}
\newcommand{\DTLfetch}[4]{%
  \edtlgetrowforvalue{#1}{\dtlcolumnindex{#1}{#2}}{#3}%
  \dtlgetentryfromcurrentrow{\dtlcurrentvalue}{\dtlcolumnindex{#1}{#4}}%
  \dtlcurrentvalue
}
\newcommand*{\dtlgetrowforvalue}[3]{%
  \dtlgetrowindex{\dtl@rowidx}{#1}{#2}{#3}%
  \ifx\dtl@rowidx\dtlnovalue
    \PackageError{datatool}{No row found in database `#1' for
     column `\number#2' matching `#3'}{}%
  \else
    \dtlrownum=\dtl@rowidx\relax
    \edef\dtldbname{#1}%
    \expandafter\toks@\expandafter=\csname dtldb@#1\endcsname
    \edef\@dtl@dogetrow{\noexpand\@dtlgetrow{\the\toks@}{\dtl@rowidx}}%
    \@dtl@dogetrow
  \fi
}
\newcommand*{\@dtlgetrow}[2]{%
  \def\@dtl@getrow##1% before stuff
    \db@row@elt@w% start of the row
      \db@row@id@w #2\db@row@id@end@% row id
        ##2%
      \db@row@id@w #2\db@row@id@end@% row id
    \db@row@elt@end@% end of the row
        ##3% after stuff
    \q@nil{\dtlbeforerow={##1}\dtlcurrentrow={##2}\dtlafterrow={##3}}%
  \@dtl@getrow#1\q@nil
}
\newcommand*{\dtlrecombine}{%
   \@dtl@toks@gconcat@middle@cx{dtldb@\dtldbname}%
   {\dtlbeforerow}%
   {%
      \noexpand\db@row@elt@w
        \noexpand\db@row@id@w
```

```
          \number\dtlrownum
        \noexpand\db@row@id@end@
         \the\dtlcurrentrow
        \noexpand\db@row@id@w
          \number\dtlrownum
        \noexpand\db@row@id@end@
      \noexpand\db@row@elt@end@
   }%
   {\dtlafterrow}%
}
\newcommand{\dtlrecombineomitcurrent}{%
  \dtl@decrementrows{\dtlafterrow}{\dtlrownum}
  \csname dtldb@\dtldbname\endcsname=\dtlbeforerow
  \@dtl@toks@gput@right@cx{dtldb@\dtldbname}{\the\dtlafterrow}%
  \dtl@message{Removed row \number\dtlrownum\space in database
    `\dtldbname'}%
}
\newcommand*{\dtlsplitrow}[4]{%
  \def\@dtlsplitrow##1%before stuff
    \db@col@id@w #2\db@col@id@end@% column id
      ##2% unwanted stuff
    \db@col@id@w #2\db@col@id@end@% column id
    ##3% after stuff
    \q@nil{\def#3{##1}\def#4{##3}}%
   \@dtlsplitrow#1\q@nil
}
\newcommand*{\dtlreplaceentryincurrentrow}[2]{%
  \edef\@dtl@do@splitrow{\noexpand\dtlsplitrow
   {\the\dtlcurrentrow}%
   {\number#2}%
   {\noexpand\@dtl@before@cs}%
   {\noexpand\@dtl@after@cs}}%
  \@dtl@do@splitrow
  \toks@{#1}%
  \edef\@dtl@stuff{%
    \expandonce\@dtl@before@cs
      \noexpand\db@col@id@w \number#2\noexpand
        \noexpand\db@col@id@end@% column id
      \noexpand\db@col@elt@w
        \the\toks@
      \noexpand\db@col@elt@end@
      \noexpand\db@col@id@w \number#2\noexpand
        \noexpand\db@col@id@end@% column id
    \expandonce\@dtl@after@cs
  }%
  \expandafter\dtlcurrentrow\expandafter{\@dtl@stuff}%
  \@sdtlgetkeyforcolumn{\@dtl@key}{\dtldbname}{#2}%
  \@dtl@updatekeys{\dtldbname}{\@dtl@key}{#1}%
  \dtl@message{Updated \@dtl@key\space -> #1\space in database
    `\dtldbname'}%
```

```
}
\newcommand*{\dtlremoveentryincurrentrow}[1]{%
  \edef\@dtl@do@splitrow{\noexpand\dtlsplitrow
   {\the\dtlcurrentrow}%
   {\number#1}%
   {\noexpand\@dtl@before@cs}%
   {\noexpand\@dtl@after@cs}}%
  \@dtl@do@splitrow
  \edef\@dtl@stuff{%
    \expandonce\@dtl@before@cs
    \expandonce\@dtl@after@cs
  }%
  \expandafter\dtlcurrentrow\expandafter{\@dtl@stuff}%
  \dtl@message{Removed entry from column \number#1\space\space in database
    `\dtldbname'}%
}
\newcommand*{\dtlswapentriesincurrentrow}[2]{%
  \dtlgetentryfromcurrentrow{\@dtl@entryI}{#1}%
  \dtlgetentryfromcurrentrow{\@dtl@entryII}{#2}%
  \expandafter\dtlreplaceentryincurrentrow\expandafter
   {\@dtl@entryII}{#1}%
  \expandafter\dtlreplaceentryincurrentrow\expandafter
   {\@dtl@entryI}{#2}%
}
\newcommand*{\dtlgetentryfromcurrentrow}[2]{%
  \dtlgetentryfromrow{#1}{#2}{\dtlcurrentrow}%
}
\newcommand*{\dtlgetentryfromrow}[3]{%
  \edef\@dtl@do@getentry{\noexpand\dtl@getentryfromrow
    {\noexpand#1}{\number#2}{\the#3}}%
  \@dtl@do@getentry
}
\newcommand*{\dtl@getentryfromrow}[3]{%
  \def\dtl@dogetentry##1% before stuff
    \db@col@id@w #2\db@col@id@end@% Column id
      \db@col@elt@w ##2\db@col@elt@end@% Value
    \db@col@id@w #2\db@col@id@end@% Column id
    ##3% Remaining stuff
    \q@nil{\def#1{##2}}%
  \dtl@dogetentry#3%
    \db@col@id@w #2\db@col@id@end@%
      \db@col@elt@w \@dtlnovalue\db@col@elt@end@%
    \db@col@id@w #2\db@col@id@end@%
    \q@nil
}
\newcommand*{\dtlappendentrytocurrentrow}[2]{%
      \@dtl@updatekeys{\dtldbname}{#1}{#2}%
      \expandafter\dtlcolumnnum\expandafter
        =\dtlcolumnindex{\dtldbname}{#1}\relax
      \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
```

```
        {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
      }%
      \dtl@dogetentry
      \ifx\dtl@entry\dtlnovalue
        \protected@edef\@dtl@tmp{#2}%
        \expandafter\@dtl@toks\expandafter{\@dtl@tmp}%
        \@dtl@toks@gput@right@cx{dtlcurrentrow}%
        {%
          \noexpand\db@col@id@w
            \number\dtlcolumnnum
          \noexpand\db@col@id@end@
          \noexpand\db@col@elt@w
            \the\@dtl@toks
          \noexpand\db@col@elt@end@
          \noexpand\db@col@id@w
             \number\dtlcolumnnum
          \noexpand\db@col@id@end@
        }%
         \dtl@message{Appended #1\space -> #2\space to database
           `\dtldbname'}%
      \else
        \PackageError{datatool}{Can't append entry to row:
          there is already an entry for key `#1' in this row}{}%
      \fi
}
\newcommand*{\dtlupdateentryincurrentrow}[2]{%
      \@dtl@updatekeys{\dtldbname}{#1}{#2}%
      \expandafter\dtlcolumnnum\expandafter
        =\dtlcolumnindex{\dtldbname}{#1}\relax
      \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
        {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
      }%
      \dtl@dogetentry
      \ifx\dtl@entry\dtlnovalue
        \protected@edef\@dtl@tmp{#2}%
        \expandafter\@dtl@toks\expandafter{\@dtl@tmp}%
        \@dtl@toks@gput@right@cx{dtlcurrentrow}%
        {%
          \noexpand\db@col@id@w
            \number\dtlcolumnnum
          \noexpand\db@col@id@end@
          \noexpand\db@col@elt@w
            \the\@dtl@toks
          \noexpand\db@col@elt@end@
          \noexpand\db@col@id@w
             \number\dtlcolumnnum
          \noexpand\db@col@id@end@
        }%
         \dtl@message{Appended #1\space -> #2\space to database
           `\dtldbname'}%
```

```
      \else
        \toks@{#2}%
        \edef\do@dtlreplaceincurrentrow{%
           \noexpand\dtlreplaceentryincurrentrow{\the\toks@}{\number\dtlcolumnnum}%
        }%
        \do@dtlreplaceincurrentrow
      \fi
}
\newcommand*{\DTLgetvalue}[4]{%
  \edef\dtl@dogetvalue{\noexpand\dtl@getvalue{\noexpand#1}{#2}%
    {\number#3}{\number#4}}%
  \dtl@dogetvalue
}
\newcommand*{\dtl@getvalue}[4]{%
  \def\@dtl@getvalue ##1% stuff before row <r>
     \db@row@id@w #3\db@row@id@end@% row <r> id
        ##2% stuff in row <r> before column <c>
     \db@col@id@w #4\db@col@id@end@% column <c> id
       \db@col@elt@w ##3\db@col@elt@end@% value
     ##4% stuff after value
     \q@nil{\def#1{##3}}%
  \toks@=\csname dtldb@#2\endcsname
  \expandafter\@dtl@getvalue\the\toks@% contents of data base
     \db@row@id@w #3\db@row@id@end@%
       \db@col@id@w #4\db@col@id@end@%
         \db@col@elt@w \@dtlnovalue\db@col@elt@end@% undefined value
     \q@nil
  \ifx#1\dtlnovalue
    \PackageError{datatool}{There is no element at (row=#3,\space
      column=#4) in database `#2'}{}%
  \fi
}
\newcommand*{\DTLgetlocation}[4]{%
  \def\@dtl@getlocation##1% stuff before value
    \db@col@elt@w #4\db@col@elt@end@% value
    \db@col@id@w ##2\db@col@id@end@% column id
    ##3% stuff after this column
    \db@row@id@w ##4\db@row@id@end@% row id
    ##5% stuff after row
    \q@nil{\def#1{##4}\def#2{##2}}%
  \toks@=\csname dtldb@#3\endcsname
  \expandafter\@dtl@getlocation\the\toks@% contents of data base
    \db@col@elt@w #4\db@col@elt@end@% value
    \db@col@id@w \@dtlnovalue\db@col@id@end@% undefined column id
    \db@row@id@w \@dtlnovalue\db@row@id@end@% undefined row id
    \q@nil
  \ifx#1\dtlnovalue
    \PackageError{datatool}{There is no element `#4' in database `#3'}{}%
  \fi
}
```

```
\newcommand*{\DTLgetrowindex}[4]{%
  \toks@{#4}%
  \edef\dtl@dogetrowindex{\noexpand\@dtlgetrowindex{\noexpand#1}{#2}{\number#3}{\the\tok
  \dtl@dogetrowindex
  \ifx#1\dtlnovalue
    \PackageError{datatool}{There is no element `#4' for column
      \number#3\space in database `#2'}{}%
  \fi
}
\newcommand*{\dtlgetrowindex}[4]{%
  \toks@{#4}%
  \edef\dtl@dogetrowindex{\noexpand\@dtlgetrowindex{\noexpand#1}{#2}{\number#3}{\the\tok
  \dtl@dogetrowindex
}
\newcommand*{\xdtlgetrowindex}[4]{%
  \protected@edef\dtl@dogetrowindex{\noexpand\@dtlgetrowindex{\noexpand#1}{#2}{\number#3
  \dtl@dogetrowindex
}
\newcommand*{\@dtlgetrowindex}[4]{%
  \def\@dtl@getrowindex##1% stuff before value
    \db@col@elt@w #4\db@col@elt@end@% value
    \db@col@id@w #3\db@col@id@end@% column id
    ##2% stuff after this column
    \db@row@id@w ##3\db@row@id@end@% row id
    ##4% stuff after row
    \q@nil{\def#1{##3}}%
  \toks@=\csname dtldb@#2\endcsname
  \expandafter\@dtl@getrowindex\the\toks@% contents of data base
    \db@col@elt@w #4\db@col@elt@end@% value
    \db@col@id@w #3\db@col@id@end@% column id
    \db@row@id@w \@dtlnovalue\db@row@id@end@% undefined row id
    \q@nil
}

\long\def\@dtlforeachrow(#1,#2)\in#3\do#4{%
  \edef\dtl@tmp{\expandafter\the\csname dtldb@#3\endcsname}%
  \expandafter\@dtl@foreachrow\dtl@tmp
    \db@row@elt@w%
    \db@row@id@w \@nil\db@row@id@end@%
    \db@row@id@w \@nil\db@row@id@end@%
    \db@row@elt@end@%
    \@@{#1}{#2}{#4}\q@nil
}
\long\def\@dtl@foreachrow\db@row@elt@w%
\db@row@id@w #1\db@row@id@end@%
#2\db@row@id@w #3\db@row@id@end@%
\db@row@elt@end@#4\@@#5#6#7\q@nil{%
  \gdef#5{#1}%
  \gdef\@dtl@loopbody{#7}%
  \global\advance\@dtl@foreach@level by 1\relax
```

```
  \ifx#5\@nnil
    \expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachnoop
  \else
    \gdef#6{#2}%
    \expandafter\let
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
      \dtlbreak
    \gdef\dtlbreak{\expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachnoop}%
    \expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachrow
    \@dtl@loopbody
    \expandafter\let\expandafter\dtlbreak
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
  \fi
  \expandafter\let\expandafter\@dtl@foreachnext
    \csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
  \global\advance\@dtl@foreach@level by -1\relax
  \@dtl@foreachnext#4\@@{#5}{#6}{#7}\q@nil
}
\long\def\@dtl@foreachnoop#1\@@#2\q@nil{}
\long\def\dtlforeachkey(#1,#2,#3,#4)\in#5\do#6{%
  \gdef\@dtl@loopbody{#6}%
  \edef\@dtl@keys{\expandafter\the\csname dtlkeys@#5\endcsname}%
  \expandafter\@dtl@foreachkey\@dtl@keys
    \db@plist@elt@w%
    \db@col@id@w -1\db@col@id@end@%
    \db@key@id@w \db@key@id@end@%
    \db@type@id@w \db@type@id@end@%
    \db@header@id@w \db@header@id@end@%
    \db@col@id@w -1\db@col@id@end@%
    \db@plist@elt@end@%
    \@@{\@dtl@updatefkcs{#1}{#2}{#3}{#4}}\q@nil
}
\newcommand*{\@dtl@updatefkcs}[8]{%
  \gdef#1{#5}%
  \gdef#2{#6}%
  \gdef#3{#7}%
  \gdef#4{#8}%
}
\long\def\@dtl@foreachkey\db@plist@elt@w%
\db@col@id@w #1\db@col@id@end@%
\db@key@id@w #2\db@key@id@end@%
\db@type@id@w #3\db@type@id@end@%
\db@header@id@w #4\db@header@id@end@%
\db@col@id@w #5\db@col@id@end@%
```

```
\db@plist@elt@end@#6\@@#7\q@nil{%
  \ifnum#1=-1\relax
    \let\@dtl@foreachnext\@dtl@foreachnoop
  \else
    #7{#2}{#1}{#3}{#4}%
    \global\advance\@dtl@foreach@level by 1\relax
    \expandafter\let
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
      \dtlbreak
    \gdef\dtlbreak{\expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachnoop}%
    \expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachkey
    \@dtl@loopbody
    \expandafter\let\expandafter\@dtl@foreachnext
      \csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
    \expandafter\let\expandafter\dtlbreak
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
    \global\advance\@dtl@foreach@level by -1\relax
  \fi
  \@dtl@foreachnext#6\@@{#7}\q@nil
}
\newcommand*{\dtlforcolumn}{\@ifstar\@sdtlforcolumn\@dtlforcolumn}
\newcommand{\@dtlforcolumn}[4]{%
  \DTLifdbexists{#2}%
  {%
    \@DTLifhaskey{#2}{#3}%
    {%
      \@sdtlforcolumn{#1}{#2}{#3}{#4}%
    }%
    {%
      \PackageError{datatool}{Database `#2' doesn't contain
        key `#3'}{}%
    }%
  }%
  {%
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
\newcommand{\@sdtlforcolumn}[4]{%
    \toks@{#4}%
    \edef\@dtl@doforcol{\noexpand\dtl@forcolumn{\noexpand#1}%
      {\expandafter\the\csname dtldb@#2\endcsname}%
      {\dtlcolumnindex{#2}{#3}}{\the\toks@}%
    }%
    \@dtl@doforcol%
}
\newcommand*{\dtlforcolumnidx}{%
```

```
  \@ifstar\@sdtlforcolumnidx\@dtlforcolumnidx
}
\newcommand{\@dtlforcolumnidx}[4]{%
  \DTLifdbexists{#2}%
  {%
    \expandafter\ifnum\csname dtlcols@#2\endcsname<#3\relax
      \PackageError{datatool}{Column index \number#3\space out of
        bounds for database `#2'}{Database `#2' only has
        \expandafter\number\csname dtlcols@#2\endcsname\space
        columns}%
    \else
      \ifnum#3<1\relax
       \PackageError{datatool}{Column index \number#3\space out of
        bounds for database `#2'}{Indices start from 1}%
      \else
        \@sdtlforcolumnidx{#1}{#2}{#3}{#4}%
      \fi
    \fi
  }%
  {%
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
\newcommand{\@sdtlforcolumnidx}[4]{%
    \toks@{#4}%
    \edef\@dtl@doforcol{\noexpand\dtl@forcolumn{\noexpand#1}%
      {\expandafter\the\csname dtldb@#2\endcsname}%
      {\number#3}{\the\toks@}%
    }%
    \@dtl@doforcol
}
\newcommand{\dtl@forcolumn}[4]{%
  \let\@dtl@oldbreak\dtlbreak
  \def\dtlbreak{\let\@dtl@forcolnext=\@dtl@forcolnoop}%
  \def\@dtl@forcolumn##1% before stuff
    \db@col@id@w #3\db@col@id@end@% column index
      \db@col@elt@w ##2\db@col@elt@end@% entry
    \db@col@id@w #3\db@col@id@end@% column index
    ##3% after stuff
    \q@nil{%
      \def#1{##2}% assign value to <cs>
      \ifx#1\@nnil
        \let\@dtl@forcolnext=\@dtl@forcolnoop
      \else
        #4%
        \let\@dtl@forcolnext=\@dtl@forcolumn
      \fi
      \@dtl@forcolnext##3\q@nil
    }%
  \@dtl@forcolumn#2%
```

```
   \db@col@id@w #3\db@col@id@end@%
     \db@col@elt@w \@nil\db@col@elt@end@%
   \db@col@id@w #3\db@col@id@end@\q@nil
  \let\dtlbreak\@dtl@oldbreak
}
\def\@dtl@forcolnoop#1\q@nil{}

\newcount\dtlforeachlevel
\newcounter{DTLrowi}
\newcounter{DTLrowii}
\newcounter{DTLrowiii}
\newcounter{DTLrow}
\def\theHDTLrow{\arabic{DTLrow}}
\def\theHDTLrowi{\theHDTLrow.\arabic{DTLrowi}}
\def\theHDTLrowii{\theHDTLrowi.\arabic{DTLrowii}}
\def\theHDTLrowiii{\theHDTLrowii.\arabic{DTLrowiii}}
\newcount\dtl@rowi
\newcount\dtl@rowii
\newcount\dtl@rowiii
\newtoks\@dtl@curi
\newtoks\@dtl@previ
\newtoks\@dtl@nexti
\newtoks\@dtl@curii
\newtoks\@dtl@previi
\newtoks\@dtl@nextii
\newtoks\@dtl@curiii
\newtoks\@dtl@previii
\newtoks\@dtl@nextiii
\newcommand*{\DTLsavelastrowcount}[1]{%
\ifnum\dtlforeachlevel>2\relax
  \def#1{0}%
\else
  \ifnum\dtlforeachlevel<0\relax
    \def#1{0}%
  \else
    \@dtl@tmpcount=\dtlforeachlevel
    \advance\@dtl@tmpcount by 1\relax
    \edef#1{\expandafter\number
      \csname c@DTLrow\romannumeral\@dtl@tmpcount\endcsname}%
  \fi
\fi}
\newenvironment{DTLenvforeach}[3][\boolean{true}]%
{%
  \def\@dtlenvforeach@args{[#1]{#2}{#3}}%
  \long@collect@body\@do@dtlenvforeach
}%
{}
\newcommand{\@do@dtlenvforeach}[1]{%
  \expandafter\@DTLforeach\@dtlenvforeach@args{#1}%
}
```

```
\newenvironment{DTLenvforeach*}[3][\boolean{true}]%
{%
  \def\s@dtlenvforeach@args{[#1]{#2}{#3}}%
  \long@collect@body\@do@sdtlenvforeach
}%
{}
\newcommand{\@do@sdtlenvforeach}[1]{%
  \expandafter\@sDTLforeach\s@dtlenvforeach@args{#1}%
}
\newcommand*{\DTLforeach}{\@ifstar\@sDTLforeach\@DTLforeach}
\newcommand{\@DTLforeach}[4][\boolean{true}]{%
  \DTLifdbexists{#2}%
  {%
    \refstepcounter{DTLrow}%
    \global\c@DTLrow=\c@DTLrow\relax
    \xdef\@dtl@dbname{#2}%
    \global\advance\dtlforeachlevel by 1\relax
    \ifnum\dtlforeachlevel>3\relax
      \PackageError{datatool}{\string\DTLforeach\space nested too
        deeply}{Only 3 levels are allowed}%
    \else
       \@DTLifdbempty{#2}%
       {}%
       {%
        \expandafter\global
          \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
            = 0\relax
        \expandafter\global\expandafter\let%
          \csname @dtl@iffirstrow\the\dtlforeachlevel\endcsname
          \DTLiffirstrow
        \gdef\DTLiffirstrow##1##2{%
          \expandafter\ifnum
           \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
           =1 %space intended
            ##1%
          \else
            ##2%
          \fi}%
        \expandafter\global\expandafter\let%
          \csname @dtl@iflastrow\the\dtlforeachlevel\endcsname
          \DTLiflastrow
        \gdef\DTLiflastrow##1##2{%
          \expandafter\ifnum
           \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
           =\csname dtlrows@#2\endcsname
            ##1%
          \else
            ##2%
          \fi}%
        \expandafter\global\expandafter\let%
```

```
  \csname @dtl@ifoddrow\the\dtlforeachlevel\endcsname
  \DTLifoddrow
\gdef\DTLifoddrow##1##2{%
  \expandafter\ifodd
   \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
    ##1%
  \else
    ##2%
  \fi}%
\expandafter\global\expandafter\let
  \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
    =\@dtl@dbname
\expandafter\global\expandafter\let
  \csname @dtl@ro@\romannumeral\dtlforeachlevel\endcsname
    = 0\relax
\dtlgforint
  \csname dtl@row\romannumeral\dtlforeachlevel\endcsname
  =1\to\csname dtlrows@#2\endcsname\step1\do
{%
  \@dtl@tmpcount=
    \csname dtl@row\romannumeral\dtlforeachlevel\endcsname
  \edef\dtl@dogetrow{\noexpand\dtlgetrow{#2}%
    {\number\@dtl@tmpcount}}%
  \dtl@dogetrow
  \expandafter\global
    \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
      = \dtlcurrentrow
  \expandafter\global
    \csname @dtl@prev\romannumeral\dtlforeachlevel\endcsname
      = \dtlbeforerow
  \expandafter\global
    \csname @dtl@next\romannumeral\dtlforeachlevel\endcsname
      = \dtlafterrow
  \ifblank{#3}{}{\@dtl@assign{#3}{#2}}%
  \ifthenelse{#1}%
  {%
    \refstepcounter{DTLrow\romannumeral\dtlforeachlevel}%
    \expandafter\edef\expandafter\DTLcurrentindex%
      \expandafter{%
         \arabic{DTLrow\romannumeral\dtlforeachlevel}}%
    #4%
    \edef\@dtl@tmp{\expandafter\the
      \csname @dtl@cur\romannumeral
        \dtlforeachlevel\endcsname}%
    \ifx\@dtl@tmp\@nnil
      \expandafter\dtl@decrementrows\expandafter
         {\csname @dtl@prev\romannumeral
            \dtlforeachlevel\endcsname
         }{\csname dtl@row\romannumeral
           \dtlforeachlevel\endcsname}%
```

```
      \expandafter\dtl@decrementrows\expandafter
         {\csname @dtl@next\romannumeral
            \dtlforeachlevel\endcsname
         }{\csname dtl@row\romannumeral
           \dtlforeachlevel\endcsname}%
      \edef\@dtl@tmp{%
        \expandafter\the
          \csname @dtl@prev\romannumeral
            \dtlforeachlevel\endcsname
        \expandafter\the
          \csname @dtl@next\romannumeral
             \dtlforeachlevel\endcsname
        }%
      \expandafter\global\expandafter
         \csname dtldb@#2\endcsname\expandafter{\@dtl@tmp}%
      \expandafter\global\expandafter
        \advance\csname dtlrows@#2\endcsname by -1\relax
      \expandafter\global\expandafter
        \advance\csname dtl@row\romannumeral
           \dtlforeachlevel\endcsname by -1\relax
    \else
      \@dtl@before=\csname @dtl@prev\romannumeral
        \dtlforeachlevel\endcsname
      \@dtl@after=\csname @dtl@next\romannumeral
        \dtlforeachlevel\endcsname
      \@dtl@toks@gconcat@middle@cx{dtldb@#2}%
      {\@dtl@before}%
      {%
        \noexpand\db@row@elt@w%
        \noexpand\db@row@id@w \expandafter\number
          \csname dtl@row\romannumeral
            \dtlforeachlevel\endcsname
        \noexpand\db@row@id@end@%
        \expandafter\the
          \csname @dtl@cur\romannumeral
            \dtlforeachlevel\endcsname
        \noexpand\db@row@id@w \expandafter\number
          \csname dtl@row\romannumeral
            \dtlforeachlevel\endcsname
        \noexpand\db@row@id@end@%
        \noexpand\db@row@elt@end@%
        }%
      {\@dtl@after}%
    \fi
  }%
  {}%
 }%
\expandafter\global\expandafter\let\expandafter\DTLiffirstrow
  \csname @dtl@iffirstrow\the\dtlforeachlevel\endcsname
\expandafter\global\expandafter\let\expandafter\DTLiflastrow
```

```
          \csname @dtl@iflastrow\the\dtlforeachlevel\endcsname
        \expandafter\global\expandafter\let\expandafter\DTLifoddrow
          \csname @dtl@ifoddrow\the\dtlforeachlevel\endcsname
      }%
    \fi
    \global\advance\dtlforeachlevel by -1\relax
  }%
  {%
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
\newcommand{\@sDTLforeach}[4][\boolean{true}]{%
  \DTLifdbexists{#2}%
  {%
    \refstepcounter{DTLrow}%
    \global\c@DTLrow=\c@DTLrow
    \xdef\@dtl@dbname{#2}%
    \global\advance\dtlforeachlevel by 1\relax
    \ifnum\dtlforeachlevel>3\relax
      \PackageError{datatool}{\string\DTLforeach\space nested too
        deeply}{Only 3 levels are allowed}%
    \else
       \@DTLifdbempty{#2}%
       {}%
       {%
        \expandafter\global
          \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
            = 0\relax
        \expandafter\global\expandafter\let%
          \csname @dtl@iffirstrow\the\dtlforeachlevel\endcsname
          \DTLiffirstrow
        \gdef\DTLiffirstrow##1##2{%
          \expandafter\ifnum
           \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
           =1 % space intended
            ##1%
          \else
            ##2%
          \fi}%
        \expandafter\global\expandafter\let%
          \csname @dtl@iflastrow\the\dtlforeachlevel\endcsname
          \DTLiflastrow
        \gdef\DTLiflastrow##1##2{%
          \expandafter\ifnum
           \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
           =\csname dtlrows@#2\endcsname
            ##1%
          \else
            ##2%
          \fi}%
```

```
      \expandafter\global\expandafter\let%
        \csname @dtl@ifoddrow\the\dtlforeachlevel\endcsname
        \DTLifoddrow
      \gdef\DTLifoddrow##1##2{%
        \expandafter\ifodd
         \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
          ##1%
        \else
          ##2%
        \fi}%
      \expandafter\gdef\csname @dtl@dbname@\romannumeral
        \dtlforeachlevel\endcsname{#2}%
      \expandafter\global\expandafter\let
        \csname @dtl@ro@\romannumeral\dtlforeachlevel\endcsname
          = 1\relax
      \@dtlforeachrow(\dtl@thisidx,\dtl@thisrow)\in{#2}\do%
      {%
        \csname dtl@row\romannumeral\dtlforeachlevel\endcsname
          = \dtl@thisidx\relax
        \expandafter\global
          \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
            = \expandafter{\dtl@thisrow}%
        \ifblank{#3}{}
        {%
          \dtlcurrentrow=\expandafter{\dtl@thisrow}%
          \@dtl@assign{#3}{#2}%
        }%
        \ifthenelse{#1}%
        {%
          \refstepcounter{DTLrow\romannumeral\dtlforeachlevel}%
          \expandafter\edef\expandafter\DTLcurrentindex%
            \expandafter{%
               \arabic{DTLrow\romannumeral\dtlforeachlevel}}%
          #4%
        }%
        {}%
       }%
      \expandafter\global\expandafter\let\expandafter\DTLiffirstrow
        \csname @dtl@iffirstrow\the\dtlforeachlevel\endcsname
      \expandafter\global\expandafter\let\expandafter\DTLiflastrow
        \csname @dtl@iflastrow\the\dtlforeachlevel\endcsname
      \expandafter\global\expandafter\let\expandafter\DTLifoddrow
        \csname @dtl@ifoddrow\the\dtlforeachlevel\endcsname
    }%
  \fi
  \global\advance\dtlforeachlevel by -1\relax
}%
{%
  \PackageError{datatool}{Database `#2' doesn't exist}{}%
}%
```

```
}
\newcommand*{\@dtlifreadonly}[2]{%
  \expandafter\ifx
    \csname @dtl@ro@\romannumeral\dtlforeachlevel\endcsname1\relax
    #1%
  \else
    #2%
  \fi
}
\newcommand*{\DTLappendtorow}[2]{%
  \ifnum\dtlforeachlevel=0\relax
    \PackageError{datatool}{\string\DTLappendrow\space can only be
      used inside \string\DTLforeach}{}%
  \else
    \expandafter\let\expandafter\@dtl@thisdb
      \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
    \@dtlifreadonly
    {%
      \PackageError{datatool}{\string\DTLappendtorow\space can't
       be used inside \DTLforeach*}{The starred version of
       \string\DTLforeach\space is read only}%
    }%
    {%
      \dtlrownum=
        \csname dtl@row\romannumeral\dtlforeachlevel\endcsname\relax
      \@dtl@updatekeys{\@dtl@thisdb}{#1}{#2}%
      \expandafter\dtlcolumnnum\expandafter
        =\dtlcolumnindex{\@dtl@thisdb}{#1}\relax
      \dtlcurrentrow =
        \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
      \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
        {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
      }%
      \dtl@dogetentry
      \ifx\dtl@entry\dtlnovalue
        \protected@edef\@dtl@tmp{#2}%
        \expandafter\@dtl@toks\expandafter{\@dtl@tmp}%
        \@dtl@toks@gput@right@cx{@dtl@cur\romannumeral\dtlforeachlevel}%
        {%
          \noexpand\db@col@id@w \number\dtlcolumnnum
            \noexpand\db@col@id@end@
          \noexpand\db@col@elt@w \the\@dtl@toks
            \noexpand\db@col@elt@end@
          \noexpand\db@col@id@w \number\dtlcolumnnum
            \noexpand\db@col@id@end@
        }%
         \dtl@message{Appended #1\space -> #2\space to database
           `\@dtl@thisdb'}%
      \else
        \PackageError{datatool}{Can't append entry to row:
```

```
          there is already an entry for key `#1' in this row}{}%
      \fi
    }%
  \fi
}
\newcommand*{\DTLremoveentryfromrow}[1]{%
  \ifnum\dtlforeachlevel=0\relax
    \PackageError{datatool}{\string\DTLremoventryfromrow\space
      can only be used inside \string\DTLforeach}{}%
  \else
    \expandafter\let\expandafter\@dtl@thisdb
      \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
    \@dtlifreadonly
    {%
      \PackageError{datatool}{\string\DTLremoveentryfromrow\space
        can't be used inside \string\DTLforeach*}{The starred
        version of \string\DTLforeach\space is read only}%
    }%
    {%
      \dtlrownum=
        \csname dtl@row\romannumeral\dtlforeachlevel\endcsname\relax
      \@DTLifhaskey{\@dtl@thisdb}{#1}%
      {%
        \@dtl@getcolumnindex{\thiscol}{\@dtl@thisdb}{#1}\relax
        \dtlcolumnnum=\thiscol\relax
        \dtlcurrentrow =
          \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
        \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
          {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
        }%
        \dtl@dogetentry
        \ifx\dtl@entry\dtlnovalue
          \PackageError{datatool}{Can't remove entry given by `#1'
             from current row in database `\@dtl@thisdb': no such
             entry}{The current row doesn't contain an entry for
             key `#1'}%
        \else
          \edef\@dtl@dosplitrow{%
            \noexpand\dtlsplitrow{\the\dtlcurrentrow}%
              {\number\dtlcolumnnum}{\noexpand\dtl@pre}%
              {\noexpand\dtl@post}%
          }%
          \@dtl@dosplitrow
          \expandafter\@dtl@toks\expandafter{\dtl@pre}%
          \expandafter\toks@\expandafter{\dtl@post}%
          \edef\@dtl@tmp{\the\@dtl@toks \the\toks@}%
          \dtlcurrentrow=\expandafter{\@dtl@tmp}%
          \expandafter\global
            \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
              = \dtlcurrentrow
```

```
          \dtl@message{Removed entry given by #1\space from current
            row of database `\@dtl@thisdb'}%
        \fi
      }%
      {%
        \PackageError{datatool}{Can't remove entry given by
          `#1' - no such key exists}{}%
      }%
    }%
  \fi
}
\newcommand*{\DTLreplaceentryforrow}[2]{%
  \ifnum\dtlforeachlevel=0\relax
    \PackageError{datatool}{\string\DTLreplaceentryforrow\space
      can only be used inside \string\DTLforeach}{}%
  \else
    \expandafter\let\expandafter\@dtl@thisdb
      \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
    \@dtlifreadonly
    {%
      \PackageError{datatool}{\string\DTLreplaceentryforrow\space
        can't be used inside \string\DTLforeach*}{The starred version
        of \string\DTLforeach\space is read only}%
    }%
    {%
      \dtlrownum=
        \csname dtl@row\romannumeral\dtlforeachlevel\endcsname\relax
      \@DTLifhaskey{\@dtl@thisdb}{#1}%
      {%
        \@dtl@getcolumnindex{\thiscol}{\@dtl@thisdb}{#1}\relax
        \dtlcolumnnum=\thiscol\relax
        \dtlcurrentrow =
          \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
        \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
          {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
        }%
        \dtl@dogetentry
        \ifx\dtl@entry\dtlnovalue
          \PackageError{datatool}{Can't replace entry given by `#1'
             from current row in database `\@dtl@thisdb': no such
             entry}{The current row doesn't contain an entry for
             key `#1'}%
        \else
          \edef\@dtl@dosplitrow{%
            \noexpand\dtlsplitrow{\the\dtlcurrentrow}%
              {\number\dtlcolumnnum}{\noexpand\dtl@pre}%
              {\noexpand\dtl@post}%
          }%
          \@dtl@dosplitrow
          \protected@edef\@dtl@tmp{#2}%
```

```
          \expandafter\@dtl@toks\expandafter{\@dtl@tmp}% new value
          \expandafter\@dtl@before\expandafter{\dtl@pre}%
          \expandafter\@dtl@after\expandafter{\dtl@post}%
          \@dtl@toks@gconcat@middle@cx
            {@dtl@cur\romannumeral\dtlforeachlevel}%
            {\@dtl@before}%
            {%
              \noexpand\db@col@id@w \number\dtlcolumnnum
                \noexpand\db@col@id@end@%
              \noexpand\db@col@elt@w \the\@dtl@toks
                \noexpand\db@col@elt@end@%
              \noexpand\db@col@id@w \number\dtlcolumnnum
                \noexpand\db@col@id@end@%
            }%
            {\@dtl@after}%
         \dtl@message{Updated #1\space -> #2\space in database
           `\@dtl@thisdb'}%
       \fi
     }%
     {%
        \PackageError{datatool}{Can't replace key `#1' - no such
          key in database `\@dtl@thisdb'}{}%
     }%
   }%
 \fi
}
\newcommand*{\DTLremovecurrentrow}{%
 \ifnum\dtlforeachlevel=0\relax
   \PackageError{datatool}{\string\DTLremovecurrentrow\space can
     only be used inside \string\DTLforeach}{}%
 \else
   \expandafter\let\expandafter\@dtl@thisdb
     \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
   \@dtlifreadonly
   {%
     \PackageError{datatool}{\string\DTLreplaceentryforrow\space
       can't be used inside \string\DTLforeach*}{The starred version
       of \string\DTLforeach\space is read only}%
   }%
   {%
     \expandafter\global
       \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
         ={\@nil}%
   }%
 \fi
}
\newcommand{\DTLaddentryforrow}[5]{%
 \DTLifdbexists{#1}%
 {%
   \def\@dtl@notdone{\PackageError{datatool}{Unable to add entry
```

```
      given by key `#4': condition not met for any row in database
      `#1'}{}%
    \DTLforeach[#3]{#1}{#2}%
    {%
      \DTLappendtorow{#4}{#5}%
      \let\@dtl@notdone\relax
      \dtlbreak
    }%
    \@dtl@notdone
  }%
  {%
    \PackageError{datatool}{Unable to add entry given by key `#4':
      database `#1' doesn't exist}{}%
  }%
}
\newcommand*{\DTLforeachkeyinrow}[2]{%
  \ifnum\dtlforeachlevel=0\relax
    \PackageError{datatool}{\string\DTLforeachkeyinrow\space can only
     be used inside \string\DTLforeach}{}%
  \else
    \expandafter\let\expandafter\@dtl@thisdb
      \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
    \dtlforeachkey(\dtlkey,\dtlcol,\dtltype,\dtlheader)\in
      \@dtl@thisdb\do{%
      \dtlcurrentrow =
        \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
      \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
        {\noexpand#1}{\dtlcol}}%
      \dtl@dogetentry
      \ifx#1\dtlnovalue
        \ifnum0\dtltype=0\relax
          \let#1=\@dtlstringnull
        \else
          \let#1=\@dtlnumbernull
        \fi
      \fi
      \global\let#1#1%
      \def\@dtl@loop@body{#2}%
      \@dtl@loop@body
    }%
  \fi
}
\newcommand{\DTLiffirstrow}[2]{%
  \PackageError{datatool}{\string\DTLiffirstrow\space can only
  be used inside \string\DTLforeach}{}%
}
\newcommand{\DTLiflastrow}[2]{%
  \PackageError{datatool}{\string\DTLiflastrow\space can only
  be used inside \string\DTLforeach}{}%
}
```

```
\newcommand{\DTLifoddrow}[2]{%
  \PackageError{datatool}{\string\DTLifoddrow\space can only
  be used inside \string\DTLforeach}{}%
}
\newcommand*{\dtlbetweencols}{}
\newcommand*{\dtlbeforecols}{}
\newcommand*{\dtlaftercols}{}
\newcommand*{\dtlstringalign}{l}
\newcommand*{\dtlintalign}{r}
\newcommand*{\dtlrealalign}{r}
\newcommand*{\dtlcurrencyalign}{r}
\newcommand*{\dtladdalign}[4]{%
  \ifnum#3=1\relax
    \protected@edef#1{\dtlbeforecols}%
  \else
    \protected@edef#1{#1\dtlbetweencols}%
  \fi
  \ifstrempty{#2}%
  {%
    \protected@edef#1{#1c}%
  }%
  {%
    \ifcase#2\relax
      \protected@edef#1{#1\dtlstringalign}%
    \or
      \protected@edef#1{#1\dtlintalign}%
    \or
      \protected@edef#1{#1\dtlrealalign}%
    \or
      \protected@edef#1{#1\dtlcurrencyalign}%
    \else
      \protected@edef#1{#1c}%
      \PackageError{datatool}{Unknown data type `#2'}{}%
    \fi
  }%
  \ifnum#3=#4\relax
    \protected@edef#1{#1\dtlaftercols}%
  \fi
}
\newcommand*{\dtlheaderformat}[1]{\null\hfil\textbf{#1}\hfil\null}
\newcommand*{\dtlstringformat}[1]{#1}
\newcommand*{\dtlintformat}[1]{#1}
\newcommand*{\dtlrealformat}[1]{#1}
\newcommand*{\dtlcurrencyformat}[1]{#1}
\newcommand*{\dtldisplaystarttab}{}
\newcommand*{\dtldisplayendtab}{}
\newcommand*{\dtldisplayafterhead}{}
\newcommand*{\dtldisplayvalign}{c}
\newcommand*{\dtldisplaystartrow}{}
\newcommand{\dtldisplaycr}{\tabularnewline}
```

```
\newcommand*{\DTLdisplaydb}[2][]{%
  \def\@dtl@doamp{\gdef\@dtl@doamp{&}}%
  \def\@dtl@resetdoamp{\gdef\@dtl@doamp{\gdef\@dtl@doamp{&}}}%
  \edef\@dtl@maxcols{\expandafter\number
    \csname dtlcols@#2\endcsname}%
  \DTLnumitemsinlist{#1}{\@dtl@tmp}%
  \dtlsub{\@dtl@maxcols}{\@dtl@maxcols}{\@dtl@tmp}%
  \dtlclip{\@dtl@maxcols}{\@dtl@maxcols}%
  \def\@dtl@tabargs{}%
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{#1}%
    {}%
    {%
       \dtladdalign\@dtl@tabargs\@dtl@type\@dtl@idx\@dtl@maxcols
    }%
  }%
  \edef\@dtl@dobegintab{\noexpand\begin{tabular}[\dtldisplayvalign]{\@dtl@tabargs}}%
  \@dtl@dobegintab
  \dtldisplaystarttab
  \@dtl@resetdoamp
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{#1}%
    {}%
    {%
      \@dtl@doamp
      \dtlheaderformat{\@dtl@head}%
    }%
  }%
  \\%
  \dtldisplayafterhead
  \@dtl@resetdoamp
  \@sDTLforeach{#2}{}{%
    \DTLiffirstrow{}{\dtldisplaycr\dtldisplaystartrow}%
    \@dtl@resetdoamp
    \DTLforeachkeyinrow{\@dtl@val}%
    {%
      \expandafter\DTLifinlist\expandafter{\dtlkey}{#1}%
      {}%
      {%
        \global\let\@dtl@val\@dtl@val
        \@dtl@doamp
        \@dtl@datatype=0\dtltype\relax
        \ifcase\@dtl@datatype
          \dtlstringformat\@dtl@val
        \or
```

```
          \dtlintformat\@dtl@val
        \or
          \dtlrealformat\@dtl@val
        \or
          \dtlcurrencyformat\@dtl@val
        \else
          \@dtl@val
        \fi
      }%
    }%
  }%
  \dtldisplayendtab
  \end{tabular}%
}
\define@key{displaylong}{caption}{\def\@dtl@cap{#1}}
\define@key{displaylong}{contcaption}{\def\@dtl@contcap{#1}}
\define@key{displaylong}{shortcaption}{\def\@dtl@shortcap{#1}}
\define@key{displaylong}{label}{\def\@dtl@label{#1}}
\define@key{displaylong}{foot}{\def\@dtl@foot{#1}}
\define@key{displaylong}{lastfoot}{\def\@dtl@lastfoot{#1}}
\define@key{displaylong}{omit}{\def\@dtl@omitlist{#1}}
\newcommand*{\@dtl@resetdostartrow}{%
  \gdef\@dtl@dostartrow{%
    \gdef\@dtl@dostartrow{\dtldisplaycr\dtldisplaystartrow}}%
}
\newcommand*{\DTLdisplaylongdb}[2][]{%
  \def\@dtl@cap{\@nil}%
  \def\@dtl@contcap{\@nil}%
  \def\@dtl@label{\@nil}%
  \def\@dtl@shortcap{\@dtl@cap}%
  \def\@dtl@foot{\@nil}%
  \def\@dtl@lastfoot{\@nil}%
  \def\@dtl@omitlist{}%
  \setkeys{displaylong}{#1}%
  \def\@dtl@doamp{\gdef\@dtl@doamp{&}}%
  \def\@dtl@resetdoamp{\gdef\@dtl@doamp{\gdef\@dtl@doamp{&}}}%
  \@dtl@resetdostartrow
  \edef\@dtl@maxcols{\expandafter\number
    \csname dtlcols@#2\endcsname}%
  \DTLnumitemsinlist{\@dtl@omitlist}{\@dtl@tmp}%
  \dtlsub{\@dtl@maxcols}{\@dtl@maxcols}{\@dtl@tmp}%
  \dtlclip{\@dtl@maxcols}{\@dtl@maxcols}%
  \def\@dtl@tabargs{}%
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{\@dtl@omitlist}%
    {}%
    {%
      \dtladdalign\@dtl@tabargs\@dtl@type\@dtl@idx\@dtl@maxcols
```

```
  }%
}%
\edef\@dtl@dobegintab{\noexpand\begin{longtable}{\@dtl@tabargs}}%
\@dtl@dobegintab
\ifx\@dtl@foot\@nnil
\else
  \@dtl@foot\endfoot
\fi
\ifx\@dtl@lastfoot\@nnil
\else
  \@dtl@lastfoot\endlastfoot
\fi
\ifx\@dtl@cap\@nnil
  \@dtl@resetdoamp
  \dtldisplaystarttab
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{\@dtl@omitlist}%
    {}%
    {%
      \@dtl@doamp{\dtlheaderformat{\@dtl@head}}%
    }%
  }%
  \@dtl@resetdoamp
  \@dtl@resetdostartrow
  \endhead\dtldisplayafterhead
\else
  \caption[\@dtl@shortcap]{\@dtl@cap}%
  \ifx\@dtl@label\@nnil
  \else
    \label{\@dtl@label}%
  \fi
  \dtldisplaycr
\dtldisplaystarttab
  \@dtl@resetdoamp
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{\@dtl@omitlist}%
    {}%
    {%
      \@dtl@doamp{\dtlheaderformat{\@dtl@head}}%
    }%
  }%
  \@dtl@resetdoamp
  \dtldisplaycr\dtldisplayafterhead
  \endfirsthead
  \ifx\@dtl@contcap\@nnil
    \caption{\@dtl@cap}%
```

```
    \else
      \caption{\@dtl@contcap}%
    \fi
    \dtldisplaycr\dtldisplaystarttab
      \@dtl@resetdoamp
      \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
      \in{#2}\do
      {%
        \expandafter\DTLifinlist\expandafter{\@dtl@key}{\@dtl@omitlist}%
        {}%
        {%
          \@dtl@doamp{\dtlheaderformat{\@dtl@head}}%
        }%
      }%
      \@dtl@resetdoamp
      \@dtl@resetdostartrow
    \dtldisplaycr\dtldisplayafterhead
    \endhead
  \fi
  \@sDTLforeach{#2}{}{%
    \@dtl@dostartrow
    \@dtl@resetdoamp
    \DTLforeachkeyinrow{\@dtl@val}%
    {%
      \global\let\@dtl@val\@dtl@val
      \expandafter\DTLifinlist\expandafter{\dtlkey}{\@dtl@omitlist}%
      {}%
      {%
        \@dtl@doamp
        \@dtl@datatype=0\dtltype\relax
        \ifcase\@dtl@datatype
          \dtlstringformat\@dtl@val
        \or
          \dtlintformat\@dtl@val
        \or
          \dtlrealformat\@dtl@val
        \or
          \dtlcurrencyformat\@dtl@val
        \fi
      }%
    }%
  }%
  \dtldisplayendtab
  \end{longtable}%
}
\newcommand*{\dtlswaprows}[3]{%
  \ifnum#2=#3\relax
  \else
    \ifnum#2<#3\relax
      \edef\@dtl@rowAidx{\number#2}%
```

```
      \edef\@dtl@rowBidx{\number#3}%
    \else
      \edef\@dtl@rowAidx{\number#3}%
      \edef\@dtl@rowBidx{\number#2}%
    \fi
    \edef\@dtl@dosplit{\noexpand\dtlgetrow{#1}{\@dtl@rowAidx}}%
    \@dtl@dosplit
    \expandafter\def\expandafter\@dtl@firstpart\expandafter
      {\the\dtlbeforerow}%
    \@dtl@toksA=\dtlcurrentrow
    \edef\@dtl@dosplit{\noexpand\@dtlgetrow
      {\the\dtlafterrow}{\@dtl@rowBidx}}%
    \@dtl@dosplit
    \expandafter\def\expandafter\@dtl@secondpart\expandafter
      {\the\dtlbeforerow}%
    \@dtl@toksB=\dtlcurrentrow
    \expandafter\def\expandafter\@dtl@thirdpart\expandafter
      {\the\dtlafterrow}%
    \toks@=\expandafter{\@dtl@firstpart}%
    \@dtl@toks=\expandafter{\@dtl@secondpart}%
    \edef\@dtl@tmp{\the\toks@
     \noexpand\db@row@elt@w%
     \noexpand\db@row@id@w \@dtl@rowAidx\noexpand\db@row@id@end@%
     \the\@dtl@toksB
     \noexpand\db@row@id@w \@dtl@rowAidx\noexpand\db@row@id@end@%
     \noexpand\db@row@elt@end@%
     \the\@dtl@toks}%
    \toks@=\expandafter{\@dtl@tmp}%
    \@dtl@toks=\expandafter{\@dtl@thirdpart}%
    \edef\@dtl@tmp{\the\toks@
     \noexpand\db@row@elt@w%
     \noexpand\db@row@id@w \@dtl@rowBidx\noexpand\db@row@id@end@%
     \the\@dtl@toksA
     \noexpand\db@row@id@w \@dtl@rowBidx\noexpand\db@row@id@end@%
     \noexpand\db@row@elt@end@%
    \the\@dtl@toks}%
    \expandafter\global\csname dtldb@#1\endcsname=\expandafter
      {\@dtl@tmp}%
  \fi
}
\newcommand*{\dtl@decrementrows}[2]{%
  \def\@dtl@newlist{}%
  \edef\@dtl@min{\number#2}%
  \expandafter\@dtl@decrementrows\the#1%
    \db@row@elt@w%
      \db@row@id@w \@nil\db@row@id@end@%
      \db@row@id@w \@nil\db@row@id@end@%
    \db@row@elt@end@%
    \@nil
  #1=\expandafter{\@dtl@newlist}%
```

```
}
\def\@dtl@decrementrows\db@row@elt@w\db@row@id@w #1\db@row@id@end@%
#2\db@row@id@w #3\db@row@id@end@\db@row@elt@end@#4\@nil{%
  \def\@dtl@thisrow{#1}%
  \ifx\@dtl@thisrow\@nnil
    \let\@dtl@donextdec=\@dtl@gobbletonil
  \else
    \ifnum\@dtl@thisrow>\@dtl@min
      \@dtl@tmpcount=\@dtl@thisrow\relax
      \advance\@dtl@tmpcount by -1\relax
      \toks@{#2}%
      \@dtl@toks=\expandafter{\@dtl@newlist}%
      \edef\@dtl@newlist{\the\@dtl@toks
        \noexpand\db@row@elt@w% row header
        \noexpand\db@row@id@w \number\@dtl@tmpcount
          \noexpand\db@row@id@end@% row id
         \the\toks@ % row contents
        \noexpand\db@row@id@w \number\@dtl@tmpcount
          \noexpand\db@row@id@end@% row id
        \noexpand\db@row@elt@end@% row end
      }%
    \else
      \toks@{#2}%
      \@dtl@toks=\expandafter{\@dtl@newlist}%
      \edef\@dtl@newlist{\the\@dtl@toks
        \noexpand\db@row@elt@w% row header
        \noexpand\db@row@id@w #1%
          \noexpand\db@row@id@end@% row id
         \the\toks@ % row contents
        \noexpand\db@row@id@w #3%
          \noexpand\db@row@id@end@% row id
        \noexpand\db@row@elt@end@% row end
      }%
    \fi
    \let\@dtl@donextdec=\@dtl@decrementrows
  \fi
  \@dtl@donextdec#4\@nil
}
\newcommand*{\DTLremoverow}[2]{%
  \DTLifdbexists{#1}%
  {%
    \ifnum#2>0\relax
      \expandafter\ifnum\csname dtlrows@#1\endcsname<#2\relax
        \expandafter\ifnum\csname dtlrows@#1\endcsname=1\relax
          \PackageError{datatool}{Can't remove row `\number#2' from
            database `#1': no such row}{Database `#1' only has
            1 row}%
        \else
          \PackageError{datatool}{Can't remove row `\number#2' from
            database `#1': no such row}{Database `#1' only has
```

```
            \expandafter\number\csname dtlrows@#1\endcsname\space
            rows}%
        \fi
      \else
        \@DTLremoverow{#1}{#2}%
      \fi
    \else
      \PackageError{datatool}{Can't remove row \number#2: index
        out of bounds}{Row indices start at 1}%
    \fi
  }%
  {%
    \PackageError{datatool}{Can't remove row: database `#1' doesn't
      exist}{}%
  }%
}
\newcommand*{\@DTLremoverow}[2]{%
      \edef\dtl@dogetrow{\noexpand\dtlgetrow{#1}{\number#2}}%
      \dtl@dogetrow
      \expandafter\dtl@decrementrows\expandafter
        {\dtlbeforerow}{#2}%
      \expandafter\dtl@decrementrows\expandafter
        {\dtlafterrow}{#2}%
      \edef\dtl@tmp{\the\dtlbeforerow \the\dtlafterrow}%
      \expandafter\global\csname dtldb@#1\endcsname
        =\expandafter{\dtl@tmp}%
      \expandafter\global\expandafter\advance
        \csname dtlrows@#1\endcsname by -1\relax
}
\newcommand*{\DTLsumforkeys}[1][\boolean{true}\and
 \DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlsumforkeys
}
\newcommand*{\@dtlsumforkeys}[4][]{%
  \def#4{0}%
  \@for\@dtl@db@name:=#2\do{%
    \@sDTLforeach{\@dtl@db@name}%
    {#1}% assignment list
    {%
      \@for\@dtl@key:=#3\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
        \dtlcurrentrow=\expandafter{\dtl@thisrow}%
        \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
        \expandafter\ifthenelse\expandafter{\@dtl@cond}%
          {\DTLadd{#4}{#4}{\DTLthisval}}{}%
      }%
    }%
  }%
}
```

```
\newcommand*{\DTLsumcolumn}[3]{%
  \def#3{0}%
  \DTLifdbexists{#1}%
  {%
     \@sDTLifhaskey{#1}{#2}%
     {%
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
         \DTLadd{#3}{#3}{\DTLthisval}%
       }%
     }%
     {%
       \PackageError{datatool}{Key `#2' doesn't
         exist in database `#1'}{}%
     }%
  }%
  {%
    \PackageError{datatool}{Data base `#1' doesn't
     exist}{}%
  }%
}
\newcommand*{\DTLmeanforkeys}[1][\boolean{true}\and
 \DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlmeanforkeys
}
\newcount\@dtl@elements
\newcommand*{\@dtlmeanforkeys}[4][]{%
  \def#4{0}%
  \@dtl@elements=0\relax
  \@for\@dtl@db@name:=#2\do{%
    \@sDTLforeach{\@dtl@db@name}%
    {#1}% assignment list
    {%
      \@for\@dtl@key:=#3\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
        \dtlcurrentrow=\expandafter{\dtl@thisrow}%
        \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
        \expandafter\ifthenelse\expandafter{\@dtl@cond}%
        {%
          \DTLadd{#4}{#4}{\DTLthisval}%
          \advance\@dtl@elements by 1\relax
        }{}%
      }%
    }%
  }%
  \ifnum\@dtl@elements=0\relax
    \PackageError{datatool}{Unable to evaluate mean: no data}{}%
  \else
    \edef\@dtl@n{\number\@dtl@elements}%
```

```
    \DTLdiv{#4}{#4}{\@dtl@n}%
  \fi
}
\newcommand*{\DTLmeanforcolumn}[3]{%
  \def#3{0}%
  \@dtl@elements=0\relax
  \DTLifdbexists{#1}%
  {%
     \@sDTLifhaskey{#1}{#2}%
     {%
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
         \DTLadd{#3}{#3}{\DTLthisval}%
         \advance\@dtl@elements by 1\relax
       }%
       \ifnum\@dtl@elements=0\relax
         \PackageError{datatool}{Can't compute mean for
          column `#2' in database `#1': no data}{}%
       \else
         \edef\@dtl@n{\number\@dtl@elements}%
         \DTLdiv{#3}{#3}{\@dtl@n}%
       \fi
     }%
     {%
       \PackageError{datatool}{Key `#2' doesn't
         exist in database `#1'}{}%
     }%
  }%
  {%
    \PackageError{datatool}{Data base `#1' doesn't
      exist}{}%
  }%
}
\newcommand*{\DTLvarianceforkeys}[1][\boolean{true}\and
 \DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlvarianceforkeys
}
\newcommand*{\@dtlvarianceforkeys}[4][]{%
  \@dtlmeanforkeys[#1]{#2}{#3}{\dtl@mean}%
  \def#4{0}%
  \@dtl@elements=0\relax
  \@for\@dtl@db@name:=#2\do{%
    \@sDTLforeach{\@dtl@db@name}%
    {#1}% assignment list
    {%
      \@for\@dtl@key:=#3\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
        \dtlcurrentrow=\expandafter{\dtl@thisrow}%
        \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
```

```
        \expandafter\ifthenelse\expandafter{\@dtl@cond}%
        {%
           \DTLsub{\dtl@diff}{\DTLthisval}{\dtl@mean}%
           \DTLmul{\dtl@diff}{\dtl@diff}{\dtl@diff}%
           \DTLadd{#4}{#4}{\dtl@diff}%
           \advance\@dtl@elements by 1\relax
        }{}%
      }%
    }%
  }%
  \ifnum\@dtl@elements=0\relax
    \PackageError{datatool}{Unable to evaluate variance: no data}{}%
  \else
    \edef\@dtl@n{\number\@dtl@elements}%
    \DTLdiv{#4}{#4}{\@dtl@n}%
  \fi
}
\newcommand*{\DTLvarianceforcolumn}[3]{%
  \DTLmeanforcolumn{#1}{#2}{\dtl@mean}%
  \def#3{0}%
  \@dtl@elements=0\relax
  \DTLifdbexists{#1}%
  {%
     \@sDTLifhaskey{#1}{#2}%
     {%
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
         \DTLsub{\dtl@diff}{\DTLthisval}{\dtl@mean}%
         \DTLmul{\dtl@diff}{\dtl@diff}{\dtl@diff}%
         \DTLadd{#3}{#3}{\dtl@diff}%
         \advance\@dtl@elements by 1\relax
       }%
       \ifnum\@dtl@elements=0\relax
         \PackageError{datatool}{Can't compute variance for
          column `#2' in database `#1': no data}{}%
       \else
         \edef\@dtl@n{\number\@dtl@elements}%
         \DTLdiv{#3}{#3}{\@dtl@n}%
       \fi
     }%
     {%
       \PackageError{datatool}{Key `#2' doesn't
         exist in database `#1'}{}%
     }%
  }%
  {%
    \PackageError{datatool}{Data base `#1' doesn't
      exist}{}%
  }%
}
```

```
\newcommand*{\DTLsdforkeys}[1][\boolean{true}\and
\DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlsdforkeys
}
\newcommand*{\@dtlsdforkeys}[4][]{%
  \@dtlvarianceforkeys[#1]{#2}{#3}{#4}%
  \DTLsqrt{#4}{#4}%
}
\newcommand*{\DTLsdforcolumn}[3]{%
  \DTLvarianceforcolumn{#1}{#2}{#3}%
  \DTLsqrt{#3}{#3}%
}
\newcommand*{\DTLminforkeys}[1][\boolean{true}\and
 \DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlminforkeys
}
\newcommand*{\@dtlminforkeys}[4][]{%
  \def#4{}%
  \@for\@dtl@db@name:=#2\do{%
    \@sDTLforeach{\@dtl@db@name}%
    {#1}% assignment list
    {%
      \@for\@dtl@key:=#3\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
        \dtlcurrentrow=\expandafter{\dtl@thisrow}%
        \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
        \expandafter\ifthenelse\expandafter{\@dtl@cond}%
        {%
          \ifdefempty{#4}%
          {%
            \let#4\DTLthisval
          }%
          {%
            \DTLmin{#4}{#4}{\DTLthisval}%
          }%
        }{}%
      }%
    }%
  }%
}
\newcommand*{\DTLminforcolumn}[3]{%
  \def#3{}%
  \DTLifdbexists{#1}%
  {%
     \@sDTLifhaskey{#1}{#2}%
     {%
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
```

```
        \ifdefempty{#3}%
        {%
          \let#3\DTLthisval
        }%
        {%
          \DTLmin{#3}{#3}{\DTLthisval}%
        }%
      }%
    }%
    {%
      \PackageError{datatool}{Key `#2' doesn't
        exist in database `#1'}{}%
    }%
  }%
  {%
    \PackageError{datatool}{Data base `#1' doesn't
     exist}{}%
  }%
}
\newcommand*{\DTLmaxforkeys}[1][\boolean{true}\and
 \DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlmaxforkeys
}
\newcommand*{\@dtlmaxforkeys}[4][]{%
  \def#4{}%
  \@for\@dtl@db@name:=#2\do{%
    \@sDTLforeach{\@dtl@db@name}%
    {#1}% assignment list
    {%
      \@for\@dtl@key:=#3\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
        \dtlcurrentrow=\expandafter{\dtl@thisrow}%
        \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
        \expandafter\ifthenelse\expandafter{\@dtl@cond}%
        {%
          \ifdefempty{#4}%
          {%
            \let#4\DTLthisval
          }%
          {%
            \DTLmax{#4}{#4}{\DTLthisval}%
          }%
        }{}%
      }%
    }%
  }%
}
\newcommand*{\DTLmaxforcolumn}[3]{%
  \def#3{}%
```

```
  \DTLifdbexists{#1}%
  {%
     \@sDTLifhaskey{#1}{#2}%
     {%
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
         \ifdefempty{#3}%
         {%
           \let#3\DTLthisval
         }%
         {%
           \DTLmax{#3}{#3}{\DTLthisval}%
         }%
       }%
     }%
     {%
       \PackageError{datatool}{Key `#2' doesn't
         exist in database `#1'}{}%
     }%
  }%
  {%
    \PackageError{datatool}{Data base `#1' doesn't
      exist}{}%
  }%
}
\newcommand*{\DTLcomputebounds}[8][\boolean{true}]{%
\let#5=\relax
\let#6=\relax
\let#7=\relax
\let#8=\relax
\@for\dtl@thisdb:=#2\do{%
  \@sDTLforeach[#1]{\dtl@thisdb}{\DTLthisX=#3,\DTLthisY=#4}{%
    \expandafter\DTLconverttodecimal\expandafter{\DTLthisX}{\dtl@decx}%
    \expandafter\DTLconverttodecimal\expandafter{\DTLthisY}{\dtl@decy}%
    \ifx#5\relax
      \let#5=\dtl@decx
      \let#6=\dtl@decy
      \let#7=\dtl@decx
      \let#8=\dtl@decy
    \else
      \dtlmin{#5}{#5}{\dtl@decx}%
      \dtlmin{#6}{#6}{\dtl@decy}%
      \dtlmax{#7}{#7}{\dtl@decx}%
      \dtlmax{#8}{#8}{\dtl@decy}%
    \fi
  }%
}%
}
\newcommand*{\DTLgetvalueforkey}[5]{%
  \DTLgetrowforkey{\@dtl@row}{#3}{#4}{#5}%
```

```
  \@sdtl@getcolumnindex{\@dtl@col}{#3}{#2}%
  {%
    \dtlcurrentrow=\expandafter{\@dtl@row}%
    \edef\@dtl@dogetval{\noexpand\dtlgetentryfromcurrentrow
      {\noexpand\@dtl@val}{\@dtl@col}}%
    \@dtl@dogetval
    \global\let#1=\@dtl@val
  }%
}
\newcommand*{\DTLgetrowforkey}[4]{%
  \global\let#1=\@empty
  \@sDTLforeach{#2}{\dtl@refvalue=#3}{%
    \DTLifnull{\dtl@refvalue}%
    {}%
    {%
      \ifthenelse{\equal{\dtl@refvalue}{#4}}%
      {%
        \xdef#1{\the\dtlcurrentrow}%
        \dtlbreak
      }%
      {}%
    }%
  }%
}
\newtoks\@dtl@list
\newcommand*{\DTLsort}{\@ifstar\@sDTLsort\@DTLsort}
\newcommand{\@DTLsort}[3][]{%
  \dtlsort[#1]{#2}{#3}{\dtlcompare}%
}
\newcommand*{\@sDTLsort}[3][]{%
  \dtlsort[#1]{#2}{#3}{\dtlicompare}%
}
\newcommand{\dtlsort}[4][]{%
  \DTLifdbexists{#3}%
  {%
     \ifnum\DTLrowcount{#3}>100\relax
       \typeout{Sorting `#3' - this may take a while.}%
     \fi
    \edef\@dtl@replacementkeys{#1}%
    \def\@dtl@sortorder{}%
    \@for\@dtl@level:=#2\do
    {%
      \expandafter\@dtl@getsortdirection\@dtl@level=\relax
      \DTLifhaskey{#3}{\@dtl@key}%
      {%
         \ifdefempty\@dtl@sortorder
         {\let\@dtl@sortorder=\@dtl@level}%
         {\eappto\@dtl@sortorder{,\@dtl@level}}%
      }%
      {%
```

```
          \PackageError{datatool}%
          {%
             Can't sort on `\@dtl@level'.
             No such key `\@dtl@key' in database `#3'%
          }{}%
        }%
      }%
      \ifdefempty\@dtl@sortorder
      {%
         \PackageWarning{datatool}{No keys provided to sort database `#3'}%
      }%
      {%
        \let\@dtl@comparecs=#4%
        \dtl@sortdata{#3}%
      }%
    }%
    {%
      \PackageError{datatool}{Database `#3' doesn't exist}{}%
    }%
}
\newtoks\@dtl@rowa
\newtoks\@dtl@rowb
\newcommand*{\dtl@sortdata}[1]{%
  \def\@dtl@sortedlist{}%
  \edef\@dtl@dbname{#1}%
  \@dtlforeachrow(\@dtl@rowAnum,\@dtl@rowAcontents)\in\@dtl@dbname\do{%
    \@dtl@rowa=\expandafter{\@dtl@rowAcontents}%
    \def\@dtl@newlist{}%
  \@dtl@insertdonefalse
  \dtlrownum=0\relax
  \expandafter\@dtl@foreachrow\@dtl@sortedlist
    \db@row@elt@w%
    \db@row@id@w \@nil\db@row@id@end@%
    \db@row@id@w \@nil\db@row@id@end@%
    \db@row@elt@end@%
    \@@{\@dtl@rowBnum}{\@dtl@rowBcontents}%
    {%
      \@dtl@rowb=\expandafter{\@dtl@rowBcontents}%
      \dtlrownum=\@dtl@rowBnum
      \if@dtl@insertdone
        \advance\dtlrownum by 1\relax
      \else
        \@dtl@sortcriteria{\@dtl@rowa}{\@dtl@rowb}%
        \ifnum\dtl@sortresult<0\relax
          \toks@=\expandafter{\@dtl@newlist}%
          \edef\@dtl@newlist{%
            \the\toks@
            \noexpand\db@row@elt@w%
             \noexpand\db@row@id@w \number\dtlrownum
             \noexpand\db@row@id@end@%
```

```
            \the\@dtl@rowa
            \noexpand\db@row@id@w \number\dtlrownum
            \noexpand\db@row@id@end@%
          \noexpand\db@row@elt@end@%
        }%
        \advance\dtlrownum by 1\relax
        \@dtl@insertdonetrue
      \fi
    \fi
    \toks@=\expandafter{\@dtl@newlist}%
    \edef\@dtl@newlist{\the\toks@
      \noexpand\db@row@elt@w%
       \noexpand\db@row@id@w \number\dtlrownum
       \noexpand\db@row@id@end@%
       \the\@dtl@rowb
       \noexpand\db@row@id@w \number\dtlrownum
       \noexpand\db@row@id@end@%
      \noexpand\db@row@elt@end@%
    }%
  }\q@nil
  \if@dtl@insertdone
  \else
    \advance\dtlrownum by 1\relax
    \toks@=\expandafter{\@dtl@newlist}%
    \edef\@dtl@newlist{\the\toks@
      \noexpand\db@row@elt@w%
       \noexpand\db@row@id@w \number\dtlrownum
       \noexpand\db@row@id@end@%
       \the\@dtl@rowa
       \noexpand\db@row@id@w \number\dtlrownum
       \noexpand\db@row@id@end@%
      \noexpand\db@row@elt@end@%
    }%
  \fi
  \let\@dtl@sortedlist=\@dtl@newlist
}%
\expandafter\global\csname dtldb@#1\endcsname=\expandafter
  {\@dtl@sortedlist}%
}
\newcommand{\@dtl@sortcriteria}[2]{%
  \@for\@dtl@level:=\@dtl@sortorder\do
  {%
    \expandafter\@dtl@getsortdirection\@dtl@level=\relax
    \let\@dtl@keya=\@dtl@key
    \let\@dtl@keyb=\@dtl@key
    \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@dbname}{\@dtl@key}%
    \dtlgetentryfromrow{\@dtl@a}{\@dtl@col}{#1}%
    \dtlgetentryfromrow{\@dtl@b}{\@dtl@col}{#2}%
    \ifx\@dtl@a\dtlnovalue
      \@dtl@setnull{\@dtl@a}{\@dtl@key}%
```

```
    \fi
    \ifx\@dtl@b\dtlnovalue
      \@dtl@setnull{\@dtl@b}{\@dtl@key}%
    \fi
    \DTLifnull{\@dtl@a}%
    {%
      \@for\@dtl@keya:=\@dtl@replacementkeys\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@dbname}{\@dtl@keya}%
        \dtlgetentryfromrow{\@dtl@a}{\@dtl@col}{#1}%
        \ifx\@dtl@a\dtlnovalue
          \@dtl@setnull{\@dtl@a}{\@dtl@key}%
        \fi
        \DTLifnull{\@dtl@a}{}{\@endfortrue}%
      }%
      \ifx\@dtl@keya\@nnil
        \let\@dtl@keya\@dtl@key
        \@dtl@setnull{\@dtl@a}{\@dtl@key}%
      \fi
    }%
    {}%
    \DTLifnull{\@dtl@b}%
    {%
      \@for\@dtl@keyb:=\@dtl@replacementkeys\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@dbname}{\@dtl@keyb}%
        \dtlgetentryfromrow{\@dtl@b}{\@dtl@col}{#2}%
        \ifx\@dtl@b\dtlnovalue
          \@dtl@setnull{\@dtl@b}{\@dtl@key}%
        \fi
        \DTLifnull{\@dtl@b}{}{\@endfortrue}%
      }%
      \ifx\@dtl@keyb\@nnil
        \let\@dtl@keyb\@dtl@key
        \@dtl@setnull{\@dtl@b}{\@dtl@key}%
      \fi
    }%
    {}%
    \@dtl@toksA=\expandafter{\@dtl@a}%
    \@dtl@toksB=\expandafter{\@dtl@b}%
    \edef\@dtl@docompare{\noexpand\dtl@compare@
      {\@dtl@keya}{\@dtl@keyb}%
      {\noexpand\@dtl@toksA}{\noexpand\@dtl@toksB}}%
    \@dtl@docompare
    \ifnum\dtl@sortresult=0\relax
      \@endforfalse
    \else
      \@endfortrue
    \fi
  }%
  \multiply\dtl@sortresult by -\@dtl@sortdirection\relax
}
```

```
\def\@dtl@getsortdirection#1=#2\relax{%
  \def\@dtl@key{#1}%
  \def\@dtl@sortdirection{#2}%
  \ifdefempty{\@dtl@sortdirection}%
  {%
    \def\@dtl@sortdirection{-1}%
  }%
  {%
    \@dtl@get@sortdirection#2%
    \def\@dtl@dir{ascending}%
    \ifx\@dtl@sortdirection\@dtl@dir
      \def\@dtl@sortdirection{-1}%
    \else
      \def\@dtl@dir{descending}%
      \ifx\@dtl@sortdirection\@dtl@dir
        \def\@dtl@sortdirection{1}%
      \else
        \PackageError{datatool}{Invalid sort direction
        `\@dtl@sortdirection'}{The sort direction can only be
         one of `ascending' or `descending'}%
        \def\@dtl@sortdirection{-1}%
      \fi
    \fi
  }%
}
\def\@dtl@get@sortdirection#1={\def\@dtl@sortdirection{#1}}
\newtoks\@dtl@toksA
\newtoks\@dtl@toksB
\newcommand{\dtl@compare}[3]{%
  \dtl@compare@{#1}{#1}{#2}{#3}%
}
\newcommand{\dtl@compare@}[4]{%
  \DTLgetdatatype{\@dtl@typeA}{\@dtl@dbname}{#1}%
  \ifx\@dtl@typeA\DTLunsettype
    \let\@dtl@typeA\DTLstringtype
  \fi
  \DTLgetdatatype{\@dtl@typeB}{\@dtl@dbname}{#2}%
  \ifx\@dtl@typeB\DTLunsettype
    \let\@dtl@typeB\DTLstringtype
  \fi
  \@dtl@tmpcount=\@dtl@typeA\relax
  \multiply\@dtl@tmpcount by \@dtl@typeB\relax
  \ifnum\@dtl@tmpcount=0\relax
    \edef\@dtl@tmpcmp{%
      \noexpand\@dtl@comparecs{\noexpand\dtl@sortresult}%
        {\the#3}{\the#4}%
      }%
    \@dtl@tmpcmp
    \ifdtlverbose
      \edef\@dtl@a{\the#3}%
```

```
      \edef\@dtl@b{\the#4}%
    \fi
  \else
    \edef\@dtl@a{\the#3}%
    \edef\@dtl@b{\the#4}%
    \DTLifnumlt{\@dtl@a}{\@dtl@b}%
    {%
      \dtl@sortresult=-1\relax
    }%
    {%
      \DTLifnumgt{\@dtl@a}{\@dtl@b}%
      {%
        \dtl@sortresult=1\relax
      }%
      {%
        \dtl@sortresult=0\relax
      }%
    }%
  \fi
  \ifdtlverbose
    \@onelevel@sanitize\@dtl@a
    \@onelevel@sanitize\@dtl@b
    \dtl@message{`\@dtl@a' <=> `\@dtl@b' = \number\dtl@sortresult}%
  \fi
}
\newwrite\@dtl@write
\newcommand*{\DTLsavedb}[2]{%
  \DTLifdbexists{#1}%
  {%
    \openout\@dtl@write=#2\relax
    \def\@dtl@header{}%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)%
      \in{#1}\do
    {%
      \IfSubStringInString{\@dtl@separator}{\@dtl@key}%
      {%
        \ifdefempty{\@dtl@header}%
        {%
          \protected@edef\@dtl@header{%
            \@dtl@delimiter\@dtl@key\@dtl@delimiter}%
        }%
        {%
          \toks@=\expandafter{\@dtl@header}%
          \protected@edef\@dtl@header{%
            \the\toks@\@dtl@separator
            \@dtl@delimiter\@dtl@key\@dtl@delimiter}%
        }%
      }%
      {%
        \ifdefempty{\@dtl@header}%
```

```
        {%
          \protected@edef\@dtl@header{\@dtl@key}%
        }%
        {%
          \toks@=\expandafter{\@dtl@header}%
          \protected@edef\@dtl@header{\the\toks@
            \@dtl@separator\@dtl@key}%
        }%
      }%
    }%
    \protected@write\@dtl@write{}{\@dtl@header}%
    \@sDTLforeach{#1}{}%
    {%
      \def\@dtl@row{}%
      \DTLforeachkeyinrow{\@dtl@val}%
      {%
        \IfSubStringInString{\@dtl@separator}{\@dtl@val}%
        {%
          \ifdefempty{\@dtl@row}%
          {%
            \protected@edef\@dtl@row{%
              \@dtl@delimiter\@dtl@val\@dtl@delimiter}%
          }%
          {%
            \toks@=\expandafter{\@dtl@row}%
            \protected@edef\@dtl@row{\the\toks@\@dtl@separator
              \@dtl@delimiter\@dtl@val\@dtl@delimiter}%
          }%
        }%
        {%
          \ifdefempty{\@dtl@row}%
          {%
            \protected@edef\@dtl@row{\@dtl@val}%
          }%
          {%
            \toks@=\expandafter{\@dtl@row}%
            \protected@edef\@dtl@row{\the\toks@\@dtl@separator
              \@dtl@val}%
          }%
        }%
      }%
        \protected@write\@dtl@write{}{\@dtl@row}%
    }%
    \closeout\@dtl@write
  }%
  {%
    \PackageError{datatool}{Can't save database `#1': no such
      database}{}%
  }%
}
```

```
\newcommand*{\DTLsavetexdb}[2]{%
  \DTLifdbexists{#1}%
  {%
    \openout\@dtl@write=#2\relax
    \protected@write\@dtl@write{}{\string\DTLnewdb{#1}}%
    \@sDTLforeach{#1}{}%
    {%
      \protected@write\@dtl@write{}{\string\DTLnewrow*{#1}}%
      \DTLforeachkeyinrow{\@dtl@val}%
      {%
        \DTLifnull{\@dtl@val}%
        {\def\@dtl@val{}}%
        {}%
        \protected@write\@dtl@write{}{%
          \string\DTLnewdbentry*{#1}{\dtlkey}{\@dtl@val}}%
      }%
    }%
    \dtlforeachkey(\@dtl@k,\@dtl@c,\@dtl@t,\@dtl@h)\in{#1}\do
    {%
      \@onelevel@sanitize\@dtl@h
      \protected@write\@dtl@write{}{%
        \string\DTLsetheader*{#1}{\@dtl@k}{\@dtl@h}}%
    }%
    \protected@write{\@dtl@write}{}{\string\def\string\dtllastloadeddb{#1}}%
    \closeout\@dtl@write
  }%
  {%
    \PackageError{datatool}{Can't save database `#1': no such
      database}{}%
  }%
}
\newcommand*{\dtl@saverawdbhook}{}
\newcommand*{\DTLsaverawdb}[2]{%
  \DTLifdbexists{#1}%
  {%
    \openout\@dtl@write=#2\relax
    \protected@write{\@dtl@write}{}{%
      \string\DTLifdbexists{#1}\expandafter\@gobble\string\%^^J%
      {%
         \string\PackageError{datatool}{Database `#1' ^^Jalready exists}{}%
         \expandafter\@gobble\string\%^^J%
         \string\aftergroup\string\endinput
      }%
      {%
      }\expandafter\@gobble\string\%
    }%
    {%
      \def\db@row@elt@w{\expandafter\@gobble\string\%^^J\string\db@row@elt@w\space}%
      \def\db@row@elt@end@{\expandafter\@gobble\string\%^^J\string\db@row@elt@end@\space
      \def\db@row@id@w{\expandafter\@gobble\string\%^^J\string\db@row@id@w\space}%
```

```
\def\db@row@id@end@{\expandafter\@gobble\string\%^^J\string\db@row@id@end@\space}%
\def\db@col@elt@w{\expandafter\@gobble\string\%^^J\string\db@col@elt@w\space}%
\def\db@col@elt@end@{\expandafter\@gobble\string\%^^J\string\db@col@elt@end@\space
\def\db@col@id@w{\expandafter\@gobble\string\%^^J\string\db@col@id@w\space}%
\def\db@col@id@end@{\expandafter\@gobble\string\%^^J\string\db@col@id@end@\space}%
\def\db@plist@elt@w{\expandafter\@gobble\string\%^^J\string\db@plist@elt@w\space}%
\def\db@plist@elt@end@{\expandafter\@gobble\string\%^^J\string\db@plist@elt@end@\s
\def\db@key@id@w{\expandafter\@gobble\string\%^^J\string\db@key@id@w\space}%
\def\db@key@id@end@{\expandafter\@gobble\string\%^^J\string\db@key@id@end@\space}%
\def\db@type@id@w{\expandafter\@gobble\string\%^^J\string\db@type@id@w\space}%
\def\db@type@id@end@{\expandafter\@gobble\string\%^^J\string\db@type@id@end@\space
\def\db@header@id@w{\expandafter\@gobble\string\%^^J\string\db@header@id@w\space}%
\def\db@header@id@end@{\expandafter\@gobble\string\%^^J\string\db@header@id@end@\s
\protected@write{\@dtl@write}{}{\string\bgroup\string\makeatletter}%
\protected@write{\@dtl@write}{}{%
  \string\dtl@message{Reconstructing database^^J`#1'}%
  \expandafter\@gobble\string\%}%
\protected@write{\@dtl@write}{}{%
 \string\expandafter
 \string\global\string\expandafter^^J\string\newtoks
 \string\csname\space dtlkeys@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{%
  \string\expandafter
  \string\global^^J
  \string\csname\space dtlkeys@#1\string\endcsname
 =\expandafter\@gobble\string\{\expandafter\@gobble\string\%}%
\expandafter\protected@xdef\csname dtl@rawwritedbkeys@#1\endcsname{%
  \the\csname dtlkeys@#1\endcsname}%
\protected@write{\@dtl@write}{}{\csname dtl@rawwritedbkeys@#1\endcsname}%
\protected@write{\@dtl@write}{}%
   {\expandafter\@gobble\string\}\expandafter\@gobble\string\%}%
\dtl@saverawdbhook
\protected@write{\@dtl@write}{}{%
 \string\expandafter\string\global
 \string\expandafter^^J\string\newtoks
   \string\csname\space dtldb@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{%
  \string\expandafter
  \string\global^^J\string\csname\space dtldb@#1\string\endcsname
 =\expandafter\@gobble\string\{\expandafter\@gobble\string\%}%
\expandafter\protected@xdef\csname dtl@rawwritedb@#1\endcsname{\the\csname dtldb@#
\protected@write{\@dtl@write}{}{\csname dtl@rawwritedb@#1\endcsname}%
\protected@write{\@dtl@write}{}{\expandafter\@gobble\string\}\expandafter\@gobble\
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\expandafter\string\newcount
 \string\csname\space dtlrows@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\csname\space dtlrows@#1\string\endcsname
  =\expandafter\number\csname dtlrows@#1\endcsname\string\relax}%
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
```

```
       \string\expandafter\string\newcount
       \string\csname\space dtlcols@#1\string\endcsname}%
      \protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
       \string\csname\space dtlcols@#1\string\endcsname
        =\expandafter\number\csname dtlcols@#1\endcsname\string\relax}%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \edef\dtl@tmp{%
        \string\expandafter^^J
        \string\gdef
        \string\csname\space dtl@ci@#1@\@dtl@key\string\endcsname
        {\csname dtl@ci@#1@\@dtl@key\endcsname}\expandafter\@gobble\string\%
      }%
      \expandafter\write\expandafter\@dtl@write\expandafter{\dtl@tmp}%
    }%
      \protected@write{\@dtl@write}{}{\string\egroup}%
    }%
    \protected@write{\@dtl@write}{}{\string\def\string\dtllastloadeddb{#1}}%
    \closeout\@dtl@write
  }%
  {%
    \PackageError{datatool}{Can't save database `#1': no such
      database}{}%
  }%
}
\newcommand*{\DTLprotectedsaverawdb}[2]{%
  \DTLifdbexists{#1}%
  {%
    \openout\@dtl@write=#2\relax
    \protected@write{\@dtl@write}{}{%
      \string\DTLifdbexists{#1}\expandafter\@gobble\string\%^^J%
      {%
         \string\PackageError{datatool}{Database `#1' ^^Jalready exists}{}%
         \expandafter\@gobble\string\%^^J%
         \string\aftergroup\string\endinput
      }%
      {%
      }\expandafter\@gobble\string\%
    }%
    {%
      \protected@write{\@dtl@write}{}{\string\bgroup\string\makeatletter}%
      \protected@write{\@dtl@write}{}{\string\dtl@message{Reconstructing database
        ^^J`#1'}\expandafter\@gobble\string\%}%
      \protected@write{\@dtl@write}{}{%
       \string\expandafter
       \string\global\string\expandafter^^J\string\newtoks
       \string\csname\space dtlkeys@#1\string\endcsname}%
      \protected@write{\@dtl@write}{}{%
        \string\expandafter
        \string\global^^J
```

```
        \string\csname\space dtlkeys@#1\string\endcsname
       =\expandafter\@gobble\string\{\expandafter\@gobble\string\%}%
      \edef\dtl@rawwrite@keys{\the\csname dtlkeys@#1\endcsname}%
      \@onelevel@sanitize\dtl@rawwrite@keys
      \expandafter\write\expandafter\@dtl@write\expandafter
         {\dtl@rawwrite@keys\expandafter\@gobble\string\}}%
      \protected@write{\@dtl@write}{}{%
       \string\expandafter\string\global
       \string\expandafter^^J\string\newtoks
         \string\csname\space dtldb@#1\string\endcsname}%
      \protected@write{\@dtl@write}{}{%
        \string\expandafter
        \string\global^^J\string\csname\space dtldb@#1\string\endcsname
       =\expandafter\@gobble\string\{\expandafter\@gobble\string\%}%
      \edef\dtl@rawwrite@db{\the\csname dtldb@#1\endcsname}%
      \@onelevel@sanitize\dtl@rawwrite@db
      \expandafter\write\expandafter\@dtl@write\expandafter
         {\dtl@rawwrite@db\expandafter\@gobble\string\}}%
      \protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
       \string\expandafter\string\newcount
       \string\csname\space dtlrows@#1\string\endcsname}%
      \protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
       \string\csname\space dtlrows@#1\string\endcsname
        =\expandafter\number\csname dtlrows@#1\endcsname\string\relax}%
      \protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
       \string\expandafter\string\newcount
       \string\csname\space dtlcols@#1\string\endcsname}%
      \protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
       \string\csname\space dtlcols@#1\string\endcsname
        =\expandafter\number\csname dtlcols@#1\endcsname\string\relax}%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \edef\dtl@tmp{%
        \string\expandafter^^J
        \string\gdef
        \string\csname\space dtl@ci@#1@\@dtl@key\string\endcsname
        {\csname dtl@ci@#1@\@dtl@key\endcsname}\expandafter\@gobble\string\%
      }%
      \expandafter\write\expandafter\@dtl@write\expandafter{\dtl@tmp}%
    }%
      \protected@write{\@dtl@write}{}{\string\egroup}%
    }%
    \protected@write{\@dtl@write}{}{\string\def\string\dtllastloadeddb{#1}}%
    \closeout\@dtl@write
  }%
  {%
    \PackageError{datatool}{Can't save database `#1': no such
      database}{}%
  }%
}
```

```
\newcommand*{\DTLloaddbtex}[2]{%
  \IfFileExists{#2}%
  {%
    \input{#2}%
    \ifdef#1%
    {%
      \PackageError{datatool}{Command \string#1\space is already defined}%
      {}%
    }%
    {%
      \let#1\dtllastloadeddb
    }%
  }%
  {%
    \PackageError{datatool}{File `#2' doesn't exist.}{}%
  }%
}
\newread\@dtl@read
\newcount\dtl@entrycr
\define@boolkey{loaddb}[dtl]{noheader}[true]{}
\define@boolkey{loaddb}[dtl]{autokeys}[true]{}
\dtlautokeysfalse
\define@key{loaddb}{keys}{%
  \dtl@entrycr=0\relax
  \@for\@dtl@key:=#1\do
  {%
    \advance\dtl@entrycr by 1\relax
    \expandafter
      \edef\csname @dtl@inky@\romannumeral\dtl@entrycr\endcsname{%
        \@dtl@key}%
  }%
}
\define@key{loaddb}{headers}{%
  \dtl@entrycr=0\relax
  \@for\@dtl@head:=#1\do
  {%
    \advance\dtl@entrycr by 1\relax
    \toks@=\expandafter{\@dtl@head}%
    \expandafter
      \edef\csname @dtl@inhd@\romannumeral\dtl@entrycr\endcsname{%
        \the\toks@}%
  }%
}
\newcount{\dtl@omitlines}
\define@key{loaddb}{omitlines}{\dtl@omitlines=#1\relax}
\newcommand*{\dtldefaultkey}{Column}
\newcommand*{\@dtl@readline}[2]{%
  \begingroup
  \catcode\endlinechar=\active%
  \global\read#1 to #2%
```

```
  \endgroup%
  \ifx#2\empty%
  \else%
    \expandafter\@dtl@stripeol#2%
    \let#2\@dtl@strippedline%
  \fi%
}
\begingroup
\catcode\endlinechar=\active%
\gdef\@dtl@stripeol#1
{\gdef\@dtl@strippedline{#1}}
\endgroup
\newcommand*{\@dtl@readrawline}[2]{%
  \@dtl@rawread#1 to #2%
  \dtl@domappings\@dtl@line
}
\newif\ifDTLnewdbonload
\DTLnewdbonloadtrue
\newcommand*{\DTLloaddb}{%
  \let\@dtl@doreadline\@dtl@readline
  \@dtlloaddb
}
\newcommand*{\@dtlloaddb}[3][]{%
  \IfFileExists{#3}{%
  \begingroup
    \catcode`\"12\relax
     \dtlnoheaderfalse
     \setkeys{loaddb}{#1}%
    \openin\@dtl@read=#3%
    \dtl@message{Reading `#3'}%
    \loop
    \ifnum \dtl@omitlines > \z@
      \advance\dtl@omitlines by \m@ne
      \read\@dtl@read to \@dtl@line
    \repeat
    \ifDTLnewdbonload
     \DTLnewdb{#2}%
    \fi
    \ifeof\@dtl@read
      \PackageWarning{datatool}{File `#3' has no data}%
    \else
      \ifdtlnoheader
      \else
        \loop
          \@dtl@conditionfalse
          \ifeof\@dtl@read
          \else
           \@dtl@doreadline\@dtl@read\@dtl@line
            \ifdefempty{\@dtl@line}%
            {%
```

```
      \@dtl@conditiontrue
    }%
    {%
    }%
  \fi
 \if@dtl@condition
 \repeat
 \protected@edef\@dtl@lin@{%
    \@dtl@separator\@dtl@line\@dtl@separator}%
 \dtl@entrycr=0\relax
\loop
 \expandafter\@dtl@lopoff\@dtl@lin@\to\@dtl@lin@\@dtl@key
 \advance\dtl@entrycr by 1\relax
 \ifdtlautokeys
   \csedef{@dtl@inky@\romannumeral\dtl@entrycr}%
     {\dtldefaultkey\number\dtl@entrycr}%
 \else
   \ifdefempty{\@dtl@key}%
   {%
     \edef\@dtl@key{\dtldefaultkey\number\dtl@entrycr}%
   }%
   {}%
 \fi
 \expandafter\@dtl@toks\expandafter{\@dtl@key}%
 \@ifundefined{@dtl@inky@\romannumeral\dtl@entrycr}%
 {%
   \expandafter
     \edef\csname @dtl@inky@\romannumeral
       \dtl@entrycr\endcsname{\the\@dtl@toks}%
 }%
 {%
   \@ifundefined{@dtl@inhd@\romannumeral\dtl@entrycr}%
   {%
     \expandafter
       \edef\csname @dtl@inhd@\romannumeral
         \dtl@entrycr\endcsname{\the\@dtl@toks}%
   }%
   {}%
 }%
 \ifx\@dtl@lin@\@dtl@separator
   \@dtl@conditionfalse
 \else
   \@dtl@conditiontrue
 \fi
\if@dtl@condition
\repeat
\fi
\ifeof\@dtl@read
   \ifdtlnoheader
     \PackageWarning{datatool}{No data in `#3'}%
```

```
   \else
     \PackageWarning{datatool}{Only header row found in `#3'}%
   \fi
\else
  \@dtl@conditiontrue
  \loop
   \@dtl@doreadline\@dtl@read\@dtl@line
    \ifdefempty{\@dtl@line}%
    {%
    }%
    {%
      \@sDTLnewrow{#2}%
      \expandafter\@dtl@toks\expandafter{\@dtl@line}%
      \edef\@dtl@lin@{\@dtl@separator\the\@dtl@toks
        \@dtl@separator}%
      \dtl@entrycr=0\relax
      {%
        \@dtl@conditiontrue
        \loop
          \expandafter\@dtl@lopoff\@dtl@lin@\to
            \@dtl@lin@\@dtl@thisentry
          \advance\dtl@entrycr by 1\relax
          \@ifundefined{@dtl@inky@\romannumeral\dtl@entrycr}%
          {%
            \edef\@dtl@thiskey{\dtldefaultkey
              \number\dtl@entrycr}%
            \expandafter\let
              \csname @dtl@inky@\romannumeral
                \dtl@entrycr\endcsname\@dtl@thiskey
          }%
          {%
            \edef\@dtl@thiskey{%
              \csname @dtl@inky@\romannumeral
                \dtl@entrycr\endcsname}%
          }%
          \expandafter\@dtl@toks\expandafter{\@dtl@thisentry}%
          \edef\@do@dtlnewentry{\noexpand\@sDTLnewdbentry
            {#2}{\@dtl@thiskey}{\the\@dtl@toks}}%
          \@do@dtlnewentry
          \ifx\@dtl@lin@\@dtl@separator
            \@dtl@conditionfalse
          \fi
        \if@dtl@condition
        \repeat
      }%
  }%
  \ifeof\@dtl@read \@dtl@conditionfalse\fi
  \if@dtl@condition
  \repeat
\fi
```

```
    \fi
    \closein\@dtl@read
     \edef\@dtl@maxcols{\expandafter
       \number\csname dtlcols@#2\endcsname}%
     \dtlgforint\dtl@entrycr=1\to\@dtl@maxcols\step1\do
     {%
       \@ifundefined{@dtl@inhd@\romannumeral\dtl@entrycr}%
       {}%
       {%
          \expandafter\let\expandafter\@dtl@head
            \csname @dtl@inhd@\romannumeral\dtl@entrycr\endcsname
          \@dtl@toks=\expandafter{\@dtl@head}%
          \edef\@dtl@dosetheader{\noexpand\@dtl@setheaderforindex
            {#2}{\number\dtl@entrycr}{\the\@dtl@toks}}%
          \@dtl@dosetheader
       }%
     }%
  \endgroup
  }{%
    \PackageError{datatool}{Can't load database `#2' (file `#3'
    doesn't exist)}{}%
  }%
}
\newcommand*\DTLloadrawdb{%
  \let\@dtl@doreadline\@dtl@readrawline
  \@dtlloaddb
}
\begingroup
\catcode`\%=\active
\catcode`$=\active
\catcode`&=\active
\catcode`~=\active
\catcode`_=\active
\catcode`^=\active
\catcode`#=\active
\catcode`?=6\relax
\catcode`<=1\relax
\catcode`>=2\relax
\catcode`\{=\active
\catcode`\}=\active
\gdef\@dtl@rawread?1to?2<\relax
<<\catcode`\%=\active
\catcode`$=\active
\catcode`&=\active
\catcode`~=\active
\catcode`_=\active
\catcode`^=\active
\catcode`#=\active
\catcode`\{=\active
\catcode`\}=\active
```

```
\def%<\noexpand\%>\relax
\def$<\noexpand\$>\relax
\def&<\&>\relax
\def#<\#>\relax
\def~<\noexpand\textasciitilde>\relax
\def_<\noexpand\_>\relax
\def^<\noexpand\textasciicircum>\relax
\@dtl@activatebraces
\@dtl@doreadraw?1?2>>>
\gdef\@dtl@doreadraw?1?2<\relax
\begingroup\catcode\endlinechar=\active\global\read?1 to \dtl@tmp\endgroup
\expandafter\@dtl@stripeol\dtl@tmp
\let\dtl@tmp\@dtl@strippedline
\protected@xdef?2<\dtl@tmp>\relax
>
\endgroup
\begingroup
\catcode`\{=\active
\catcode`\}=\active
\catcode`<=1\relax
\catcode`>=2\relax
\gdef\@dtl@activatebraces<%
 \catcode`\{=\active
 \catcode`\}=\active
 \def{<\noexpand\{>%
 \def}<\noexpand\}>%
>%
\endgroup
\newcommand*{\DTLrawmap}[2]{%
  \expandafter\@dtl@toks\expandafter{\@dtl@rawmappings}%
  \ifdefempty{\@dtl@rawmappings}%
  {%
    \def\@dtl@rawmappings{{#1}{#2}}%
  }%
  {%
    \def\@dtl@tmp{{#1}{#2}}%
    \protected@edef\@dtl@rawmappings{\the\@dtl@toks,\@dtl@tmp}%
  }%
}
\newcommand*{\@dtl@rawmappings}{}
\newcommand*{\dtl@domappings}[1]{%
  \@for\@dtl@map:=\@dtl@rawmappings\do{%
    \expandafter\DTLsubstituteall\expandafter#1\@dtl@map
  }%
}
\newcommand*{\dtlshowdb}[1]{%
  \expandafter\showthe\csname dtldb@#1\endcsname
}
\newcommand*{\dtlshowdbkeys}[1]{%
  \expandafter\showthe\csname dtlkeys@#1\endcsname
```

```
}
\newcommand*{\dtlshowtype}[2]{%
  \DTLgetdatatype{\@dtl@type}{#1}{#2}\show\@dtl@type
}
```

30.16 Rollback v2.32 (datatool-base-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datatool-base}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{etoolbox}
\RequirePackage{amsmath}
\RequirePackage{xkeyval}
\RequirePackage{xfor}
\RequirePackage{ifthen}
\RequirePackage{substr}[2009/10/20]
\ifundef{\ifdtlverbose}
{
  \define@boolkey{datatool-base.sty}[dtl]{verbose}[true]{}
}%
{}
\define@choicekey{datatool-base.sty}{math}[\val\nr]{fp,pgfmath}{%
  \renewcommand*\@dtl@mathprocessor{#1}%
}
\define@boolkey{datatool-base.sty}[@dtl@]{utf8}[true]{}
\ifdef\UTFviii@two@octets
{\booltrue{@dtl@utf8}}%
{\boolfalse{@dtl@utf8}}
\newcommand*{\dtlenableUTFviii}{\booltrue{@dtl@utf8}}
\newcommand*{\dtldisableUTFviii}{\boolfalse{@dtl@utf8}}
\providecommand*{\@dtl@mathprocessor}{fp}
\ProcessOptionsX
\RequirePackage{datatool-\@dtl@mathprocessor}[=v2.32]
\newcommand*{\dtl@message}[1]{%
  \ifdtlverbose\typeout{#1}\fi
}
\newtoks\@dtl@toks
\newcount\@dtl@tmpcount
\newlength\dtl@tmplength
\newcommand{\dtl@ifsingle}[3]{%
  \def\@dtl@arg{#1}%
  \ifdefempty{\@dtl@arg}%
  {%
    #3%
  }%
  {%
    \@dtl@ifsingle#1\@nil{#2}{#3}%
  }%
}
\def\@dtl@ifsingle#1#2\@nil#3#4{%
  \def\dtl@sg@arg{#2}%
```

```
  \ifdefempty{\dtl@sg@arg}%
  {%
   #3%
  }%
  {%
   #4%
  }%
}
\newcommand{\dtl@ifsingleorUTFviii}[3]{%
  \ifbool{@dtl@utf8}
  {%
    \def\@dtl@arg{#1}%
    \ifdefempty{\@dtl@arg}%
    {%
      #3%
    }%
    {%
      \expandafter\dtl@if@two@octets#1\relax\relax\dtl@end@if@two@octets
      {%
        \dtl@getfirst@UTFviii#1\@nil\end@dtl@getfirst@UTFviii
        \ifdefempty\dtl@rest{#2}{#3}%
      }%
      {%
        \@dtl@ifsingle#1\@nil{#2}{#3}%
      }%
    }%
  }%
  {%
    \dtl@ifsingle{#1}{#2}{#3}%
  }%
}%
\newcommand{\dtlifintopenbetween}[5]{%
  \ifnum#1>#2\relax
    \ifnum#1<#3\relax
      #4%
    \else
      #5%
    \fi
  \else
    #5%
  \fi
}
\newcommand{\dtlifintclosedbetween}[5]{%
  \dtlifintopenbetween{#1}{#2}{#3}{#4}%
  {%
    \ifnum#1=#2\relax
      #4%
    \else
      \ifnum#1=#3\relax
        #4%
```

```
      \else
        #5%
      \fi
    \fi
  }%
}
\long\def\long@collect@body#1{%
  \@envbody{\@xp#1\@xp{\the\@envbody}}%
  \edef\process@envbody{\the\@envbody\@nx\end{\@currenvir}}%
  \@envbody\@emptytoks \def\begin@stack{b}%
  \begingroup
  \@xp\let\csname\@currenvir\endcsname\long@collect@@body
  \edef\process@envbody{\@xp\@nx\csname\@currenvir\endcsname}%
  \process@envbody
}
\long\def\long@addto@envbody#1{%
  \toks@{#1}%
  \edef\@dtl@tmp{\the\@envbody\the\toks@}%
  \global\@envbody\@xp{\@dtl@tmp}%
}
\long\def\long@collect@@body#1\end#2{%
  \protected@edef\begin@stack{%
    \long@push@begins#1\begin\end \@xp\@gobble\begin@stack
  }%
  \ifx\@empty\begin@stack
    \endgroup
    \@checkend{#2}%
    \long@addto@envbody{#1}%
  \else
    \long@addto@envbody{#1\end{#2}}%
  \fi
  \process@envbody
}
\long\def\long@push@begins#1\begin#2{%
  \ifx\end#2\else b\@xp\long@push@begins\fi
}
\newcommand*{\DTLifinlist}[4]{%
  \def\@dtl@doifinlist##1,#1,##2\end@dtl@doifinlist{%
     \def\@before{##1}%
     \def\@after{##2}%
  }%
  \expandafter\@dtl@doifinlist\expandafter,#2,#1,\@nil
    \end@dtl@doifinlist
  \ifx\@after\@nnil
    #4%
  \else
    #3%
  \fi
}
\newif\ifDTLlistskipempty
```

```
\DTLlistskipemptytrue
\newrobustcmd{\DTLlistelement}[2]{%
  \begingroup
    \@dtl@tmpcount=0\relax
    \@for\@dtl@element:=#1\do{%
     \ifDTLlistskipempty
       \ifdefempty{\@dtl@element}%
       {}%
       {%
         \advance\@dtl@tmpcount by 1\relax%
         \ifnum\@dtl@tmpcount=#2 \@dtl@element\@endfortrue\fi
       }%
     \else
       \advance\@dtl@tmpcount by 1\relax%
       \ifnum\@dtl@tmpcount=#2 \@dtl@element\@endfortrue\fi
     \fi
    }%
    \if@endfor \else\@dtl@listelement@outofrange{#2}\fi
  \endgroup
}
\newrobustcmd{\DTLfetchlistelement}[3]{%
 \begingroup
   \@dtl@tmpcount=0\relax
   \@for\@dtl@element:=#1\do{%
    \ifDTLlistskipempty
      \ifdefempty{\@dtl@element}%
      {}%
      {%
        \advance\@dtl@tmpcount by 1\relax%
        \ifnum\@dtl@tmpcount=#2 \@endfortrue\fi
      }%
    \else
      \advance\@dtl@tmpcount by 1\relax%
      \ifnum\@dtl@tmpcount=#2 \@endfortrue\fi
    \fi
   }%
   \if@endfor \else\def\@dtl@element{\@dtl@listelement@outofrange{#2}}\fi
   \edef\x{%
     \endgroup
     \noexpand\def\noexpand#3{\expandonce\@dtl@element}%
   }\x
}
\newcommand{\@dtl@listelement@outofrange}[1]{%
  \PackageWarning{datatool-base}{List index `\number#1' out of range}%
}
\newrobustcmd{\DTLnumitemsinlist}[2]{%
  \@dtl@tmpcount=0\relax
  \@for\@dtl@element:=#1\do{%
     \ifDTLlistskipempty
       \ifdefempty{\@dtl@element}%
```

```
       {}%
       {\advance\@dtl@tmpcount by 1\relax}%
     \else
       \advance\@dtl@tmpcount by 1\relax
     \fi
  }%
  \edef#2{\number\@dtl@tmpcount}%
}
\newcommand*{\dtl@choplast}[3]{%
  \let#2\@empty
  \let#3\@empty
  \@for\@dtl@element:=#1\do{%
    \ifdefempty{#3}%
    {%
    }%
    {%
      \ifdefempty{#2}%
      {%
        \expandafter\toks@\expandafter{#3}%
        \edef#2{{\the\toks@}}%
      }%
      {%
        \expandafter\toks@\expandafter{#3}%
        \expandafter\@dtl@toks\expandafter{#2}%
        \edef#2{\the\@dtl@toks,{\the\toks@}}%
      }%
    }%
    \let#3=\@dtl@element%
  }%
}
\newcommand*{\dtl@chopfirst}[3]{%
  \let#2=\@empty
  \let#3=\@empty
  \@for\@dtl@element:=#1\do{%
    \let#2=\@dtl@element
    \@endfortrue
  }%
  \if@endfor
   \let#3=\@forremainder
  \fi
  \@endforfalse
}
\newcommand{\dtl@sortlist}[2]{%
\def\@dtl@sortedlist{}%
\@for\@dtl@currentrow:=#1\do{%
\expandafter\dtl@insertinto\expandafter
  {\@dtl@currentrow}{\@dtl@sortedlist}{#2}%
\@endforfalse}%
\let#1=\@dtl@sortedlist
}
```

```
\newcommand{\dtl@insertinto}[3]{%
  \def\@dtl@newsortedlist{}%
  \@dtl@insertdonefalse
  \@for\dtl@srtelement:=#2\do{%
    \if@dtl@insertdone
      \expandafter\toks@\expandafter{\dtl@srtelement}%
      \edef\@dtl@newstuff{{\the\toks@}}%
    \else
      \expandafter#3\expandafter{\dtl@srtelement}{#1}%
      \ifnum\dtl@sortresult<0\relax
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \@dtl@toks{#1}%
        \edef\@dtl@newstuff{{\the\@dtl@toks},{\the\toks@}}%
        \@dtl@insertdonetrue
      \else
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \edef\@dtl@newstuff{{\the\toks@}}%
      \fi
    \fi
    \ifdefempty{\@dtl@newsortedlist}%
    {%
      \expandafter\toks@\expandafter{\@dtl@newstuff}%
      \edef\@dtl@newsortedlist{\the\toks@}%
    }%
    {%
      \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
      \expandafter\@dtl@toks\expandafter{\@dtl@newstuff}%
      \edef\@dtl@newsortedlist{\the\toks@,\the\@dtl@toks}%
    }%
    \@endforfalse
  }%
  \ifdefempty{\@dtl@newsortedlist}%
  {%
    \@dtl@toks{#1}%
    \edef\@dtl@newsortedlist{{\the\@dtl@toks}}%
  }%
  {%
    \if@dtl@insertdone
    \else
      \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
      \@dtl@toks{#1}%
      \edef\@dtl@newsortedlist{\the\toks@,{\the\@dtl@toks}}%
    \fi
  }%
  \global\let#2=\@dtl@newsortedlist
}
\newcommand{\dtlsortlist}[2]{%
\def\@dtl@sortedlist{}%
\@for\@dtl@currentrow:=#1\do{%
\expandafter\dtlinsertinto\expandafter
```

```
  {\@dtl@currentrow}{\@dtl@sortedlist}{#2}%
\@endforfalse}%
\let#1=\@dtl@sortedlist
}
\newcommand{\dtlinsertinto}[3]{%
  \def\@dtl@newsortedlist{}%
  \@dtl@insertdonefalse
  \@for\dtl@srtelement:=#2\do{%
    \expandafter\DTLifSubString\expandafter{\dtl@srtelement}{,}
    {%
      \expandafter\toks@\expandafter{\dtl@srtelement}%
      \edef\dtl@srtelement{{\the\toks@}}%
    }%
    {%
    }
    \if@dtl@insertdone
      \let\@dtl@newstuff\dtl@srtelement
    \else
      \expandafter#3\expandafter\dtl@sortresult
        \expandafter{\dtl@srtelement}{#1}%
      \ifnum\dtl@sortresult>0\relax
        \DTLifSubString{#1}{,}%
        {%
          \@dtl@toks{{#1}}%
        }%
        {%
          \@dtl@toks{#1}%
        }%
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \edef\@dtl@newstuff{\the\@dtl@toks,\the\toks@}%
        \@dtl@insertdonetrue
      \else
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \edef\@dtl@newstuff{{\the\toks@}}%
        \let\@dtl@newstuff\dtl@srtelement
      \fi
    \fi
    \ifdefempty{\@dtl@newsortedlist}%
    {%
      \expandafter\toks@\expandafter{\@dtl@newstuff}%
      \edef\@dtl@newsortedlist{\the\toks@}%
    }%
    {%
      \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
      \expandafter\@dtl@toks\expandafter{\@dtl@newstuff}%
      \edef\@dtl@newsortedlist{\the\toks@,\the\@dtl@toks}%
    }%
    \@endforfalse
  }%
  \ifdefempty{\@dtl@newsortedlist}%
```

```
  {%
    \DTLifSubString{#1}{,}%
    {%
      \@dtl@toks{{#1}}%
    }%
    {%
      \@dtl@toks{#1}%
    }%
    \edef\@dtl@newsortedlist{\the\@dtl@toks}%
  }%
  {%
    \if@dtl@insertdone
    \else
      \DTLifSubString{#1}{,}%
      {%
        \@dtl@toks{{#1}}%
      }%
      {%
        \@dtl@toks{#1}%
      }%
      \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
      \edef\@dtl@newsortedlist{\the\toks@,\the\@dtl@toks}%
    \fi
  }%
  \global\let#2=\@dtl@newsortedlist
}
\newcommand*{\edtlinsertinto}[3]{%
  \protected@edef\dtl@srtelement{#1}%
  \expandafter\dtlinsertinto\expandafter{\dtl@srtelement}{#2}{#3}%
}
\newif\if@dtl@insertdone
\newcount\dtl@sortresult
\newcommand*{\DTLlistformatsep}{, }
\newcommand*{\DTLlistformatoxford}{}
\ifdef\andname
{\newcommand*{\DTLandname}{\andname}}
{\newcommand*{\DTLandname}{\&}}
\newcommand*{\DTLlistformatlastsep}{ \DTLandname\space}
\newcommand*{\DTLlistformatitem}[1]{#1}
\newcommand*{\@dtl@formatlist@handler}[1]{%
  \@dtl@formatlist@itemsep
  \@dtl@formatlist@lastitem
  \renewcommand{\@dtl@formatlist@lastitem}{%
    \renewcommand{\@dtl@formatlist@itemsep}{%
      \DTLlistformatsep
      \renewcommand*{\@dtl@formatlist@prelastitemsep}{%
        \DTLlistformatoxford}}%
    \renewcommand{\@dtl@formatlist@prelastitem}{%
      \@dtl@formatlist@prelastitemsep
       \DTLlistformatlastsep}%
```

```
    \DTLlistformatitem{#1}%
  }%
}%
\newrobustcmd*{\DTLformatlist}{%
 \@ifstar{\s@dtlformatlist}{\@dtlformatlist}%
}
\newcommand*{\s@dtlformatlist}[1]{%
  \def\@dtl@formatlist@itemsep{}%
  \def\@dtl@formatlist@lastitem{}%
  \def\@dtl@formatlist@prelastitem{}%
  \def\@dtl@formatlist@prelastitemsep{}%
  \@for\@dtl@formatlist@item:=#1\do{%
    \ifDTLlistskipempty
      \ifdefempty{\@dtl@formatlist@item}%
      {}%
      {\expandafter\@dtl@formatlist@handler\expandafter{\@dtl@formatlist@item}}%
    \else
      \expandafter\@dtl@formatlist@handler\expandafter{\@dtl@formatlist@item}%
    \fi
  }%
  \@dtl@formatlist@prelastitem\@dtl@formatlist@lastitem
}
\newcommand*{\@dtlformatlist}[1]{{\s@dtlformatlist{#1}}}
\newcommand{\@dtl@toks@gput@right@cx}[2]{%
  \def\@dtl@toks@name{#1}%
  \edef\@dtl@stuff{#2}%
  \global\csname\@dtl@toks@name\endcsname\expandafter
    \expandafter\expandafter{\expandafter\the
    \csname\expandafter\@dtl@toks@name\expandafter\endcsname\@dtl@stuff}%
}
\newcommand{\@dtl@toks@gconcat@middle@cx}[4]{%
  \def\@dtl@toks@name{#1}%
  \edef\@dtl@stuff{#3}%
  \global\csname\@dtl@toks@name\endcsname\expandafter\expandafter
    \expandafter\expandafter\expandafter
    \expandafter\expandafter{\expandafter\expandafter\expandafter
    \the\expandafter\expandafter\expandafter#2%
    \expandafter\@dtl@stuff\the#4}%
}
\newcount\@dtl@numgrpsepcount
\newcommand*{\@dtl@decimal}{.}
\newcommand*{\@dtl@numbergroupchar}{,}
\newcommand*{\DTLsetnumberchars}[2]{%
  \renewcommand*{\@dtl@numbergroupchar}{#1}%
  \renewcommand*{\@dtl@decimal}{#2}%
  \@dtl@construct@getnums
  \@dtl@construct@stripnumgrpchar{#1}%
}
\edef\@dtl@construct@getintfrac#1{%
  \noexpand\def\noexpand\@dtl@getintfrac##1#1##2\noexpand\relax{%
```

```
    \noexpand\@dtl@get@intpart{##1}%
    \noexpand\def\noexpand\@dtl@fracpart{##2}%
    \noexpand\ifdefempty{\noexpand\@dtl@fracpart}
    {%
      \noexpand\def\noexpand\@dtl@fracpart{0}%
    }%
    {%
      \noexpand\@dtl@getfracpart##2\noexpand\relax
      \noexpand\@dtl@choptrailingzeroes{\noexpand\@dtl@fracpart}%
    }%
  }%
  \noexpand\def\noexpand\@dtl@getfracpart##1#1\noexpand\relax{%
    \noexpand\def\noexpand\@dtl@fracpart{##1}%
  }%
  \noexpand\def\noexpand\DTLconverttodecimal##1##2{%
    \noexpand\dtl@ifsingle{##1}%
    {%
      \noexpand\expandafter\noexpand\toks@\noexpand\expandafter{##1}%
      \noexpand\edef\noexpand\@dtl@tmp{\noexpand\the\noexpand\toks@}%
    }%
    {%
      \noexpand\def\noexpand\@dtl@tmp{##1}%
    }%
    \noexpand\@dtl@standardize@currency\noexpand\@dtl@tmp
    \noexpand\ifdefempty{\noexpand\@dtl@org@currency}%
    {%
    }%
    {%
      \noexpand\let\noexpand\@dtl@currency\noexpand\@dtl@org@currency
    }%
    \noexpand\expandafter
      \noexpand\@dtl@getintfrac\noexpand\@dtl@tmp#1\noexpand\relax
    \noexpand\edef##2{\noexpand\@dtl@intpart.\noexpand\@dtl@fracpart}%
  }%
}
\newcommand*{\@dtl@construct@getnums}{%
  \expandafter\@dtl@construct@getintfrac\expandafter{\@dtl@decimal}%
}
\newcommand*{\@dtl@get@intpart}[1]{%
  \@dtl@tmpcount=1\relax
  \def\@dtl@intpart{#1}%
  \ifx\@dtl@intpart\@empty
    \def\@dtl@intpart{0}%
  \else
    \def\@dtl@intpart{}%
    \@dtl@get@int@part#1.\relax%
  \fi
  \ifnum\@dtl@tmpcount<0\relax
    \edef\@dtl@intpart{-\@dtl@intpart}%
  \fi
```

```
  \@dtl@strip@numgrpchar{\@dtl@intpart}%
}
\def\@dtl@get@int@part#1#2\relax{%
  \def\@dtl@argi{#1}%
  \def\@dtl@argii{#2}%
  \ifx\protect#1\relax%
    \let\@dtl@get@nextintpart=\@dtl@get@int@part
  \else
    \expandafter\ifx\@dtl@argi\$%
      \let\@dtl@get@nextintpart=\@dtl@get@int@part
    \else
      \ifx-#1%
        \multiply\@dtl@tmpcount by -1\relax
        \let\@dtl@get@nextintpart=\@dtl@get@int@part
      \else
        \if\@dtl@argi+%
          \let\@dtl@get@nextintpart=\@dtl@get@int@part
        \else
          \def\@dtl@intpart{#1}%
          \ifx.\@dtl@argii
            \let\@dtl@get@nextintpart=\@gobble
          \else
            \let\@dtl@get@nextintpart=\@dtl@get@next@intpart
          \fi
        \fi
      \fi
    \fi
  \fi
  \@dtl@get@nextintpart#2\relax
}
\def\@dtl@get@next@intpart#1.\relax{%
  \edef\@dtl@intpart{\@dtl@intpart#1}%
}
\newcommand*{\@dtl@choptrailingzeroes}[1]{%
  \def\@dtl@tmpcpz{}%
  \expandafter\@dtl@chop@trailingzeroes#1\@nil%
  \let#1=\@dtl@tmpcpz
}
\def\@dtl@chop@trailingzeroes#1#2\@nil{%
  \dtlifnumeq{#2}{0}%
  {%
    \edef\@dtl@tmpcpz{\@dtl@tmpcpz#1}%
    \let\@dtl@chopzeroesnext=\@dtl@gobbletonil
  }%
  {%
    \edef\@dtl@tmpcpz{\@dtl@tmpcpz#1}%
    \let\@dtl@chopzeroesnext=\@dtl@chop@trailingzeroes
  }%
  \@dtl@chopzeroesnext#2\@nil
}
```

```
\def\@dtl@gobbletonil#1\@nil{}
\newcommand*{\dtl@truncatedecimal}[1]{%
  \expandafter\@dtl@truncatedecimal#1.\@nil#1%
}
\def\@dtl@truncatedecimal#1.#2\@nil#3{%
  \def#3{#1}%
}
\newcommand*{\@dtl@strip@numgrpchar}[1]{%
  \def\@dtl@stripped{}%
  \edef\@dtl@do@stripnumgrpchar{%
    \noexpand\@@dtl@strip@numgrpchar#1\@dtl@numbergroupchar
    \noexpand\relax
  }%
  \@dtl@do@stripnumgrpchar
  \let#1=\@dtl@stripped
}
\edef\@dtl@construct@stripnumgrpchar#1{%
  \noexpand\def\noexpand\@@dtl@strip@numgrpchar##1#1##2\noexpand\relax{%
    \noexpand\expandafter\noexpand\toks@\noexpand\expandafter
      {\noexpand\@dtl@stripped}%
    \noexpand\edef\noexpand\@dtl@stripped{%
      \noexpand\the\noexpand\toks@
      ##1%
    }%
    \noexpand\def\noexpand\@dtl@tmp{##2}%
    \noexpand\ifx\noexpand\@dtl@tmp\noexpand\@empty
      \noexpand\let\noexpand\@dtl@next=\noexpand\relax
    \noexpand\else
      \noexpand\let\noexpand\@dtl@next=\noexpand\@@dtl@strip@numgrpchar
    \noexpand\fi
    \noexpand\@dtl@next##2\noexpand\relax
  }%
}
\newcommand*{\DTLdecimaltolocale}[2]{%
  \edef\@dtl@tmpdtl{#1}%
  \expandafter\@dtl@decimaltolocale\@dtl@tmpdtl.\relax
  \dtlifnumeq{\@dtl@fracpart}{0}%
  {%
    \edef#2{\@dtl@intpart}%
  }%
  {%
    \edef#2{\@dtl@intpart\@dtl@decimal\@dtl@fracpart}%
  }%
}
\def\@dtl@decimaltolocale#1.#2\relax{%
  \@dtl@decimaltolocaleint{#1}%
  \def\@dtl@fracpart{#2}%
  \ifdefempty\@dtl@fracpart
  {%
    \def\@dtl@fracpart{0}%
```

```
  }%
  {%
    \@dtl@decimaltolocalefrac#2\relax
  }%
}
\def\@dtl@decimaltolocaleint#1{%
  \@dtl@tmpcount=0\relax
  \@dtl@countdigits#1.\relax
  \@dtl@numgrpsepcount=\@dtl@tmpcount\relax
  \divide\@dtl@numgrpsepcount by 3\relax
  \multiply\@dtl@numgrpsepcount by 3\relax
  \advance\@dtl@numgrpsepcount by -\@dtl@tmpcount\relax
  \ifnum\@dtl@numgrpsepcount<0\relax
    \advance\@dtl@numgrpsepcount by 3\relax
  \fi
  \def\@dtl@intpart{}%
  \@dtl@decimal@to@localeint#1.\relax
}
\def\@dtl@countdigits#1#2\relax{%
  \advance\@dtl@tmpcount by 1\relax
  \ifx.#2\relax
    \let\@dtl@countnext=\@gobble
  \else
    \let\@dtl@countnext=\@dtl@countdigits
  \fi
  \@dtl@countnext#2\relax
}
\def\@dtl@decimal@to@localeint#1#2\relax{%
  \advance\@dtl@numgrpsepcount by 1\relax
  \ifx.#2\relax
    \edef\@dtl@intpart{\@dtl@intpart#1}%
    \let\@dtl@localeintnext=\@gobble
  \else
    \ifnum\@dtl@numgrpsepcount=3\relax
      \edef\@dtl@intpart{\@dtl@intpart#1\@dtl@numbergroupchar}%
      \@dtl@numgrpsepcount=0\relax
    \else
      \ifnum\@dtl@numgrpsepcount>3\relax
        \@dtl@numgrpsepcount=0\relax
      \fi
      \edef\@dtl@intpart{\@dtl@intpart#1}%
    \fi
    \let\@dtl@localeintnext=\@dtl@decimal@to@localeint
  \fi
  \@dtl@localeintnext#2\relax
}
\def\@dtl@decimaltolocalefrac#1.\relax{%
  \count@=0\relax
  \@dtl@digitcount#1\relax
  \ifnum\count@>9\relax
```

```
    \@dtl@chopexcessfrac#1000000000\@nil
  \else
    \def\@dtl@fracpart{#1}%
  \fi
}
\newcommand*{\@dtl@chopexcessfrac}[9]{%
  \def\@dtl@fracpart{#1#2#3#4#5#6#7#8#9}%
  \@dtl@gobbletonil
}
\newcommand*{\@dtl@digitcount}[1]{%
  \ifx\relax#1\relax
    \let\@dtl@digitcountnext\relax
  \else
    \advance\count@ by \@ne
    \let\@dtl@digitcountnext\@dtl@digitcount
  \fi
  \@dtl@digitcountnext
}
\newcommand*{\DTLdecimaltocurrency}[2]{%
  \edef\@dtl@tmpdtl{#1}%
  \expandafter\@dtl@decimaltolocale\@dtl@tmpdtl.\relax
  \dtl@truncatedecimal\@dtl@tmpdtl
  \@dtl@tmpcount=\@dtl@tmpdtl\relax
  \expandafter\@dtl@toks\expandafter{\@dtl@currency}%
  \dtlifnumeq{\@dtl@fracpart}{0}%
  {%
    \ifnum\@dtl@tmpcount<0\relax
      \@dtl@tmpcount = -\@dtl@tmpcount\relax
      \edef#2{-\the\@dtl@toks\the\@dtl@tmpcount\@dtl@decimal00}%
    \else
      \edef#2{\the\@dtl@toks\@dtl@intpart\@dtl@decimal00}%
    \fi
  }%
  {%
    \ifnum\@dtl@tmpcount<0\relax
      \@dtl@tmpcount = -\@dtl@tmpcount\relax
      \ifnum\@dtl@fracpart<10\relax
        \edef#2{%
          -\the\@dtl@toks\number\@dtl@tmpcount
          \@dtl@decimal\@dtl@fracpart0%
        }%
      \else
        \edef#2{%
          -\the\@dtl@toks\number\@dtl@tmpcount
          \@dtl@decimal\@dtl@fracpart
        }%
      \fi
    \else
      \ifnum\@dtl@fracpart<10\relax
        \edef#2{\the\@dtl@toks\@dtl@intpart\@dtl@decimal\@dtl@fracpart0}%
```

```
      \else
        \edef#2{\the\@dtl@toks\@dtl@intpart\@dtl@decimal\@dtl@fracpart}%
      \fi
    \fi
  }%
}
\@dtl@construct@getnums
\expandafter\@dtl@construct@stripnumgrpchar\expandafter
  {\@dtl@numbergroupchar}
\newcommand*{\@dtl@currencies}{\$,\pounds}
\newcommand*{\DTLnewcurrencysymbol}[1]{%
  \expandafter\toks@\expandafter{\@dtl@currencies}%
  \@dtl@toks{#1}%
  \edef\@dtl@currencies{\the\@dtl@toks,\the\toks@}%
}
\AtBeginDocument{%
  \@ifundefined{texteuro}{}{\DTLnewcurrencysymbol{\texteuro}}%
  \@ifundefined{textdollar}{}{\DTLnewcurrencysymbol{\textdollar}}%
  \@ifundefined{textstirling}{}{\DTLnewcurrencysymbol{\textstirling}}%
  \@ifundefined{textyen}{}{\DTLnewcurrencysymbol{\textyen}}%
  \@ifundefined{textwon}{}{\DTLnewcurrencysymbol{\textwon}}%
  \@ifundefined{textcurrency}{}{\DTLnewcurrencysymbol{\textcurrency}}%
  \@ifundefined{euro}{}{\DTLnewcurrencysymbol{\euro}}%
  \@ifundefined{yen}{}{\DTLnewcurrencysymbol{\yen}}%
}
\newcommand{\@dtl@standardize@currency}[1]{%
  \def\@dtl@org@currency{}%
  \@for\@dtl@thiscurrency:=\@dtl@currencies\do{%
    \expandafter\toks@\expandafter{\@dtl@thiscurrency}%
    \edef\@dtl@dosubs{\noexpand\DTLsubstitute{\noexpand#1}%
     {\the\toks@}{\noexpand\$}}%
    \@dtl@dosubs
    \ifdefempty{\@dtl@replaced}%
    {%
    }%
    {%
      \let\@dtl@org@currency=\@dtl@replaced
      \@endfortrue
    }%
  }%
  \@endforfalse
}
\newcommand*{\@dtl@currency}{\$}
\newcommand*{\DTLsetdefaultcurrency}[1]{%
  \renewcommand*{\@dtl@currency}{#1}%
}
\newcommand*{\DTLadd}[3]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \dtladd{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
```

```
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
\newcommand*{\DTLgadd}[3]{%
  \DTLadd{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLaddall}[2]{%
  \def\@dtl@sum{0}%
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \dtladd{\@dtl@sum}{\@dtl@sum}{\@dtl@num}%
  }%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@sum}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@sum}{#1}%
  }%
}
\newcommand*{\DTLgaddall}[2]{%
  \DTLaddall{\@dtl@tmpi}{#2}%
  \global\let#1=\@dtl@tmpi
}
\newcommand*{\DTLsub}[3]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \dtlsub{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
\newcommand*{\DTLgsub}[3]{%
  \DTLsub{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLmul}[3]{%
  \let\@dtl@thisreplaced=\@empty
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
```

```
  {%
  }%
  {%
    \let\@dtl@thisreplaced=\@dtl@replaced
  }%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \ifdefempty{\@dtl@replaced}%
  {%
  }%
  {%
    \let\@dtl@thisreplaced=\@dtl@replaced
  }%
  \dtlmul{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
  \ifdefempty{\@dtl@thisreplaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
\newcommand*{\DTLgmul}[3]{%
 \DTLmul{\@dtl@tmpii}{#2}{#3}%
 \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLdiv}[3]{%
  \let\@dtl@thisreplaced=\@empty
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
  {%
  }%
  {%
    \let\@dtl@thisreplaced=\@dtl@replaced
  }%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \dtldiv{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
  \ifdefempty{\@dtl@thisreplaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \ifdefequal{\@dtl@thisreplaced}{\@dtl@replaced}%
    {%
      \DTLdecimaltolocale{\@dtl@tmp}{#1}%
    }%
    {%
      \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
    }%
  }%
}
```

```
\newcommand*{\DTLgdiv}[3]{%
  \DTLdiv{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLabs}[2]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtlabs{\@dtl@tmp}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
\newcommand*{\DTLgabs}[2]{%
  \DTLabs{\@dtl@tmpii}{#2}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLneg}[2]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtlneg{\@dtl@tmp}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
\newcommand*{\DTLgneg}[2]{%
  \DTLneg{\@dtl@tmpii}{#2}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLsqrt}[2]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtlroot{\@dtl@tmpi}{\@dtl@numi}{2}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmpi}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmpi}{#1}%
  }%
}
\newcommand*{\DTLgsqrt}[2]{%
\DTLsqrt{\@dtl@tmpii}{#2}%
\global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLmin}[3]{%
```

```
 \DTLconverttodecimal{#2}{\@dtl@numi}%
 \DTLconverttodecimal{#3}{\@dtl@numii}%
 \dtlifnumlt{\@dtl@numi}{\@dtl@numii}%
 {%
   \dtl@ifsingle{#2}%
   {\let#1=#2}%
   {\def#1{#2}}%
 }%
 {%
  \dtl@ifsingle{#3}%
  {\let#1=#3}%
  {\def#1{#3}}%
 }%
}
\newcommand*{\DTLgmin}[3]{%
  \DTLmin{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLminall}[2]{%
  \let\@dtl@min=\@empty
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \ifdefempty{\@dtl@min}%
    {%
      \let\@dtl@min=\@dtl@num
    }%
    {%
      \dtlmin{\@dtl@min}{\@dtl@min}{\@dtl@num}%
    }%
  }%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@min}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@min}{#1}%
  }%
}
\newcommand*{\DTLgminall}[2]{%
  \DTLminall{\@dtl@tmpi}{#2}%
  \global\let#1=\@dtl@tmpi
}
\newcommand*{\DTLmax}[3]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \dtlmax{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
  \dtlifnumgt{\@dtl@numi}{\@dtl@numii}%
  {%
   \dtl@ifsingle{#2}%
   {\let#1=#2}%
```

```
   {\def#1{#2}}%
  }%
  {%
   \dtl@ifsingle{#3}%
   {\let#1=#3}%
   {\def#1{#3}}%
  }%
}
\newcommand*{\DTLgmax}[3]{%
  \DTLmax{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLmaxall}[2]{%
  \let\@dtl@max=\@empty
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \ifdefempty{\@dtl@max}%
    {%
      \let\@dtl@max\@dtl@num
    }%
    {%
      \dtlmax{\@dtl@max}{\@dtl@max}{\@dtl@num}%
    }%
  }%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@max}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@max}{#1}%
  }%
}
\newcommand*{\DTLgmaxall}[2]{%
\DTLmaxall{\@dtl@tmpi}{#2}%
\global\let#1=\@dtl@tmpi
}
\newcommand*{\DTLmeanforall}[2]{%
  \def\@dtl@mean{0}%
  \def\@dtl@n{0}%
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \dtladd{\@dtl@mean}{\@dtl@mean}{\@dtl@num}%
    \dtladd{\@dtl@n}{\@dtl@n}{1}%
  }%
  \dtldiv{\@dtl@mean}{\@dtl@mean}{\@dtl@n}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@mean}{#1}%
  }%
  {%
```

```
    \DTLdecimaltocurrency{\@dtl@mean}{#1}%
  }%
}
\newcommand*{\DTLgmeanforall}[2]{%
  \DTLmeanforall{\@dtl@tmpi}{#2}%
  \global\let#1=\@dtl@tmpi
}
\newcommand*{\DTLvarianceforall}[2]{%
  \def\@dtl@mean{0}%
  \def\@dtl@n{0}%
  \let\@dtl@decvals=\@empty
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \ifdefempty{\@dtl@decvals}%
    {%
     \let\@dtl@decvals=\@dtl@num
    }%
    {%
      \expandafter\toks@\expandafter{\@dtl@decvals}%
      \edef\@dtl@decvals{\the\toks@,\@dtl@num}%
    }%
    \dtladd{\@dtl@mean}{\@dtl@mean}{\@dtl@num}%
    \dtladd{\@dtl@n}{\@dtl@n}{1}%
  }%
  \dtldiv{\@dtl@mean}{\@dtl@mean}{\@dtl@n}%
  \def\@dtl@var{0}%
  \@for\@dtl@num:=\@dtl@decvals\do{%
    \dtlsub{\@dtl@diff}{\@dtl@num}{\@dtl@mean}%
    \dtlmul{\@dtl@diff}{\@dtl@diff}{\@dtl@diff}%
    \dtladd{\@dtl@var}{\@dtl@var}{\@dtl@diff}%
  }%
  \dtldiv{\@dtl@var}{\@dtl@var}{\@dtl@n}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@var}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@var}{#1}%
  }%
}
\newcommand*{\DTLgvarianceforall}[2]{%
  \DTLvarianceforall{\@dtl@tmpi}{#2}%
  \global\let#1=\@dtl@tmpi
}
\newcommand*{\DTLsdforall}[2]{%
  \def\@dtl@mean{0}%
  \def\@dtl@n{0}%
  \let\@dtl@decvals=\@empty
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
```

```
    \ifdefempty{\@dtl@decvals}%
    {%
     \let\@dtl@decvals=\@dtl@num
    }%
    {%
     \expandafter\toks@\expandafter{\@dtl@decvals}%
     \edef\@dtl@decvals{\the\toks@,\@dtl@num}%
    }%
    \dtladd{\@dtl@mean}{\@dtl@mean}{\@dtl@num}%
    \dtladd{\@dtl@n}{\@dtl@n}{1}%
  }%
  \dtldiv{\@dtl@mean}{\@dtl@mean}{\@dtl@n}%
  \def\@dtl@sd{0}%
  \@for\@dtl@num:=\@dtl@decvals\do{%
    \dtlsub{\@dtl@diff}{\@dtl@num}{\@dtl@mean}%
    \dtlmul{\@dtl@diff}{\@dtl@diff}{\@dtl@diff}%
    \dtladd{\@dtl@sd}{\@dtl@sd}{\@dtl@diff}%
  }%
  \dtldiv{\@dtl@sd}{\@dtl@sd}{\@dtl@n}%
  \dtlroot{\@dtl@sd}{\@dtl@sd}{2}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@sd}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@sd}{#1}%
  }%
}
\newcommand*{\DTLgsdforall}[2]{%
  \DTLsdforall{\@dtl@tmpi}{#2}%
  \global\let#1=\@dtl@tmpi
}
\newcommand*{\DTLround}[3]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtlround{\@dtl@tmp}{\@dtl@numi}{#3}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
\newcommand*{\DTLground}[3]{%
  \DTLround{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLtrunc}[3]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtltrunc{\@dtl@tmp}{\@dtl@numi}{#3}%
```

```
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
\newcommand*{\DTLgtrunc}[3]{%
  \DTLtrunc{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*{\DTLclip}[2]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtlclip{\@dtl@tmp}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
\newcommand*{\DTLgclip}[3]{%
  \DTLclip{\@dtl@tmpii}{#2}%
  \global\let#1=\@dtl@tmpii
}
\newcommand*\DTLinitials[1]{%
  \def\dtl@initialscmd{}%
  \dtl@subnobrsp{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{ }%
  \DTLsubstituteall{\dtl@string}{\ }{ }%
  \DTLsubstituteall{\dtl@string}{\space}{ }%
  \expandafter\dtl@initials\dtl@string{} \@nil%
  \dtl@initialscmd
}%
\edef\dtl@construct@subnobrsp{%
  \noexpand\def\noexpand\@dtl@subnobrsp##1\noexpand\protect
  \expandafter\noexpand\csname nobreakspace \endcsname ##2{%
  \noexpand\toks@{##1}%
  \noexpand\expandafter\noexpand\@dtl@toks\noexpand\expandafter{%
  \noexpand\@dtl@string}%
  \noexpand\edef\noexpand\@dtl@string{\noexpand\the\noexpand\@dtl@toks
  \noexpand\the\noexpand\toks@}%
  \noexpand\def\noexpand\@dtl@tmp{##2}%
  \noexpand\ifx\noexpand\@dtl@tmp\noexpand\@nnil
    \noexpand\let\noexpand\@dtl@subnobrspnext=\noexpand\relax
  \noexpand\else
    \noexpand\toks@{ }%
    \noexpand\expandafter\noexpand\@dtl@toks\noexpand\expandafter{%
```

```
    \noexpand\@dtl@string}%
    \noexpand\edef\noexpand\@dtl@string{\noexpand\the\noexpand\@dtl@toks
    \noexpand\the\noexpand\toks@}%
    \noexpand\let\noexpand\@dtl@subnobrspnext=\noexpand\@dtl@subnobrsp
  \noexpand\fi
  \noexpand\@dtl@subnobrspnext
  }%
  \noexpand\def\noexpand\dtl@subnobrsp##1##2{%
  \noexpand\def\noexpand\@dtl@string{}%
  \noexpand\@dtl@subnobrsp ##1\noexpand\protect\expandafter\noexpand
  \csname nobreakspace \endcsname \noexpand\@nil
  \noexpand\let##2=\noexpand\@dtl@string
  }%
}
\dtl@construct@subnobrsp
\newcommand*{\DTLstoreinitials}[2]{%
  \def\dtl@initialscmd{}%
  \dtl@subnobrsp{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{ }%
  \DTLsubstituteall{\dtl@string}{\ }{ }%
  \DTLsubstituteall{\dtl@string}{\space}{ }%
  \expandafter\dtl@initials\dtl@string{} \@nil
  \let#2=\dtl@initialscmd
}
\def\dtl@initials#1#2 #3{%
  \dtl@ifsingle{#1}%
  {%
    \ifcat\noexpand#1\relax\relax
       \def\@dtl@donextinitials{\@dtl@initials#2 {#3}}%
    \else
       \def\@dtl@donextinitials{\@dtl@initials#1#2 {#3}}%
    \fi
  }%
  {%
    \def\@dtl@donextinitials{\@dtl@initials{#1}#2 {#3}}%
  }%
  \@dtl@donextinitials
}
\def\@dtl@initials#1#2 #3{%
  \dtl@initialshyphen#2-{}\dtl@endhyp
  \expandafter\@dtl@toks\expandafter{\dtl@initialscmd}%
  \toks@{#1}%
  \ifdefempty{\dtl@inithyphen}%
  {%
  }%
  {%
    \edef\dtl@initialscmd{\the\@dtl@toks\the\toks@}%
    \expandafter\@dtl@toks\expandafter{\dtl@initialscmd}%
    \expandafter\toks@\expandafter{\dtl@inithyphen}%
  }%
```

```
  \def\dtl@tmp{#3}%
  \ifx\@nnil\dtl@tmp
   \edef\dtl@initialscmd{\the\@dtl@toks\the\toks@\DTLafterinitials}%
   \let\dtl@initialsnext=\@gobble
  \else
   \edef\dtl@initialscmd{\the\@dtl@toks\the\toks@\DTLbetweeninitials}%
   \let\dtl@initialsnext=\dtl@initials
  \fi
  \dtl@initialsnext{#3}%
}
\def\dtl@initialshyphen#1-#2#3\dtl@endhyp{%
  \def\dtl@inithyphen{#2}%
  \ifdefempty{\dtl@inithyphen}%
  {%
  }%
  {%
   \edef\dtl@inithyphen{%
     \DTLafterinitialbeforehyphen\DTLinitialhyphen#2}%
  }%
}
\newcommand*{\DTLafterinitials}{.}
\newcommand*{\DTLbetweeninitials}{.}
\newcommand*{\DTLafterinitialbeforehyphen}{.}
\newcommand*{\DTLinitialhyphen}{-}
\newcommand*{\DTLifAllUpperCase}[3]{%
  \protected@edef\dtl@tuc{#1}%
  \expandafter\dtl@testifuppercase\dtl@tuc\@nil\relax
  \if@dtl@condition#2\else#3\fi
}
\def\dtl@testifuppercase#1#2{%
  \def\dtl@argi{#1}%
  \def\dtl@argii{#2}%
  \def\dtl@tc@rest{}%
  \ifx\dtl@argi\@nnil
    \let\dtl@testifuppernext=\@nnil
  \else
    \ifx#1\protect
      \let\dtl@testifuppernext=\dtl@testifuppercase
    \else
      \ifx\uppercase#1\relax
        \@dtl@conditiontrue
        \def\dtl@tc@rest{}%
        \let\dtl@testifuppernext=\relax
      \else
        \edef\dtl@tc@arg{\string#1}%
        \expandafter\dtl@test@ifuppercase\dtl@tc@arg\end
        \ifx\dtl@argii\@nnil
          \let\dtl@testifuppernext=\@dtl@gobbletonil
        \fi
      \fi
```

```
    \fi
  \fi
  \ifx\dtl@testifuppernext\relax
   \edef\dtl@dotestifuppernext{%
     \noexpand\dtl@testifuppercase}%
  \else
   \ifx\dtl@testifuppernext\@nnil
     \edef\dtl@dotestifuppernext{#2}%
   \else
     \expandafter\toks@\expandafter{\dtl@tc@rest}%
     \@dtl@toks{#2}%
     \edef\dtl@dotestifuppernext{%
       \noexpand\dtl@testifuppernext\the\toks@\the\@dtl@toks}%
   \fi
  \fi
  \dtl@dotestifuppernext
}
\def\dtl@test@ifuppercase#1#2\end{%
  \def\dtl@tc@rest{#2}%
  \IfSubStringInString{\string\MakeUppercase}{#1#2}%
  {%
     \@dtl@conditiontrue
     \def\dtl@tc@rest{}%
     \let\dtl@testifuppernext=\relax
  }%
  {%
    \IfSubStringInString{\string\MakeTextUppercase}{#1#2}%
    {%
       \@dtl@conditiontrue
       \def\dtl@tc@rest{}%
       \let\dtl@testifuppernext=\relax
    }%
    {%
      \edef\dtl@uccode{\the\uccode`#1}%
      \edef\dtl@code{\number`#1}%
      \ifnum\dtl@code=\dtl@uccode\relax
        \@dtl@conditiontrue
        \let\dtl@testifuppernext=\dtl@testifuppercase
      \else
        \ifnum\dtl@uccode=0\relax
          \@dtl@conditiontrue
          \let\dtl@testifuppernext=\dtl@testifuppercase
        \else
          \@dtl@conditionfalse
          \let\dtl@testifuppernext=\@dtl@gobbletonil
        \fi
      \fi
    }%
  }%
}
```

```
\newcommand*{\DTLifAllLowerCase}[3]{%
  \protected@edef\dtl@tlc{#1}%
  \expandafter\dtl@testiflowercase\dtl@tlc\@nil\relax
  \if@dtl@condition#2\else#3\fi
}
\def\dtl@testiflowercase#1#2{%
  \def\dtl@argi{#1}%
  \def\dtl@argii{#2}%
  \ifx\dtl@argi\@nnil
    \let\dtl@testiflowernext=\@nnil
  \else
    \ifx#1\protect
      \let\dtl@testiflowernext=\dtl@testiflowercase
    \else
      \ifx\lowercase#1\relax
        \@dtl@conditiontrue
        \def\dtl@tc@rest{}%
        \let\dtl@testiflowernext=\relax
      \else
        \edef\dtl@tc@arg{\string#1}%
        \expandafter\dtl@test@iflowercase\dtl@tc@arg\end
        \ifx\dtl@argii\@nnil
          \let\dtl@testiflowernext=\@dtl@gobbletonil
        \fi
      \fi
    \fi
  \fi
  \ifx\dtl@testiflowernext\relax
   \edef\dtl@dotestiflowernext{%
     \noexpand\dtl@testiflowercase}%
  \else
   \ifx\dtl@testiflowernext\@nnil
     \edef\dtl@dotestiflowernext{#2}%
   \else
     \expandafter\toks@\expandafter{\dtl@tc@rest}%
     \@dtl@toks{#2}%
     \edef\dtl@dotestiflowernext{%
       \noexpand\dtl@testiflowernext\the\toks@\the\@dtl@toks}%
   \fi
  \fi
  \dtl@dotestiflowernext
}
\def\dtl@test@iflowercase#1#2\end{%
  \def\dtl@tc@rest{#2}%
  \IfSubStringInString{\string\MakeLowercase}{#1#2}%
  {%
    \@dtl@conditiontrue
    \def\dtl@tc@rest{}%
    \let\dtl@testiflowernext=\relax
  }%
```

```
  {%
    \IfSubStringInString{\string\MakeTextLowercase}{#1#2}%
    {%
      \@dtl@conditiontrue
      \def\dtl@tc@rest{}%
      \let\dtl@testiflowernext=\relax
    }%
    {%
      \edef\dtl@lccode{\the\lccode`#1}%
      \edef\dtl@code{\number`#1}%
      \ifnum\dtl@code=\dtl@lccode\relax
        \@dtl@conditiontrue
        \let\dtl@testiflowernext=\dtl@testiflowercase
      \else
        \ifnum\dtl@lccode=0\relax
          \@dtl@conditiontrue
          \let\dtl@testiflowernext=\dtl@testiflowercase
        \else
          \@dtl@conditionfalse
          \let\dtl@testiflowernext=\@dtl@gobbletonil
        \fi
      \fi
    }%
  }%
}
\newcommand{\DTLsubstitute}[3]{%
  \expandafter\DTLsplitstring\expandafter
    {#1}{#2}{\@dtl@beforepart}{\@dtl@afterpart}%
  \ifdefempty{\@dtl@replaced}%
  {%
  }%
  {%
    \def#1{}%
    \expandafter\@dtl@toks\expandafter{\@dtl@beforepart}%
    \expandafter\toks@\expandafter{#1}%
    \protected@edef#1{\the\toks@\the\@dtl@toks#3}%
    \expandafter\@dtl@toks\expandafter{\@dtl@afterpart}%
    \expandafter\toks@\expandafter{#1}%
    \edef#1{\the\toks@\the\@dtl@toks}%
  }%
}
\newcommand*{\DTLsplitstring}[4]{%
  \def\dtl@splitstr##1#2##2\@nil{%
    \def#3{##1}%
    \def#4{##2}%
    \ifdefempty{#4}%
    {%
      \let\@dtl@replaced=\@empty
    }%
    {%
```

```
      \def\@dtl@replaced{#2}%
      \dtl@split@str##2\@nil
    }%
  }%
  \def\dtl@split@str##1#2\@nil{\def#4{##1}}%
  \dtl@splitstr#1#2\@nil
}
\newcommand{\DTLsubstituteall}[3]{%
  \def\@dtl@splitsubstr{}%
  \let\@dtl@afterpart=#1\relax
  \@dtl@dosubstitute{#2}{#3}%
  \expandafter\toks@\expandafter{\@dtl@splitsubstr}%
  \expandafter\@dtl@toks\expandafter{\@dtl@afterpart}%
  \long\edef#1{\the\toks@\the\@dtl@toks}%
}
\def\@dtl@dosubstitute#1#2{%
  \expandafter\DTLsplitstring\expandafter
   {\@dtl@afterpart}{#1}{\@dtl@beforepart}{\@dtl@afterpart}%
  \expandafter\toks@\expandafter{\@dtl@splitsubstr}%
  \expandafter\@dtl@toks\expandafter{\@dtl@beforepart}%
  \edef\@dtl@splitsubstr{\the\toks@\the\@dtl@toks}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \let\@dtl@dosubstnext=\@dtl@dosubstitutenoop
  }%
  {%
    \expandafter\toks@\expandafter{\@dtl@splitsubstr}%
    \@dtl@toks{#2}%
    \edef\@dtl@splitsubstr{\the\toks@\the\@dtl@toks}%
    \let\@dtl@dosubstnext=\@dtl@dosubstitute
  }%
  \@dtl@dosubstnext{#1}{#2}%
}
\def\@dtl@dosubstitutenoop#1#2{}
\newif\if@dtl@condition
\newcount\@dtl@datatype
\newcommand{\@dtl@checknumerical}[1]{%
  \@dtl@numgrpsepfalse
  \dtl@ifsingle{#1}%
  {%
    \expandafter\toks@\expandafter{#1}%
    \edef\@dtl@tmp{\the\toks@}%
  }%
  {%
    \def\@dtl@tmp{#1}%
  }%
  \ifdefempty\@dtl@tmp
  {%
    \@dtl@datatype=0\relax
  }%
```

```
  {%
    \@dtl@tmpcount=0\relax
    \@dtl@datatype=0\relax
    \@dtl@numgrpsepcount=2\relax
    \@dtl@standardize@currency\@dtl@tmp
    \ifdefempty{\@dtl@org@currency}%
    {%
    }%
    {%
      \let\@dtl@currency\@dtl@org@currency
    }%
    \expandafter\@dtl@checknumericalstart\@dtl@tmp\@nil\@nil
  }%
  \ifnum\@dtl@numgrpsepcount>-1\relax
    \if@dtl@numgrpsep
      \ifnum\@dtl@numgrpsepcount=3\relax
      \else
        \@dtl@datatype=0\relax
      \fi
    \fi
  \fi
}
\newcommand*{\@dtl@protect}{\protect}
\newcommand*{\@dtl@minus}{-}
\newcommand*{\@dtl@plus}{+}
\newcommand*{\@dtl@dollar}{\$}
\def\@dtl@checknumericalstart#1#2\@nil\@nil{%
  \def\@dtl@tmp{#1}%
  \ifx\@dtl@tmp\@dtl@protect
    \@dtl@checknumericalstart#2\@nil\@nil\relax
  \else
    \ifx\@dtl@tmp\@dtl@minus
      \def\@dtl@tmp{#2}%
      \ifdefempty{\@dtl@tmp}%
      {%
        \@dtl@datatype=0\relax
      }%
      {%
        \ifnum\@dtl@datatype=0\relax
          \@dtl@datatype=1\relax
        \fi
        \@dtl@checknumericalstart#2\@nil\@nil\relax
      }%
    \else
      \ifx\@dtl@tmp\@dtl@plus
        \def\@dtl@tmp{#2}%
        \ifdefempty{\@dtl@tmp}%
        {%
          \@dtl@datatype=0\relax
        }%
```

```
        {%
          \ifnum\@dtl@datatype=0\relax
            \@dtl@datatype=1\relax
          \fi
          \@dtl@checknumericalstart#2\@nil\@nil\relax
        }%
      \else
        \def\@dtl@tmp{#1}%
        \ifx\@dtl@tmp\@dtl@dollar
          \def\@dtl@tmp{#2}%
          \ifdefempty{\@dtl@tmp}%
          {%
            \@dtl@datatype=0\relax
          }%
          {%
            \@dtl@datatype=3\relax
            \@dtl@checknumericalstart#2\@nil\@nil\relax
          }%
        \else
          \ifdefempty{\@dtl@tmp}%
          {%
            \@dtl@datatype=0\relax
          }%
          {%
            \ifnum\@dtl@datatype=0\relax
              \@dtl@datatype=1\relax
            \fi
            \@dtl@checknumericalloop#1#2\@nil\@nil\relax
          }%
        \fi
      \fi
    \fi
  \fi
}
\newif\if@dtl@numgrpsep
\newcommand*{\@dtl@ifDigitOrDecimalSep}[3]{%
  \ifnum 9<1\noexpand#1\relax
    #2%
  \else
    \expandafter\ifx\@dtl@decimal#1\relax
      #2%
    \else
      #3%
    \fi
  \fi
}
\def\@dtl@checknumericalloop#1#2\@nil{%
\def\@dtl@tmp{#1}%
\ifx\@nnil\@dtl@tmp\relax
 \let\@dtl@chcknumnext=\@dtl@checknumericalnoop%
```

```
\else
  \@dtl@ifDigitOrDecimalSep{#1}{%
  \let\@dtl@chcknumnext=\@dtl@checknumericalloop%
  \expandafter\ifx\@dtl@decimal#1\relax
     \if@dtl@numgrpsep
       \ifnum\@dtl@numgrpsepcount=3\relax
        \@dtl@numgrpsepcount=-1\relax
       \else
        \@dtl@datatype=0\relax
        \let\@dtl@chcknumnext=\@dtl@checknumericalnoop
       \fi
     \else
       \@dtl@numgrpsepcount=-1\relax
     \fi
   \else
     \ifnum\@dtl@numgrpsepcount=-1\relax
     \else
       \advance\@dtl@numgrpsepcount by 1\relax
     \fi
   \fi
}{%
\ifx\@dtl@numbergroupchar\@dtl@tmp\relax
  \@dtl@numgrpseptrue
  \ifnum\@dtl@numgrpsepcount<3\relax
    \@dtl@datatype=0\relax
    \let\@dtl@chcknumnext=\@dtl@checknumericalnoop
  \else
     \@dtl@numgrpsepcount=0\relax
  \fi
\else
  \@dtl@datatype=0\relax
  \let\@dtl@chcknumnext=\@dtl@checknumericalnoop
\fi
}%
  \ifx\@dtl@decimal\@dtl@tmp\relax
   \ifnum\@dtl@datatype<3\relax
     \@dtl@datatype=2\relax
   \fi
   \advance\@dtl@tmpcount by 1\relax
   \ifnum\@dtl@tmpcount>1\relax
     \@dtl@datatype=0\relax
     \let\@dtl@chcknumnext=\@dtl@checknumericalnoop%
   \fi
  \fi
\fi
\@dtl@chcknumnext#2\@nil
}
\def\@dtl@checknumericalnoop#1\@nil#2{}
\newcommand{\DTLifnumerical}[3]{%
\@dtl@checknumerical{#1}%
```

```
\ifnum\@dtl@datatype=0\relax#3\else#2\fi
}
\newcommand{\DTLifreal}[3]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=2\relax #2\else #3\fi
}
\newcommand{\DTLifint}[3]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=1\relax #2\else #3\fi
}
\newcommand{\DTLifstring}[3]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=0\relax #2\else #3\fi
}
\newcommand{\DTLifcurrency}[3]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=3\relax #2\else #3\fi
}
\newcommand*{\DTLifcurrencyunit}[4]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=3\relax
    \ifthenelse{\equal{\@dtl@org@currency}{#2}}{#3}{#4}%
  \else
    #4%
  \fi
}
\newcommand{\DTLifcasedatatype}[5]{%
  \@dtl@checknumerical{#1}%
  \ifcase\@dtl@datatype
   #2% string
  \or
   #3% integer
  \or
   #4% number
  \or
   #5% currency
  \fi
}
\newcommand*{\dtl@testbothnumerical}[2]{%
  \dtl@ifsingle{#1}{%
   \edef\@dtl@tmp{#1}}{%
   \def\@dtl@tmp{#1}}%
  \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
  \edef\@dtl@firsttype{\number\@dtl@datatype}%
  \dtl@ifsingle{#2}{%
   \edef\@dtl@tmp{#2}}{%
   \def\@dtl@tmp{#2}}%
  \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
  \multiply\@dtl@datatype by \@dtl@firsttype\relax
  \ifnum\@dtl@datatype>0\relax
```

```
   \@dtl@conditiontrue
  \else
   \@dtl@conditionfalse
  \fi
}
\newcommand*{\DTLifnumlt}[4]{%
  \DTLconverttodecimal{#1}{\@dtl@numi}%
  \DTLconverttodecimal{#2}{\@dtl@numii}%
  \dtlifnumlt{\@dtl@numi}{\@dtl@numii}%
  {%
    #3%
  }%
  {%
    #4%
  }%
}
\newif\ifdtlcompareskipcs
\dtlcompareskipcsfalse
\newcommand*{\dtlcompare}[3]{%
  \dtl@subnobrsp{#2}{\@dtl@argA}%
  \dtl@subnobrsp{#3}{\@dtl@argB}%
  \ifdefempty{\@dtl@argA}%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      #1=0\relax
    }%
    {%
      #1=-1\relax
    }%
  }%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      #1=1\relax
    }%
    {%
      \dtl@setwordbreaksnohyphens{\@dtl@argA}{\@dtl@wordbreak}%
      \let\@dtl@argA\dtl@string
      \dtl@setwordbreaksnohyphens{\@dtl@argB}{\@dtl@wordbreak}%
      \let\@dtl@argB\dtl@string
      \expandafter\dtl@getfirst\@dtl@argA\end@dtl@getfirst
      \let\dtl@firstA=\dtl@first
      \let\dtl@restA=\dtl@rest
      \expandafter\dtl@getfirst\@dtl@argB\end@dtl@getfirst
      \let\dtl@firstB=\dtl@first
      \let\dtl@restB=\dtl@rest
      \expandafter\dtl@ifsingleorUTFviii\expandafter{\dtl@firstA}%
      {%
        \expandafter\dtl@ifsingleorUTFviii\expandafter{\dtl@firstB}%
```

```
{%
  \expandafter\dtl@setcharcode\expandafter{\dtl@firstA}{\dtl@codeA}%
  \expandafter\dtl@setcharcode\expandafter{\dtl@firstB}{\dtl@codeB}%
  \let\dtl@donextcompare\@firstofone
  \ifdtlcompareskipcs
    \ifnum\dtl@codeA=0\relax
      \ifnum\dtl@codeB=0\relax
         \edef\dtl@donext{%
           \noexpand\dtlcompare
           {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
         \dtl@donext
         \let\dtl@donextcompare\@gobble
      \else
         \edef\dtl@donext{%
           \noexpand\dtlcompare
           {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\@dtl@argB}}%
         \dtl@donext
         \let\dtl@donextcompare\@gobble
      \fi
    \else
      \ifnum\dtl@codeB=0\relax
         \edef\dtl@donext{%
           \noexpand\dtlcompare
           {\noexpand#1}{\expandonce\@dtl@argA}{\expandonce\dtl@restB}}%
         \dtl@donext
         \let\dtl@donextcompare\@gobble
      \fi
    \fi
  \fi
  \dtl@donextcompare
  {%
    \ifnum\dtl@codeA=-1\relax
      \ifnum\dtl@codeB=-1\relax
         \edef\dtl@donext{%
           \noexpand\dtlcompare
           {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
         \dtl@donext
      \else
         \edef\dtl@donext{%
           \noexpand\dtlcompare
             {\noexpand#1}%
             {\expandonce\dtl@restA}%
             {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
         \dtl@donext
      \fi
    \else
      \ifnum\dtl@codeB=-1\relax
         \edef\dtl@donext{%
           \noexpand\dtlcompare
             {\noexpand#1}%
```

```
                {\expandonce\dtl@firstA\expandonce\dtl@restA}%
                {\expandonce\dtl@restB}}%
             \dtl@donext
          \else
            \ifnum\dtl@codeA<\dtl@codeB
              #1=-1\relax
            \else
              \ifnum\dtl@codeA>\dtl@codeB
                #1=1\relax
              \else
                 \ifdefempty{\dtl@restA}%
                 {%
                   \ifdefempty{\dtl@restB}%
                   {%
                     #1=0\relax
                   }%
                   {%
                     #1=-1\relax
                   }%
                 }%
                 {%
                   \ifdefempty{\dtl@restB}%
                   {%
                     #1=1\relax
                   }%
                   {%
                     \protected@edef\dtl@donext{%
                       \noexpand\dtlcompare
                         {\noexpand#1}{\dtl@restA}{\dtl@restB}}%
                     \dtl@donext
                   }%
                 }%
              \fi
            \fi
          \fi
        \fi
      }%
    }%
    {%
      \edef\dtl@donext{%
        \noexpand\dtlcompare
          {\noexpand#1}%
          {\expandonce\dtl@firstA\expandonce\dtl@restA}%
          {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
      \dtl@donext
    }%
  }%
  {%
    \edef\dtl@donext{%
      \noexpand\dtlcompare
```

```
            {\noexpand#1}%
            {\expandonce\dtl@firstA\expandonce\dtl@restA}%
            {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
        \dtl@donext
      }%
    }%
  }%
}
\def\dtl@if@two@octets#1#2\dtl@end@if@two@octets#3#4{%
  \ifbool{@dtl@utf8}
  {%
    \ifx\UTFviii@two@octets#1\relax
     #3%
    \else
     #4%
    \fi
  }%
  {%
    #4%
  }%
}
\def\dtl@getfirst@UTFviii#1#2#3\end@dtl@getfirst@UTFviii{%
  \def\dtl@first{#1#2}%
  \ifx\@nil#3\relax
    \def\dtl@rest{}%
  \else
    \expandafter\def\expandafter\dtl@rest\expandafter{\@dtl@firsttonil#3}%
  \fi
}
\def\@dtl@firsttonil#1\@nil{#1}
\def\dtl@getfirst#1#2\end@dtl@getfirst{%
  \def\dtl@first{#1}%
  \ifdefempty{\dtl@first}%
  {%
    \def\dtl@rest{#2}%
  }%
  {%
    \ifbool{@dtl@utf8}
    {%
      \expandafter\dtl@if@two@octets#1#2\relax\dtl@end@if@two@octets
      {%
        \dtl@getfirst@UTFviii#1#2\@nil\end@dtl@getfirst@UTFviii
      }%
      {%
        \dtl@ifsingle{#1}{\def\dtl@rest{#2}}{\dtl@getfirst#1#2\end@dtl@getfirst}%
      }%
    }%
    {%
      \dtl@ifsingle{#1}{\def\dtl@rest{#2}}{\dtl@getfirst#1#2\end@dtl@getfirst}%
    }%
```

```
  }%
}%
\newcount\dtl@codeA
\newcount\dtl@codeB
\newcommand*{\dtl@setcharcode}[2]{%
  \ifstrempty{#1}%
  {%
    #2=-1\relax
  }%
  {%
    \ifx\@dtl@wordbreak#1\relax
     #2=`\ \relax
   \else
     \ifcat\noexpand#1\relax
       #2=0\relax
     \else
       \expandafter\dtl@if@two@octets#1\relax\relax\dtl@end@if@two@octets
       {%
        \dtlsetUTFviiicharcode{#1}{#2}%
       }%
       {%
         \dtlsetcharcode{#1}{#2}%
       }%
     \fi
   \fi
  }%
}
\newcommand*{\dtlsetcharcode}[2]{#2=`#1\relax}
\newcommand*{\dtlsetlccharcode}[2]{#2=\lccode`#1\relax}
\newcommand*\dtlsetUTFviiicharcode[2]{\dtlsetdefaultUTFviiicharcode{#1}{#2}}
\newcommand*\dtlsetUTFviiilccharcode[2]{\dtlsetdefaultUTFviiilccharcode{#1}{#2}}
\newcommand*\dtlsetdefaultUTFviiicharcode[2]{%
 \ifboolexpr
 {
      test {\ifstrequal{#1}{À}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Ã}}
   or test {\ifstrequal{#1}{Ä}}
 }%
 {%
   #2=`A\relax
 }%
 {%
   \ifstrequal{#1}{Ç}%
   {%
     #2=`C\relax
   }%
   {%
     \ifboolexpr
```

```
{
     test {\ifstrequal{#1}{È}}
  or test {\ifstrequal{#1}{É}}
  or test {\ifstrequal{#1}{Ê}}
  or test {\ifstrequal{#1}{Ë}}
}%
{%
  #2=`E\relax
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{Ì}}
    or test {\ifstrequal{#1}{Í}}
    or test {\ifstrequal{#1}{Î}}
    or test {\ifstrequal{#1}{Ï}}
  }%
  {%
    #2=`I\relax
  }%
  {%
    \ifstrequal{#1}{Ñ}%
    {%
      #2=`N\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{Ò}}
        or test {\ifstrequal{#1}{Ó}}
        or test {\ifstrequal{#1}{Ô}}
        or test {\ifstrequal{#1}{Õ}}
        or test {\ifstrequal{#1}{Ö}}
      }%
      {%
        #2=`O\relax
      }%
      {%
        \ifboolexpr
        {
             test {\ifstrequal{#1}{Ù}}
          or test {\ifstrequal{#1}{Ú}}
          or test {\ifstrequal{#1}{Û}}
          or test {\ifstrequal{#1}{Ü}}
        }%
        {%
          #2=`U\relax
        }%
        {%
          \ifstrequal{#1}{Ý}%
```

```
{%
  #2=`Y\relax
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{à}}
    or test {\ifstrequal{#1}{á}}
    or test {\ifstrequal{#1}{á}}
    or test {\ifstrequal{#1}{ã}}
    or test {\ifstrequal{#1}{ä}}
  }%
  {%
    #2=`a\relax
  }%
  {%
    \ifstrequal{#1}{ç}%
    {%
      #2=`c\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{è}}
        or test {\ifstrequal{#1}{é}}
        or test {\ifstrequal{#1}{ê}}
        or test {\ifstrequal{#1}{ë}}
      }%
      {%
        #2=`e\relax
      }%
      {%
        \ifboolexpr
        {
             test {\ifstrequal{#1}{ì}}
          or test {\ifstrequal{#1}{í}}
          or test {\ifstrequal{#1}{î}}
          or test {\ifstrequal{#1}{ï}}
        }%
        {%
          #2=`i\relax
        }%
        {%
          \ifstrequal{#1}{ñ}%
          {%
            #2=`n\relax
          }%
          {%
            \ifboolexpr
            {
```

```
                                   test {\ifstrequal{#1}{ò}}
                                 or test {\ifstrequal{#1}{ó}}
                                 or test {\ifstrequal{#1}{ô}}
                                 or test {\ifstrequal{#1}{õ}}
                                 or test {\ifstrequal{#1}{ö}}
                               }%
                               {%
                                 #2=`o\relax
                               }%
                               {%
                                 \ifboolexpr
                                 {
                                      test {\ifstrequal{#1}{ù}}
                                   or test {\ifstrequal{#1}{ú}}
                                   or test {\ifstrequal{#1}{û}}
                                   or test {\ifstrequal{#1}{ü}}
                                 }%
                                 {%
                                   #2=`u\relax
                                 }%
                                 {%
                                   \ifstrequal{#1}{ý}%
                                   {%
                                     #2=`y\relax
                                   }%
                                   {%
                                     #2=64\relax
                                   }%
                                 }%
                               }%
                             }%
                           }%
                         }%
                       }%
                     }%
                   }%
                 }%
               }%
             }%
           }%
         }%
       }%
     }%
   }%
  }%
 }
 \newcommand*\dtlsetdefaultUTFviiilccharcode[2]{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{à}}
    or test {\ifstrequal{#1}{á}}
    or test {\ifstrequal{#1}{á}}
```

```
  or test {\ifstrequal{#1}{ã}}
  or test {\ifstrequal{#1}{ä}}
  or test {\ifstrequal{#1}{À}}
  or test {\ifstrequal{#1}{Á}}
  or test {\ifstrequal{#1}{Á}}
  or test {\ifstrequal{#1}{Ã}}
  or test {\ifstrequal{#1}{Ä}}
}%
{%
  #2=`a\relax
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{ç}}
    or test {\ifstrequal{#1}{Ç}}
  }
  {%
    #2=`c\relax
  }%
  {%
    \ifboolexpr
    {
         test {\ifstrequal{#1}{è}}
      or test {\ifstrequal{#1}{é}}
      or test {\ifstrequal{#1}{ê}}
      or test {\ifstrequal{#1}{ë}}
      or test {\ifstrequal{#1}{È}}
      or test {\ifstrequal{#1}{É}}
      or test {\ifstrequal{#1}{Ê}}
      or test {\ifstrequal{#1}{Ë}}
    }%
    {%
      #2=`e\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{ì}}
        or test {\ifstrequal{#1}{í}}
        or test {\ifstrequal{#1}{î}}
        or test {\ifstrequal{#1}{ï}}
        or test {\ifstrequal{#1}{Ì}}
        or test {\ifstrequal{#1}{Í}}
        or test {\ifstrequal{#1}{Î}}
        or test {\ifstrequal{#1}{Ï}}
      }%
      {%
        #2=`i\relax
      }%
```

```
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{ñ}}
    or test {\ifstrequal{#1}{Ñ}}
  }
  {%
    #2=`n\relax
  }%
  {%
    \ifboolexpr
    {
         test {\ifstrequal{#1}{ò}}
      or test {\ifstrequal{#1}{ó}}
      or test {\ifstrequal{#1}{ô}}
      or test {\ifstrequal{#1}{õ}}
      or test {\ifstrequal{#1}{ö}}
      or test {\ifstrequal{#1}{Ò}}
      or test {\ifstrequal{#1}{Ó}}
      or test {\ifstrequal{#1}{Ô}}
      or test {\ifstrequal{#1}{Õ}}
      or test {\ifstrequal{#1}{Ö}}
    }%
    {%
      #2=`o\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{ù}}
        or test {\ifstrequal{#1}{ú}}
        or test {\ifstrequal{#1}{û}}
        or test {\ifstrequal{#1}{ü}}
        or test {\ifstrequal{#1}{Ù}}
        or test {\ifstrequal{#1}{Ú}}
        or test {\ifstrequal{#1}{Û}}
        or test {\ifstrequal{#1}{Ü}}
      }%
      {%
        #2=`u\relax
      }%
      {%
        \ifboolexpr
        {
             test {\ifstrequal{#1}{ý}}
          or test {\ifstrequal{#1}{Ý}}
        }%
        {%
          #2=`y\relax
        }%
```

```
              {%
                #2=96\relax
              }%
            }%
          }%
        }%
      }%
    }%
  }%
}
\newcommand*{\dtl@setlccharcode}[2]{%
  \ifstrempty{#1}%
  {%
   #2=-1\relax
  }%
  {%
   \ifx#1\@dtl@wordbreak\relax
     #2=`\ \relax
   \else
     \ifcat\noexpand#1\relax%
       #2=0\relax
     \else
       \expandafter\dtl@if@two@octets#1\relax\relax\dtl@end@if@two@octets
       {%
        \dtlsetUTFviiilccharcode{#1}{#2}%
       }%
       {%
         \dtlsetlccharcode{#1}{#2}%
       }%
       \ifnum#2=0\relax
         #2=`#1\relax
       \fi
     \fi
   \fi
  }%
}
\newcommand*{\dtlicompare}[3]{%
  \dtl@subnobrsp{#2}{\@dtl@argA}%
  \dtl@subnobrsp{#3}{\@dtl@argB}%
  \ifdefempty{\@dtl@argA}%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      #1=0\relax
    }%
    {%
      #1=-1\relax
    }%
  }%
```

```
{%
  \ifdefempty{\@dtl@argB}%
  {%
    #1=1\relax
  }%
  {%
    \dtl@setwordbreaksnohyphens{\@dtl@argA}{\@dtl@wordbreak}%
    \let\@dtl@argA\dtl@string
    \dtl@setwordbreaksnohyphens{\@dtl@argB}{\@dtl@wordbreak}%
    \let\@dtl@argB\dtl@string
    \expandafter\dtl@getfirst\@dtl@argA\end@dtl@getfirst
    \let\dtl@firstA=\dtl@first
    \let\dtl@restA=\dtl@rest
    \expandafter\dtl@getfirst\@dtl@argB\end@dtl@getfirst
    \let\dtl@firstB=\dtl@first
    \let\dtl@restB=\dtl@rest
    \expandafter\dtl@ifsingleorUTFviii\expandafter{\dtl@firstA}%
    {%
      \expandafter\dtl@ifsingleorUTFviii\expandafter{\dtl@firstB}%
      {%
        \expandafter\dtl@setlccharcode\expandafter{\dtl@firstA}{\dtl@codeA}%
        \expandafter\dtl@setlccharcode\expandafter{\dtl@firstB}{\dtl@codeB}%
        \let\dtl@donextcompare\@firstofone
        \ifdtlcompareskipcs
          \ifnum\dtl@codeA=0\relax
            \ifnum\dtl@codeB=0\relax
               \edef\dtl@donext{%
                 \noexpand\dtlicompare
                 {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
               \dtl@donext
               \let\dtl@donextcompare\@gobble
            \else
               \edef\dtl@donext{%
                 \noexpand\dtlicompare
                 {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\@dtl@argB}}%
               \dtl@donext
               \let\dtl@donextcompare\@gobble
            \fi
          \else
            \ifnum\dtl@codeB=0\relax
               \edef\dtl@donext{%
                 \noexpand\dtlicompare
                 {\noexpand#1}{\expandonce\@dtl@argA}{\expandonce\dtl@restB}}%
               \dtl@donext
               \let\dtl@donextcompare\@gobble
            \fi
          \fi
        \fi
        \dtl@donextcompare
        {%
```

```
\ifnum\dtl@codeA=-1\relax
  \ifnum\dtl@codeB=-1\relax
     \edef\dtl@donext{%
       \noexpand\dtlicompare{\noexpand#1}%
       {\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
     \dtl@donext
  \else
     \edef\dtl@donext{%
       \noexpand\dtlicompare
         {\noexpand#1}%
         {\expandonce\dtl@restA}%
         {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
     \dtl@donext
  \fi
\else
  \ifnum\dtl@codeB=-1\relax
     \edef\dtl@donext{%
       \noexpand\dtlicompare
        {\noexpand#1}%
        {\expandonce\dtl@firstA\expandonce\dtl@restA}%
        {\expandonce\dtl@restB}}%
     \dtl@donext
  \else
    \ifnum\dtl@codeA<\dtl@codeB
     #1=-1\relax
    \else
      \ifnum\dtl@codeA>\dtl@codeB
        #1=1\relax
      \else
         \ifdefempty{\dtl@restA}%
         {%
           \ifdefempty{\dtl@restB}%
           {%
             #1=0\relax
           }%
           {%
             #1=-1\relax
           }%
         }%
         {%
           \ifdefempty{\dtl@restB}%
           {%
             #1=1\relax
           }%
           {%
             \edef\dtl@donext{%
               \noexpand\dtlicompare
                 {\noexpand#1}%
                 {\expandonce\dtl@restA}%
                 {\expandonce\dtl@restB}}%
```

```
                        \dtl@donext
                      }%
                    }%
                  \fi
                \fi
              \fi
            \fi
          }%
        }%
        {%
          \edef\dtl@donext{%
            \noexpand\dtlicompare
              {\noexpand#1}%
              {\expandonce\dtl@firstA\expandonce\dtl@restA}%
              {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
          \dtl@donext
        }%
      }%
      {%
        \edef\dtl@donext{%
          \noexpand\dtlicompare
            {\noexpand#1}%
            {\expandonce\dtl@firstA\expandonce\dtl@restA}%
            {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
        \dtl@donext
      }%
    }%
  }%
}
\newcommand*{\dtlwordindexcompare}[3]{%
  \@dtldictcompare{#1}{#2}{#3}{\@dtl@wordbreak}%
}
\newcommand*{\dtlletterindexcompare}[3]{%
  \@dtldictcompare{#1}{#2}{#3}{}%
}
\newcommand*{\@dtldictcompare}[4]{%
  \dtl@subnobrsp{#2}{\@dtl@argA}%
  \dtl@subnobrsp{#3}{\@dtl@argB}%
  \ifdefempty{\@dtl@argA}%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      #1=0\relax
    }%
    {%
      #1=-1\relax
    }%
  }%
  {%
    \ifdefempty{\@dtl@argB}%
```

```
{%
  #1=1\relax
}%
{%
  \expandafter\DTLsplitstring\expandafter
    {\@dtl@argA}{\datatoolpersoncomma}{\@dtl@beforepart}{\@dtl@afterpart}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \expandafter\DTLsplitstring\expandafter
      {\@dtl@argA}{\datatoolplacecomma}{\@dtl@beforepart}{\@dtl@afterpart}%
    \ifdefempty{\@dtl@replaced}%
    {%
      \expandafter\DTLsplitstring\expandafter
        {\@dtl@argA}{\datatoolsubjectcomma}{\@dtl@beforepart}{\@dtl@afterpart}%
      \ifdefempty{\@dtl@replaced}%
      {%
        \expandafter\DTLsplitstring\expandafter
          {\@dtl@argA}{\datatoolparenstart}{\@dtl@beforepart}{\@dtl@afterpart}%
        \ifdefempty{\@dtl@replaced}%
        {%
          \def\@dtl@A@comma{0}%
          \let\@dtl@A@before\@dtl@argA
          \def\@dtl@A@after{}%
        }%
        {%
          \let\@dtl@A@comma\@dtl@replaced
          \let\@dtl@A@before\@dtl@beforepart
          \let\@dtl@A@after\@dtl@afterpart
        }%
      }%
      {%
        \let\@dtl@A@comma\@dtl@replaced
        \let\@dtl@A@before\@dtl@beforepart
        \let\@dtl@A@after\@dtl@afterpart
      }%
    }%
    {%
      \let\@dtl@A@comma\@dtl@replaced
      \let\@dtl@A@before\@dtl@beforepart
      \let\@dtl@A@after\@dtl@afterpart
    }%
  }%
  {%
    \let\@dtl@A@comma\@dtl@replaced
    \let\@dtl@A@before\@dtl@beforepart
    \let\@dtl@A@after\@dtl@afterpart
  }%
  \expandafter\DTLsplitstring\expandafter
    {\@dtl@argB}{\datatoolpersoncomma}{\@dtl@beforepart}{\@dtl@afterpart}%
  \ifdefempty{\@dtl@replaced}%
```

```
{%
  \expandafter\DTLsplitstring\expandafter
    {\@dtl@argB}{\datatoolplacecomma}{\@dtl@beforepart}{\@dtl@afterpart}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \expandafter\DTLsplitstring\expandafter
      {\@dtl@argB}{\datatoolsubjectcomma}{\@dtl@beforepart}{\@dtl@afterpart}%
    \ifdefempty{\@dtl@replaced}%
    {%
      \expandafter\DTLsplitstring\expandafter
        {\@dtl@argB}{\datatoolparenstart}{\@dtl@beforepart}{\@dtl@afterpart}%
      \ifdefempty{\@dtl@replaced}%
      {%
        \def\@dtl@B@comma{0}%
        \let\@dtl@B@before\@dtl@argB
        \def\@dtl@B@after{}%
      }%
      {%
        \let\@dtl@B@comma\@dtl@replaced
        \let\@dtl@B@before\@dtl@beforepart
        \let\@dtl@B@after\@dtl@afterpart
      }%
    }%
    {%
      \let\@dtl@B@comma\@dtl@replaced
      \let\@dtl@B@before\@dtl@beforepart
      \let\@dtl@B@after\@dtl@afterpart
    }%
  }%
  {%
    \let\@dtl@B@comma\@dtl@replaced
    \let\@dtl@B@before\@dtl@beforepart
    \let\@dtl@B@after\@dtl@afterpart
  }%
}%
{%
  \let\@dtl@B@comma\@dtl@replaced
  \let\@dtl@B@before\@dtl@beforepart
  \let\@dtl@B@after\@dtl@afterpart
}%
\expandafter\dtl@ifcasechargroup\@dtl@A@before\dtl@end@ifcasechargroup
 {\def\@dtl@A@chargroup{2}}%
 {\def\@dtl@A@chargroup{1}}%
 {\def\@dtl@A@chargroup{0}}%
\expandafter\dtl@ifcasechargroup\@dtl@B@before\dtl@end@ifcasechargroup
 {\def\@dtl@B@chargroup{2}}%
 {\def\@dtl@B@chargroup{1}}%
 {\def\@dtl@B@chargroup{0}}%
\ifnum\@dtl@A@chargroup<\@dtl@B@chargroup
  #1=-1\relax
```

```
\else
  \ifnum\@dtl@A@chargroup>\@dtl@B@chargroup
    #1=1\relax
  \else
    \ifcase\@dtl@A@chargroup
      \edef\dtl@donext{%
        \noexpand\dtlcompare
          {\noexpand#1}%
          {\expandonce\@dtl@A@before}%
          {\expandonce\@dtl@B@before}}%
      \dtl@donext
    \or
      \ifnum\@dtl@A@before<\@dtl@B@before\relax
        #1=-1\relax
      \else
        \ifnum\@dtl@A@before>\@dtl@B@before\relax
          #1=1\relax
        \else
          #1=0\relax
        \fi
      \fi
    \or
      \@dtlwordindexcompare{#1}{\@dtl@A@before}{\@dtl@B@before}
        {\dtlicomparewords}{#4}%
      \ifnum#1=0\relax
        \let\@org@dtl@person@comma\datatoolpersoncomma
        \let\@org@dtl@place@comma\datatoolplacecomma
        \let\@org@dtl@subject@comma\datatoolsubjectcomma
        \let\@org@dtl@paren@start\datatoolparenstart
        \def\datatoolpersoncomma{3}%
        \def\datatoolplacecomma{2}%
        \def\datatoolsubjectcomma{1}%
        \def\datatoolparenstart{-1}%
        \ifnum\@dtl@A@comma>\@dtl@B@comma\relax
          #1=-1\relax
        \else
          \ifnum\@dtl@A@comma<\@dtl@B@comma\relax
            #1=1\relax
          \else
           \@dtlwordindexcompare{#1}{\@dtl@B@before}{\@dtl@A@before}
             {\dtlicomparewords}{#4}%
            \ifnum#1=0\relax
              \@dtlwordindexcompare{#1}{\@dtl@A@after}{\@dtl@B@after}
                {\dtlicomparewords}{#4}%
            \fi
          \fi
        \fi
        \let\datatoolpersoncomma\@org@dtl@person@comma
        \let\datatoolplacecomma\@org@dtl@place@comma
        \let\datatoolsubjectcomma\@org@dtl@subject@comma
```

```
              \let\datatoolparenstart\@org@dtl@paren@start
            \fi
          \fi
        \fi
      \fi
    }%
  }%
}%
\newcommand*{\datatoolpersoncomma}{,\space}
\newcommand*{\datatoolplacecomma}{,\space}
\newcommand*{\datatoolsubjectcomma}{,\space}
\newcommand*{\datatoolparenstart}{\space}
\newcommand*{\@dtlwordindexcompare}[5]{%
  \dtl@setwordbreaks{#2}{#5}%
  \let#2\dtl@string
  \dtl@setwordbreaks{#3}{}%
  \let#3\dtl@string
  \edef\@dtl@do@compare{%
    \noexpand#4{\noexpand#1}%
      {\expandonce#2}{\expandonce#3}%
  }%
  \@dtl@do@compare
}
\newcommand*{\@dtl@dict@compare}[4]{%
  \ifdefempty{#2}%
  {%
    \ifdefempty{#3}%
    {%
      #1=0\relax
    }%
    {%
      #1=-1\relax
    }%
  }%
  {%
    \ifdefempty{#3}%
    {%
      #1=1\relax
    }%
    {%
      \expandafter\dtl@grabword#2\@dtl@endgrabword\dtl@A@first\dtl@A@remain
      \expandafter\dtl@grabword#3\@dtl@endgrabword\dtl@B@first\dtl@B@remain
      \edef\@dtl@do@compare{%
        \noexpand#4{\noexpand#1}%
          {\expandonce\dtl@A@first}{\expandonce\dtl@B@first}%
      }%
      \@dtl@do@compare
      \ifnum#1=0\relax
        \@dtl@dict@compare{#1}{\dtl@A@remain}{\dtl@B@remain}{#4}%
      \fi
```

```
    }%
  }%
}
\def\dtl@grabword#1\@dtl@wordbreak#2\@dtl@endgrabword#3#4{%
  \def#3{#1}%
  \def#4{#2}%
}
\newcommand{\dtlicomparewords}[3]{%
  \dtlicompare{#1}{#2}{#3}%
}
\newcommand{\dtlcomparewords}[3]{%
  \dtlcompare{#1}{#2}{#3}%
}
\newcommand*{\dtl@setwordbreaks}[2]{%
  \expandafter\dtl@subnobrsp\expandafter{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{#2}%
  \DTLsubstituteall{\dtl@string}{\ }{#2}%
  \DTLsubstituteall{\dtl@string}{\space}{#2}%
  \DTLsubstituteall{\dtl@string}{-}{#2}%
 \toks@{#2}%
 \edef\dtl@do@setwordbreaks{%
   \noexpand\@dtl@setwordbreaks{\the\toks@}\expandonce\dtl@string\space\noexpand\@nil}%
 \def\dtl@string{}%
 \dtl@do@setwordbreaks
}
\def\@dtl@setwordbreaks#1#2 #3{%
  \def\dtl@tmp{#3}%
  \ifx\@nnil\dtl@tmp
    \let\@dtl@setwordbreaks@next\@gobbletwo
    \appto\dtl@string{#2}%
  \else
    \let\@dtl@setwordbreaks@next\@dtl@setwordbreaks
    \appto\dtl@string{#2#1}%
  \fi
  \@dtl@setwordbreaks@next{#1}#3%
}
\newcommand*{\dtl@setwordbreaksnohyphens}[2]{%
  \expandafter\dtl@subnobrsp\expandafter{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{#2}%
  \DTLsubstituteall{\dtl@string}{\ }{#2}%
  \DTLsubstituteall{\dtl@string}{\space}{#2}%
 \toks@{#2}%
 \edef\dtl@do@setwordbreaks{%
   \noexpand\@dtl@setwordbreaks{\the\toks@}\expandonce\dtl@string\space\noexpand\@nil}%
 \def\dtl@string{}%
 \dtl@do@setwordbreaks
}
\newcommand*{\@dtl@wordbreak}{ }
\def\dtl@ifcasechargroup#1#2\dtl@end@ifcasechargroup#3#4#5{%
 \expandafter\dtl@if@two@octets#1#2\relax\relax\dtl@end@if@two@octets
```

```
 {%
   \dtl@getfirst@UTFviii#1#2\@nil\end@dtl@getfirst@UTFviii
   \expandafter\dtlsetUTFviiilccharcode\expandafter{\dtl@first}{\count@}%
   \ifnum\count@<`a\relax #5\else#3\fi
 }%
 {%
  \dtlifcasechargroup{#1}%
   {#3}%
   {%
     \DTLifint{#1#2}
     {%
       #4%
     }%
     {%
       #3%
     }%
   }%
   {#5}%
 }%
}
\newcommand*{\dtlifcasechargroup}[4]{%
  \count@=`#1\relax
  \dtlifintclosedbetween{\number\count@}{48}{57}%
  {%
    #3%
  }%
  {%
    \dtlifintclosedbetween{\number\count@}{97}{122}%
    {%
      #2%
    }%
    {%
      \dtlifintclosedbetween{\number\count@}{65}{90}%
      {%
        #2%
      }%
      {%
        #4%
      }%
    }%
  }%
}
\newcommand*{\dtlparsewords}[2]{%
  \dtl@subnobrsp{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{ }%
  \DTLsubstituteall{\dtl@string}{\ }{ }%
  \DTLsubstituteall{\dtl@string}{\space}{ }%
  \DTLsubstituteall{\dtl@string}{-}{ }%
  \let\dtl@parsewordshandler#2\relax
  \edef\dtl@donext{%
```

```
     \noexpand\@dtl@parse@words\expandonce\dtl@string\space\noexpand\@nil}%
  \dtl@donext
}
\def\@dtl@parse@words#1 #2{%
  \def\dtl@tmp{#2}%
  \ifx\@nnil\dtl@tmp
   \let\parse@wordsnext=\@gobble
  \else
   \let\parse@wordsnext=\@dtl@parse@words
  \fi
  \dlt@parsewordshandler{#1}%
  \parse@wordsnext#2%
}
\newcommand*{\DTLifstringlt}{\@ifstar\@sDTLifstringlt\@DTLifstringlt}
\newcommand*{\@DTLifstringlt}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount<0\relax
    #3%
  \else
    #4%
  \fi
}
\newcommand*{\@sDTLifstringlt}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount<0\relax
    #3%
  \else
    #4%
  \fi
}
\newcommand*{\DTLiflt}{\@ifstar\@sDTLiflt\@DTLiflt}
\newcommand*{\@DTLiflt}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumlt{#1}{#2}{#3}{#4}%
  \else
   \@DTLifstringlt{#1}{#2}{#3}{#4}%
  \fi
}
\newcommand*{\@sDTLiflt}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumlt{#1}{#2}{#3}{#4}%
  \else
   \@sDTLifstringlt{#1}{#2}{#3}{#4}%
  \fi
```

```
}
\newcommand*{\DTLifnumgt}[4]{%
  \DTLconverttodecimal{#1}{\@dtl@numi}%
  \DTLconverttodecimal{#2}{\@dtl@numii}%
  \dtlifnumgt{\@dtl@numi}{\@dtl@numii}%
  {%
    #3%
  }%
  {%
    #4%
  }%
}
\newcommand*{\DTLifstringgt}{\@ifstar\@sDTLifstringgt\@DTLifstringgt}
\newcommand*{\@DTLifstringgt}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount>0\relax
    #3%
  \else
    #4%
  \fi
}
\newcommand*{\@sDTLifstringgt}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount>0\relax
    #3%
  \else
    #4%
  \fi
}
\newcommand*{\DTLifgt}{\@ifstar\@sDTLifgt\@DTLifgt}
\newcommand*{\@DTLifgt}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumgt{#1}{#2}{#3}{#4}%
  \else
   \@DTLifstringgt{#1}{#2}{#3}{#4}%
  \fi
}
\newcommand*{\@sDTLifgt}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumgt{#1}{#2}{#3}{#4}%
  \else
   \@sDTLifstringgt{#1}{#2}{#3}{#4}%
  \fi
}
```

```
\newcommand*{\DTLifnumeq}[4]{%
  \DTLconverttodecimal{#1}{\@dtl@numi}%
  \DTLconverttodecimal{#2}{\@dtl@numii}%
  \dtlifnumeq{\@dtl@numi}{\@dtl@numii}%
  {%
    #3%
  }%
  {%
    #4%
  }%
}
\newcommand*{\DTLifstringeq}{\@ifstar\@sDTLifstringeq\@DTLifstringeq}
\newcommand*{\@DTLifstringeq}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount=0\relax
    #3%
  \else
    #4%
  \fi
}
\newcommand*{\@sDTLifstringeq}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount=0\relax
    #3%
  \else
    #4%
  \fi
}
\newcommand*{\DTLifeq}{\@ifstar\@sDTLifeq\@DTLifeq}
\newcommand*{\@DTLifeq}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumeq{#1}{#2}{#3}{#4}%
  \else
   \@DTLifstringeq{#1}{#2}{#3}{#4}%
  \fi
}
\newcommand*{\@sDTLifeq}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumeq{#1}{#2}{#3}{#4}%
  \else
   \@sDTLifstringeq{#1}{#2}{#3}{#4}%
  \fi
}
\newcommand*{\DTLifSubString}[4]{%
```

```
  \protected@edef\@dtl@dotestifsubstring{\noexpand\dtl@testifsubstring
  {#1}{#2}}%
  \@dtl@dotestifsubstring
  \if@dtl@condition
   #3%
  \else
   #4%
  \fi
}
\newcommand*{\dtl@testifsubstring}[2]{%
  \dtl@subnobrsp{#1}{\@dtl@argA}%
  \dtl@subnobrsp{#2}{\@dtl@argB}%
  \dtl@setwordbreaksnohyphens{\@dtl@argA}{\@dtl@wordbreak}%
  \let\@dtl@argA\dtl@string
  \dtl@setwordbreaksnohyphens{\@dtl@argB}{\@dtl@wordbreak}%
  \let\@dtl@argB\dtl@string
  \edef\dtl@donext{%
    \noexpand\@dtl@testifsubstring{\expandonce\@dtl@argA}{\expandonce\@dtl@argB}}%
  \dtl@donext
}
\newcommand*{\@dtl@testifsubstring}[2]{%
  \def\@dtl@subs@argA{#1}%
  \def\@dtl@subs@argB{#2}%
  \ifdefempty{\@dtl@subs@argB}%
  {%
    \@dtl@conditiontrue
  }%
  {%
    \ifdefempty{\@dtl@subs@argA}%
    {%
      \@dtl@conditionfalse
    }%
    {%
      \@dtl@teststartswith{#1}{#2}%
      \if@dtl@condition
      \else
        \dtl@getfirst#1\end@dtl@getfirst
        \expandafter\dtl@ifsingle\expandafter{\dtl@first}%
        {%
          \expandafter\@dtl@testifsubstring\expandafter{\dtl@rest}{#2}%
        }%
        {%
          \protected@edef\@dtl@donext{\noexpand\@dtl@testifsubstring
             {\expandonce\dtl@first\expandonce\dtl@rest}{\expandonce\@dtl@subs@argB}}%
          \@dtl@donext
        }%
      \fi
    }%
  }%
}
```

```
\newcommand*{\DTLifStartsWith}[4]{%
  \@dtl@conditionfalse
  \protected@edef\@dtl@tmp{\noexpand\dtl@teststartswith{#1}{#2}}%
  \@dtl@tmp
  \if@dtl@condition
   #3%
  \else
   #4%
  \fi
}
\newcommand*{\dtl@teststartswith}[2]{%
  \dtl@subnobrsp{#1}{\@dtl@argA}%
  \dtl@subnobrsp{#2}{\@dtl@argB}%
  \dtl@setwordbreaksnohyphens{\@dtl@argA}{\@dtl@wordbreak}%
  \let\@dtl@argA\dtl@string
  \dtl@setwordbreaksnohyphens{\@dtl@argB}{\@dtl@wordbreak}%
  \let\@dtl@argB\dtl@string
  \edef\dtl@donext{%
    \noexpand\@dtl@teststartswith{\expandonce\@dtl@argA}{\expandonce\@dtl@argB}}%
  \dtl@donext
}

\newcommand*{\@dtl@teststartswith}[2]{%
  \def\@dtl@argA{#1}%
  \def\@dtl@argB{#2}%
  \ifdefempty{\@dtl@argA}%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      \@dtl@conditiontrue
    }%
    {%
      \@dtl@conditionfalse
    }%
  }%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      \@dtl@conditiontrue
    }%
    {%
      \expandafter\dtl@getfirst\@dtl@argA\end@dtl@getfirst
      \let\dtl@firstA=\dtl@first
      \let\dtl@restA=\dtl@rest
      \expandafter\dtl@getfirst\@dtl@argB\end@dtl@getfirst
      \let\dtl@firstB=\dtl@first
      \let\dtl@restB=\dtl@rest
      \expandafter\dtl@ifsingle\expandafter{\dtl@firstA}%
      {%
        \expandafter\dtl@ifsingle\expandafter{\dtl@firstB}%
```

```
        {%
          \expandafter\dtl@setcharcode\expandafter{\dtl@firstA}{\dtl@codeA}%
          \expandafter\dtl@setcharcode\expandafter{\dtl@firstB}{\dtl@codeB}%
          \ifnum\dtl@codeA=-1\relax
            \ifnum\dtl@codeB=-1\relax
               \protected@edef\dtl@donext{%
                 \noexpand\@dtl@teststartswith{\expandonce\dtl@restA}{\expandonce\dtl@re
               \dtl@donext
            \else
               \protected@edef\dtl@donext{%
                 \noexpand\@dtl@teststartswith
                   {\expandonce\dtl@restA}{\expandonce\dtl@firstB\expandonce\dtl@restB}}
               \dtl@donext
            \fi
          \else
            \ifnum\dtl@codeB=-1\relax
               \protected@edef\dtl@donext{%
                 \noexpand\@dtl@teststartswith
                   {\expandonce\dtl@firstA\expandonce\dtl@restA}{\expandonce\dtl@restB}}
               \dtl@donext
            \else
              \ifnum\dtl@codeA=\dtl@codeB
                \protected@edef\dtl@donext{%
                  \noexpand\@dtl@teststartswith{\expandonce\dtl@restA}{\expandonce\dtl@r
                \dtl@donext
              \else
                \@dtl@conditionfalse
              \fi
            \fi
          \fi
        }%
        {%
          \protected@edef\dtl@donext{%
            \noexpand\@dtl@teststartswith
              {\expandonce\dtl@firstA\expandonce\dtl@restA}%
              {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
          \dtl@donext
        }%
      }%
      {%
        \protected@edef\dtl@donext{%
          \noexpand\@dtl@teststartswith
            {\expandonce\dtl@firstA\expandonce\dtl@restA}%
            {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
      }%
    }%
  }%
}
\newcommand*{\DTLifnumclosedbetween}[5]{%
 \DTLconverttodecimal{#1}{\@dtl@numi}%
```

```
 \DTLconverttodecimal{#2}{\@dtl@numii}%
 \DTLconverttodecimal{#3}{\@dtl@numiii}%
 \DTLifFPclosedbetween{\@dtl@numi}{\@dtl@numii}{\@dtl@numiii}{#4}{#5}%
}
\newcommand*{\DTLifstringclosedbetween}{%
  \@ifstar\@sDTLifstringclosedbetween\@DTLifstringclosedbetween
}
\newcommand*{\@DTLifstringclosedbetween}[5]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \let\@dtl@dovalue\relax
  \ifnum\@dtl@tmpcount<0\relax
    \def\@dtl@dovalue{#5}%
  \fi
  \ifx\@dtl@dovalue\relax
    \protected@edef\@dtl@tmpcmp{%
      \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#3}}%
    \@dtl@tmpcmp
    \ifnum\@dtl@tmpcount>0\relax
      \def\@dtl@dovalue{#5}%
    \else
      \def\@dtl@dovalue{#4}%
    \fi
  \fi
  \@dtl@dovalue
}
\newcommand*{\@sDTLifstringclosedbetween}[5]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \let\@dtl@dovalue\relax
  \ifnum\@dtl@tmpcount<0\relax
    \def\@dtl@dovalue{#5}%
  \fi
  \ifx\@dtl@dovalue\relax
    \protected@edef\@dtl@tmpcmp{%
      \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#3}}%
    \@dtl@tmpcmp
    \ifnum\@dtl@tmpcount>0\relax
      \def\@dtl@dovalue{#5}%
    \else
      \def\@dtl@dovalue{#4}%
    \fi
  \fi
  \@dtl@dovalue
}
\newcommand*{\DTLifclosedbetween}{%
  \@ifstar\@sDTLifclosedbetween\@DTLifclosedbetween
}
```

```
\newcommand*{\@DTLifclosedbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \dtl@ifsingle{#1}{%
      \edef\@dtl@tmp{#1}}{%
      \def\@dtl@tmp{#1}}%
    \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
    \ifnum\@dtl@datatype>0\relax
      \DTLifnumclosedbetween{#1}{#2}{#3}{#4}{#5}%
    \else
      \@DTLifstringclosedbetween{#1}{#2}{#3}{#4}{#5}%
    \fi
  \else
    \@DTLifstringclosedbetween{#1}{#2}{#3}{#4}{#5}%
  \fi
}
\newcommand*{\@sDTLifclosedbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \dtl@ifsingle{#1}{%
      \edef\@dtl@tmp{#1}}{%
      \def\@dtl@tmp{#1}}%
    \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
    \ifnum\@dtl@datatype>0\relax
      \DTLifnumclosedbetween{#1}{#2}{#3}{#4}{#5}%
    \else
      \@sDTLifstringclosedbetween{#1}{#2}{#3}{#4}{#5}%
    \fi
  \else
    \@sDTLifstringclosedbetween{#1}{#2}{#3}{#4}{#5}%
  \fi
}
\newcommand*{\DTLifnumopenbetween}[5]{%
 \DTLconverttodecimal{#1}{\@dtl@numi}%
 \DTLconverttodecimal{#2}{\@dtl@numii}%
 \DTLconverttodecimal{#3}{\@dtl@numiii}%
 \DTLifFPopenbetween{\@dtl@numi}{\@dtl@numii}{\@dtl@numiii}{#4}{#5}%
}
\newcommand*{\DTLifstringopenbetween}{%
  \@ifstar\@sDTLifstringopenbetween\@DTLifstringopenbetween
}
\newcommand*{\@DTLifstringopenbetween}[5]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \let\@dtl@dovalue\relax
  \ifnum\@dtl@tmpcount>0\relax
  \else
    \def\@dtl@dovalue{#5}%
  \fi
```

```
  \ifx\@dtl@dovalue\relax
    \protected@edef\@dtl@tmpcmp{%
      \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#3}}%
    \@dtl@tmpcmp
    \ifnum\@dtl@tmpcount<0\relax
      \def\@dtl@dovalue{#4}%
    \else
      \def\@dtl@dovalue{#5}%
    \fi
  \fi
  \@dtl@dovalue
}
\newcommand*{\@sDTLifstringopenbetween}[5]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \let\@dtl@dovalue\relax
  \ifnum\@dtl@tmpcount>0\relax
  \else
    \def\@dtl@dovalue{#5}%
  \fi
  \ifx\@dtl@dovalue\relax
    \protected@edef\@dtl@tmpcmp{%
      \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#3}}%
    \@dtl@tmpcmp
    \ifnum\@dtl@tmpcount<0\relax
      \def\@dtl@dovalue{#4}%
    \else
      \def\@dtl@dovalue{#5}%
    \fi
  \fi
  \@dtl@dovalue
}
\newcommand*{\DTLifopenbetween}{%
  \@ifstar\@sDTLifopenbetween\@DTLifopenbetween
}
\newcommand*{\@DTLifopenbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \dtl@ifsingle{#1}{%
      \edef\@dtl@tmp{#1}}{%
      \def\@dtl@tmp{#1}}%
    \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
    \ifnum\@dtl@datatype>0\relax
      \DTLifnumopenbetween{#1}{#2}{#3}{#4}{#5}%
    \else
      \@DTLifstringopenbetween{#1}{#2}{#3}{#4}{#5}%
    \fi
  \else
    \@DTLifstringopenbetween{#1}{#2}{#3}{#4}{#5}%
```

```
  \fi
}
\newcommand*{\@sDTLifopenbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \dtl@ifsingle{#1}{%
      \edef\@dtl@tmp{#1}}{%
      \def\@dtl@tmp{#1}}%
    \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
    \ifnum\@dtl@datatype>0\relax
      \DTLifnumopenbetween{#1}{#2}{#3}{#4}{#5}%
    \else
      \@sDTLifstringopenbetween{#1}{#2}{#3}{#4}{#5}%
    \fi
  \else
    \@sDTLifstringopenbetween{#1}{#2}{#3}{#4}{#5}%
  \fi
}
\let\DTLifFPopenbetween\dtlifnumopenbetween
\let\DTLifFPclosedbetween\dtlifnumclosedbetween
\newcommand*{\dtl@testlt}[2]{%
  \DTLiflt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLislt}[2]{%
  \TE@throw\noexpand\dtl@testlt{#1}{#2}\noexpand\if@dtl@condition
}
\newcommand*{\dtl@testiclt}[2]{%
  \@sDTLiflt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisilt}[2]{%
  \TE@throw\noexpand\dtl@testiclt{#1}{#2}\noexpand\if@dtl@condition
}
\newcommand*{\dtl@testgt}[2]{%
  \DTLifgt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisgt}[2]{%
  \TE@throw\noexpand\dtl@testgt{#1}{#2}\noexpand\if@dtl@condition
}
\newcommand*{\dtl@testicgt}[2]{%
  \@sDTLifgt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisigt}[2]{%
  \TE@throw\noexpand\dtl@testicgt{#1}{#2}\noexpand\if@dtl@condition
}
\newcommand*{\dtl@testeq}[2]{%
  \DTLifeq{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLiseq}[2]{%
  \TE@throw\noexpand\dtl@testeq{#1}{#2}\noexpand\if@dtl@condition
}
```

```
\newcommand*{\dtl@testiceq}[2]{%
  \@sDTLifeq{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisieq}[2]{%
  \TE@throw\noexpand\dtl@testiceq{#1}{#2}\noexpand\if@dtl@condition
}
\newcommand*{\DTLisSubString}[2]{%
  \TE@throw\noexpand\dtl@testifsubstring{#1}{#2}%
  \noexpand\if@dtl@condition
}
\newcommand*{\DTLisPrefix}[2]{%
  \TE@throw\noexpand\dtl@teststartswith{#1}{#2}%
  \noexpand\if@dtl@condition
}
\newcommand*{\DTLisinlist}[2]{%
  \TE@throw\noexpand\dtl@testinlist{#1}{#2}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testinlist}[2]{%
  \DTLifinlist{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\dtl@testnumclosedbetween}[3]{%
  \DTLifnumclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisnumclosedbetween}[3]{%
  \TE@throw\noexpand\dtl@testnumclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testnumopenbetween}[3]{%
  \DTLifnumopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisnumopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testnumopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testclosedbetween}[3]{%
  \DTLifclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisclosedbetween}[3]{%
  \TE@throw\noexpand\dtl@testclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testiclosedbetween}[3]{%
  \@sDTLifclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisiclosedbetween}[3]{%
```

```
  \TE@throw\noexpand\dtl@testiclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testopenbetween}[3]{%
  \DTLifopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testiopenbetween}[3]{%
  \@sDTLifopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisiopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testiopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
\let\DTLisFPclosedbetween\DTLisnumclosedbetween
\newcommand*{\dtl@testFPopenbetween}[3]{%
  \DTLifFPopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisFPopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testFPopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testFPislt}[2]{%
  \dtlifnumlt{#1}{#2}%
  {%
    \@dtl@conditiontrue
  }%
  {%
    \@dtl@conditionfalse
  }%
}
\newcommand*{\DTLisFPlt}[2]{%
  \TE@throw\noexpand\dtl@testFPislt{#1}{#2}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testFPisgt}[2]{%
  \dtlifnumgt{#1}{#2}%
  {%
    \@dtl@conditiontrue
  }%
  {%
   \@dtl@conditionfalse
  }%
}
```

```
\newcommand*{\DTLisFPgt}[2]{%
  \TE@throw\noexpand\dtl@testFPisgt{#1}{#2}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testFPiseq}[2]{%
 \dtlifnumeq{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
  \@dtl@conditionfalse
 }%
}
\newcommand*{\DTLisFPeq}[2]{%
  \TE@throw\noexpand\dtl@testFPiseq{#1}{#2}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testFPislteq}[2]{%
 \dtlifnumlt{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
   \@dtl@conditionfalse
 }%
 \if@dtl@condition
 \else
  \dtl@testFPiseq{#1}{#2}%
 \fi
}
\newcommand*{\DTLisFPlteq}[2]{%
 \TE@throw\noexpand\dtl@testFPislteq{#1}{#2}%
 \noexpand\if@dtl@condition
}
\newcommand*{\dtl@testFPisgteq}[2]{%
 \dtlifnumgt{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
   \@dtl@conditionfalse
 }%
 \if@dtl@condition
 \else
  \dtl@testFPiseq{#1}{#2}%
 \fi
}
\newcommand*{\DTLisFPgteq}[2]{%
  \TE@throw\noexpand\dtl@testFPisgteq{#1}{#2}%
  \noexpand\if@dtl@condition}
```

```
\newcommand*{\dtl@teststring}[1]{%
  \DTLifstring{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
\newcommand*{\DTLisstring}[1]{%
  \TE@throw\noexpand\dtl@teststring{#1}\noexpand\if@dtl@condition}
\newcommand*{\dtl@testnumerical}[1]{%
  \DTLifnumerical{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
\newcommand*{\DTLisnumerical}[1]{%
  \TE@throw\noexpand\dtl@testnumerical{#1}\noexpand\if@dtl@condition}
\newcommand*{\dtl@testint}[1]{%
  \DTLifint{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
\newcommand*{\DTLisint}[1]{%
  \TE@throw\noexpand\dtl@testint{#1}\noexpand\if@dtl@condition}
\newcommand*{\dtl@testreal}[1]{%
  \DTLifreal{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
\newcommand*{\DTLisreal}[1]{%
  \TE@throw\noexpand\dtl@testreal{#1}\noexpand\if@dtl@condition}
\newcommand*{\dtl@testcurrency}[1]{%
  \DTLifcurrency{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
\newcommand*{\DTLiscurrency}[1]{%
  \TE@throw\noexpand\dtl@testcurrency{#1}\noexpand\if@dtl@condition}
\newcommand*{\dtl@testcurrencyunit}[2]{%
  \DTLifcurrencyunit{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
\newcommand*{\DTLiscurrencyunit}[2]{%
  \TE@throw\noexpand\dtl@testcurrencyunit{#1}{#2}%
  \noexpand\if@dtl@condition
}
\newcommand*{\dtlbreak}{%
  \PackageError{datatool}{Can't break out of anything}{}%
}
\long\def\dtlforint#1=#2\to#3\step#4\do#5{%
  \let\@dtl@orgbreak\dtlbreak
  \def\@dtl@endloophook{}%
  \def\dtlbreak{\def\@dtl@endloophook{#1=#3}}%
  #1=#2\relax
  \ifnum#4<0\relax
    \whiledo{\(#1>#3\)\TE@or\(#1=#3\)}%
    {%
      #5%
      \@dtl@endloophook
      \advance#1 by #4\relax
    }%
  \else
    \whiledo{\(#1<#3\)\TE@or\(#1=#3\)}%
    {%
      #5%
      \@dtl@endloophook
      \advance#1 by #4\relax
    }%
  \fi
```

```
  \let\dtlbreak\@dtl@orgbreak
}
\newcount\@dtl@foreach@level
\long\def\dtlgforint#1=#2\to#3\step#4\do#5{%
  \global#1=#2\relax
  \global\advance\@dtl@foreach@level by 1\relax
  \expandafter\global\expandafter
    \let\csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
    \relax
  \expandafter\global\expandafter
    \let\csname @dtl@break@\the\@dtl@foreach@level\endcsname
    \dtlbreak
  \gdef\dtlbreak{\expandafter
     \gdef\csname @dtl@endhook@\the\@dtl@foreach@level\endcsname{%
       #1=#3}}%
  \ifnum#4<0\relax
    \whiledo{\(#1>#3\)\TE@or\(#1=#3\)}%
    {%
      #5%
      \csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
      \global\advance#1 by #4\relax
    }%
  \else
    \whiledo{\(#1<#3\)\TE@or\(#1=#3\)}%
    {%
      #5%
      \csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
      \global\advance#1 by #4\relax
    }%
  \fi
  \expandafter\global\expandafter\let\expandafter\dtlbreak
    \csname @dtl@break@\the\@dtl@foreach@level\endcsname
  \global\advance\@dtl@foreach@level by -1\relax
}
\newenvironment{dtlenvgforint}[1]%
{%
  \def\@dtlenvgforint@arg{#1}%
  \long@collect@body\@do@dtlenvgforint
}%
{}
\newcommand{\@do@dtlenvgforint}[1]{%
  \expandafter\dtlgforint\@dtlenvgforint@arg\do{#1}%
}
```

30.17 Rollback v2.32 (datatool-fp-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datatool-fp}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{xkeyval}
\RequirePackage{fp}
```

```
\RequirePackage{datatool-base}[=v2.32]
\define@choicekey{datatool-fp}{verbose}[\val\nr]{true,false}[true]{%
  \ifcase\nr\relax
    \FPmessagestrue
  \or
    \FPmessagesfalse
  \fi
}
\let\ifFPmessages\ifdtlverbose
\ProcessOptionsX
\providecommand*{\@dtl@mathprocessor}{fp}
\newcommand*{\dtlifnumeq}[4]{%
  \FPifeq{#1}{#2}%
    #3%
  \else
    #4%
  \fi
}
\let\ifdtlverbose\ifFPmessages
\newcommand*{\dtlifnumlt}[4]{%
  \FPiflt{#1}{#2}%
   #3%
  \else
   #4%
  \fi
}
\newcommand*{\dtlifnumgt}[4]{%
  \FPifgt{#1}{#2}%
   #3%
  \else
   #4%
  \fi
}
\newcommand*{\dtlifnumopenbetween}[5]{%
  \let\@dtl@dovalue\relax
  \dtlifnumgt{#1}{#2}%
  {}%
  {%
    \def\@dtl@dovalue{#5}%
  }%
  \dtlifnumlt{#1}{#3}%
  {%
    \ifx\@dtl@dovalue\relax
      \def\@dtl@dovalue{#4}%
    \fi
  }%
  {%
    \def\@dtl@dovalue{#5}%
  }%
  \@dtl@dovalue
```

```
}
\newcommand*{\dtlifnumclosedbetween}[5]{%
  \let\@dtl@dovalue\relax
  \dtlifnumgt{#1}{#2}%
  {}%
  {%
    \dtlifnumeq{#1}{#2}%
    {%
      \def\@dtl@dovalue{#4}%
    }%
    {%
      \def\@dtl@dovalue{#5}%
    }%
  }%
  \dtlifnumlt{#1}{#3}%
  {%
    \ifx\@dtl@dovalue\relax
      \def\@dtl@dovalue{#4}%
    \fi
  }%
  {%
    \dtlifnumeq{#1}{#3}%
    {%
      \def\@dtl@dovalue{#4}%
    }%
    {%
      \def\@dtl@dovalue{#5}%
    }%
  }%
  \@dtl@dovalue
}
\newcommand*{\dtladd}[3]{%
  \FPadd{#1}{#2}{#3}%
}
\newcommand*{\dtlsub}[3]{%
  \FPsub{#1}{#2}{#3}%
}
\newcommand*{\dtlmul}[3]{%
  \FPmul{#1}{#2}{#3}%
}
\newcommand*{\dtldiv}[3]{%
  \FPdiv{#1}{#2}{#3}%
}
\newcommand*{\dtlroot}[2]{%
  \FProot{#1}{#2}%
}
\newcommand*{\dtlround}[3]{%
  \FPround{#1}{#2}{#3}%
}
\newcommand*{\dtltrunc}[3]{%
```

```
  \FPtrunc{#1}{#2}{#3}%
}
\newcommand*{\dtlclip}[2]{%
  \FPclip{#1}{#2}%
}
\newcommand*{\dtlmin}[3]{%
  \FPmin{#1}{#2}{#3}%
}
\newcommand*{\dtlmax}[3]{%
  \FPmax{#1}{#2}{#3}%
}
\newcommand*{\dtlabs}[2]{%
  \FPabs{#1}{#2}%
}
\newcommand*{\dtlneg}[2]{%
  \FPneg{#1}{#2}%
}
```

30.18 Rollback v2.32 (datatool-pgfmath-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datatool-pgfmath}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{xkeyval}
\RequirePackage{pgfrcs,pgfkeys,pgfmath}
\ProcessOptionsX
\providecommand*{\@dtl@mathprocessor}{pgfmath}
\newcommand*{\dtlifnumeq}[4]{%
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
  \pgfmathifthenelse{\number0#1==\number0#2}%
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
\RequirePackage{datatool-base}[=v2.32]
\newcommand*{\dtlifnumlt}[4]{%
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
  \pgfmathifthenelse{\number0#1 < \number0#2}%
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
\newcommand*{\dtlifnumgt}[4]{%
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
  \pgfmathifthenelse{\number0#1 > \number0#2}%
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
\newcommand*{\dtlifnumopenbetween}[5]{%
  \def\@dtl@truepart{#4}%
```

```
  \def\@dtl@falsepart{#5}%
  \pgfmathifthenelse
    {(\number0#2 < \number0#1) && (\number0#1 < \number0#3)}%
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
\newcommand*{\dtlifnumclosedbetween}[5]{%
  \def\@dtl@truepart{#4}%
  \def\@dtl@falsepart{#5}%
  \pgfmathifthenelse
    {(\number0#2 <= \number0#1) && (\number0#1 <= \number0#3)}
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
\newcommand*{\dtladd}[3]{%
  \pgfmathadd{#2}{#3}%
  \let#1\pgfmathresult
}
\newcommand*{\dtlsub}[3]{%
  \pgfmathsubtract{#2}{#3}%
  \let#1\pgfmathresult
}
\newcommand*{\dtlmul}[3]{%
  \pgfmathmultiply{#2}{#3}%
  \let#1\pgfmathresult
}
\newcommand*{\dtldiv}[3]{%
  \pgfmathdivide{#2}{#3}%
  \let#1\pgfmathresult
}
\newcommand*{\dtlroot}[2]{%
  \pgfmathsqrt{#2}%
  \let#1\pgfmathresult
}
\newcommand*{\dtlround}[3]{%
  \ifnum#3=0\relax
    \pgfmathparse{int(round(#2))}%
    \let#1\pgfmathresult
  \else
    \pgfmathparse{int(10^#3)}%
    \let\dtl@tmpshift\pgfmathresult
    \pgfmathparse{int(floor(#2))}%
    \let\dtl@int@round\pgfmathresult
    \pgfmathparse{int(round((#2-\dtl@int@round) * \dtl@tmpshift))}%
    \@dtl@tmpcount=0\relax
    \expandafter\@dtl@countdigits\pgfmathresult.\relax
    \advance\@dtl@tmpcount by -#3\relax
    \def\@dtl@intpart{}%
    \def\@dtl@fracpart{}%
    \expandafter\@dtl@gatherintfrac\pgfmathresult\relax
```

```
    \edef\@dtl@intpart{\number\numexpr\dtl@int@round
      +\number0\@dtl@intpart}%
    \edef#1{\@dtl@intpart.\@dtl@fracpart}%
  \fi
}
\newcommand*{\@dtl@gatherintfrac}[1]{%
  \ifx\relax#1\relax
  \else
    \advance\@dtl@tmpcount by -1\relax
    \ifnum\@dtl@tmpcount<0\relax
      \edef\@dtl@fracpart{\@dtl@fracpart#1}%
    \else
      \edef\@dtl@intpart{\@dtl@intpart#1}%
    \fi
    \expandafter\@dtl@gatherintfrac
  \fi
}
\newcommand*{\dtltrunc}[3]{%
  \ifnum#3=0\relax
    \pgfmathparse{int(floor(#2))}%
    \let#1\pgfmathresult
  \else
    \pgfmathparse{int(10^#3)}%
    \let\dtl@tmpshift\pgfmathresult
    \pgfmathparse{int(floor(#2))}%
    \let\dtl@int@trunc\pgfmathresult
    \pgfmathparse{int(floor((#2-\dtl@int@trunc) * \dtl@tmpshift))}%
    \@dtl@tmpcount=0\relax
    \expandafter\@dtl@countdigits\pgfmathresult.\relax
    \advance\@dtl@tmpcount by -#3\relax
    \def\@dtl@intpart{}%
    \def\@dtl@fracpart{}%
    \expandafter\@dtl@gatherintfrac\pgfmathresult\relax
    \edef\@dtl@intpart{\number\numexpr\dtl@int@trunc
      +\number0\@dtl@intpart}%
    \edef#1{\@dtl@intpart.\@dtl@fracpart}%
  \fi
}
\newcommand*{\dtlclip}[2]{%
 \edef#1{#2}%
}
\newcommand*{\dtlmin}[3]{%
  \pgfmathmin{#2}{#3}%
  \let#1\pgfmathresult
}
\newcommand*{\dtlmax}[3]{%
  \pgfmathmax{#2}{#3}%
  \let#1\pgfmathresult
}
\newcommand*{\dtlabs}[2]{%
```

```
  \pgfmathabs{#2}%
  \let#1\pgfmathresult
}
\newcommand*{\dtlneg}[2]{%
  \pgfmathmul{-1}{#2}%
  \let#1\pgfmathresult
}
```

30.19 Rollback v2.32 (person-2019-09-27.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{person}[2019/09/27 v2.32 (NLCT)]
\RequirePackage{ifthen}
\RequirePackage{datatool}[=v2.32]
\newcounter{people}
\newcounter{person}
\newcommand*{\@people@list}{,}
\newcommand*{\@get@firstperson}[1]{%
  \expandafter\@@get@firstperson\@people@list,\@nil{#1}}
  \def\@@get@firstperson,#1,#2\@nil#3{%
  \def#3{#1}%
}
\newcommand*{\malelabels}{male,Male,MALE,M,m}
\newcommand*{\addmalelabel}[1]{%
  \expandafter\@dtl@toksA\expandafter{\malelabels}%
  \expandafter\@dtl@toksB\expandafter{#1}%
  \edef\malelabels{\the\@dtl@toksA,\the\@dtl@toksB}%
}
\newcommand*{\addfemalelabel}[1]{%
  \expandafter\@dtl@toksA\expandafter{\femalelabels}%
  \expandafter\@dtl@toksB\expandafter{#1}%
  \edef\femalelabels{\the\@dtl@toksA,\the\@dtl@toksB}%
}
\newcommand*{\femalelabels}{female,Female,FEMALE,F,f}
\newcommand{\ifmalelabel}[3]{%
   \expandafter\DTLifinlist\expandafter{#1}{\malelabels}{#2}{#3}%
}
\newcommand{\iffemalelabel}[3]{%
   \expandafter\DTLifinlist\expandafter{#1}{\femalelabels}{#2}{#3}%
}
\newcommand*{\newperson}[4][anon]{%
  \@ifundefined{person@#1@name}%
  {%
    \ifmalelabel{#4}%
    {%
      \expandafter\gdef\csname person@#1@gender\endcsname{male}%
    }%
    {%
      \iffemalelabel{#4}%
      {%
```

```
        \expandafter\gdef\csname person@#1@gender\endcsname{female}%
      }%
      {%
         \PackageError{person}{Unknown gender `#4' for person
         `#1'}{Allowed gender labels are: \malelabels\space or
         \femalelabels}%
         \@namedef{person@#1@gender}{other}%
      }%
    }%
    \expandafter
      \protected@xdef\csname person@#1@fullname\endcsname{#2}%
    \expandafter
      \protected@xdef\csname person@#1@name\endcsname{#3}%
    \protected@xdef\@people@list{\@people@list#1,}%
    \stepcounter{people}%
  }%
  {%
    \PackageError{person}{Person `#1' has already been defined}{}%
  }%
}
\newcommand*{\removeperson}[1][anon]{%
  \edef\@person@label{#1}%
  \expandafter\@removeperson\expandafter{\@person@label}%
}
\newcommand*{\@removeperson}[1]{%
   \ifpersonexists{#1}%
   {%
     \def\@remove@person##1,#1,##2\@nil{%
       \def\@prsn@pre{##1}\def\@prsn@post{##2}}%
     \expandafter\@remove@person\@people@list\@nil
     \xdef\@people@list{\@prsn@pre,\@prsn@post}%
     \addtocounter{people}{-1}%
     \expandafter\global\expandafter
       \let\csname person@#1@name\endcsname\undefined
     \expandafter\global\expandafter
       \let\csname person@#1@fullname\endcsname\undefined
     \expandafter\global\expandafter
       \let\csname person@#1@gender\endcsname\undefined
   }%
   {%
     \PackageError{person}{Can't remove person `#1': no such
     person}{}%
   }%
}
\newcommand*{\removepeople}[1]{%
  \@for\@thisperson:=#1\do{%
    \ifx\@thisperson\@empty
    \else
      \expandafter\removeperson\expandafter[\@thisperson]%
    \fi
```

```
  }%
}
\newcommand*{\removeallpeople}{%
  \@for\@thisperson:=\@people@list\do{%
    \expandafter\global\expandafter
      \let\csname person@\@thisperson @name\endcsname\undefined
    \expandafter\global\expandafter
      \let\csname person@\@thisperson @fullname\endcsname\undefined
    \expandafter\global\expandafter
      \let\csname person@\@thisperson @gender\endcsname\undefined
  }%
  \setcounter{people}{0}%
  \gdef\@people@list{,}%
}
\newcommand{\ifpersonexists}[3]{%
  \@ifundefined{person@#1@name}{#3}{#2}%
}
\newcommand{\ifmale}[3]{%
   \ifpersonexists{#1}%
   {%
      \edef\@gender{\csname person@#1@gender\endcsname}%
      \ifx\@gender\@male@label
        #2%
      \else
        #3%
      \fi
   }%
   {%
      \PackageError{person}{Person `#1' doesn't exist.}{}%
   }%
}
\def\@male@label{male}
\newcommand{\ifallmale}[3][\@people@list]{%
  \@for\@thisperson:=#1\do{%
    \ifpersonexists{\@thisperson}%
    {%
      \edef\@gender{\csname person@\@thisperson @gender\endcsname}%
      \ifx\@gender\@male@label
      \else
        \@endfortrue
      \fi
    }%
    {%
      \PackageError{person}{Person `#1' doesn't exist.}{}%
    }%
  }%
  \if@endfor
    #3%
  \else
    #2%
```

```
  \fi
}
\newcommand{\iffemale}[3]{%
   \ifpersonexists{#1}%
   {%
      \edef\@gender{\csname person@#1@gender\endcsname}%
      \ifx\@gender\@female@label
        #2%
      \else
        #3%
      \fi
   }%
   {%
      \PackageError{person}{Person `#1' doesn't exist.}{}%
   }%
}
\def\@female@label{female}
\newcommand{\ifallfemale}[3][\@people@list]{%
  \@for\@thisperson:=#1\do{%
    \edef\@gender{\csname person@\@thisperson @gender\endcsname}%
    \ifx\@gender\@female@label
    \else
      \@endfortrue
    \fi
  }%
  \if@endfor
    #3%
  \else
    #2%
  \fi
}
\def\foreachperson(#1,#2,#3,#4)#5{%
  \ifx#5\in
    \def\@do@foreachperson{\@foreachperson(#1,#2,#3,#4)#5}%
  \else
    \def\@do@foreachperson{%
      \@foreachperson(#1,#2,#3,#4)\in\@people@list#5}%
  \fi
  \@do@foreachperson
}
\long\def\@foreachperson(#1,#2,#3,#4)\in#5\do#6{%
  \@for#4:=#5\do{%
    \ifx#4\@empty
    \else
       \ifpersonexists{#4}%
       {%
         \expandafter
           \let\expandafter#1\csname person@#4@name\endcsname
         \expandafter
           \let\expandafter#2\csname person@#4@fullname\endcsname
```

```
          \expandafter
            \let\expandafter#3\csname person@#4@gender\endcsname
          \ifx#3\@male@label
            \let#3\malename
          \else
            \ifx#3\@female@label
              \let#3\femalename
            \fi
          \fi
          #6%
        }%
        {%
          \PackageError{person}{Person `#4' doesn't exist}{}%
        }%
      \fi
  }%
}
\newcommand*{\malepronoun}{he}
\newcommand*{\femalepronoun}{she}
\newcommand*{\pluralpronoun}{they}
\newcommand*{\maleobjpronoun}{him}
\newcommand*{\femaleobjpronoun}{her}
\newcommand*{\pluralobjpronoun}{them}
\newcommand*{\malepossadj}{his}
\newcommand*{\femalepossadj}{her}
\newcommand*{\pluralpossadj}{their}
\newcommand*{\maleposspronoun}{his}
\newcommand*{\femaleposspronoun}{hers}
\newcommand*{\pluralposspronoun}{theirs}
\newcommand*{\malechild}{son}
\newcommand*{\femalechild}{daughter}
\newcommand*{\pluralchild}{children}
\newcommand*{\malechildren}{sons}
\newcommand*{\femalechildren}{daughters}
\newcommand*{\maleparent}{father}
\newcommand*{\femaleparent}{mother}
\newcommand*{\pluralparent}{parents}
\newcommand*{\malesibling}{brother}
\newcommand*{\femalesibling}{sister}
\newcommand*{\pluralsibling}{siblings}
\newcommand*{\malesiblings}{brothers}
\newcommand*{\femalesiblings}{sisters}
\providecommand*{\andname}{and}
\newcommand*{\malename}{male}
\newcommand*{\femalename}{female}
\newcommand*{\personsep}{, }
\newcommand*{\personlastsep}{\space\andname\space}
\newcommand*{\twopeoplesep}{\space\andname\space}
\newcommand*{\personfullname}[1][anon]{%
  \@ifundefined{person@#1@fullname}%
```

```
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \csname person@#1@fullname\endcsname
  }%
}
\newcommand*{\peoplefullname}{%
  \setcounter{person}{1}%
  \@for\@thisperson:=\@people@list\do{%
    \ifthenelse{\equal{\@thisperson}{}}%
    {}%
    {%
      \personfullname[\@thisperson]%
      \stepcounter{person}%
      \ifnum\c@people=1\relax
      \else
        \ifnum\c@person=\c@people
          \ifnum\c@people=2\relax
            \twopeoplesep
          \else
            \personlastsep
          \fi
        \else
           \ifnum\c@person<\c@people
             \personsep
           \fi
        \fi
      \fi
    }%
  }%
}
\newcommand*{\personname}[1][anon]{%
  \@ifundefined{person@#1@name}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \csname person@#1@name\endcsname
  }%
}
\newcommand*{\peoplename}{%
  \setcounter{person}{1}%
  \@for\@thisperson:=\@people@list\do{%
    \ifthenelse{\equal{\@thisperson}{}}%
    {}%
    {%
      \personname[\@thisperson]%
      \stepcounter{person}%
      \ifnum\c@people=1\relax
```

```
      \else
        \ifnum\c@person=\c@people
          \ifnum\c@people=2\relax
            \twopeoplesep
          \else
            \personlastsep
          \fi
        \else
           \ifnum\c@person<\c@people
             \personsep
           \fi
        \fi
      \fi
    }%
  }%
}
\newcommand*{\personpronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender pronoun\endcsname
  }%
}
\newcommand*{\Personpronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter
    \MakeUppercase\csname\@gender pronoun\endcsname
  }%
}
\newcommand*{\peoplepronoun}{%
  \ifnum\c@people>1\relax
     \pluralpronoun
  \else
     \@get@firstperson{\@thisperson}%
     \personpronoun[\@thisperson]%
  \fi
}
\newcommand*{\Peoplepronoun}{%
  \ifnum\c@people>1\relax
     \expandafter\MakeUppercase\pluralpronoun
  \else
     \@get@firstperson{\@thisperson}%
```

```
     \Personpronoun[\@thisperson]%
  \fi
}
\newcommand*{\personobjpronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender objpronoun\endcsname
  }%
}
\newcommand*{\Personobjpronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter
    \MakeUppercase\csname\@gender objpronoun\endcsname
  }%
}
\newcommand*{\peopleobjpronoun}{%
  \ifnum\c@people>1\relax
     \pluralobjpronoun
  \else
     \@get@firstperson{\@thisperson}%
     \personobjpronoun[\@thisperson]%
  \fi
}
\newcommand*{\Peopleobjpronoun}{%
  \ifnum\c@people>1\relax
     \expandafter\MakeUppercase\pluralobjpronoun
  \else
     \@get@firstperson{\@thisperson}%
     \Personobjpronoun[\@thisperson]%
  \fi
}
\newcommand*{\personpossadj}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender possadj\endcsname
  }%
}
```

```
\newcommand*{\Personpossadj}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter
    \MakeUppercase\csname\@gender possadj\endcsname
  }%
}
\newcommand*{\peoplepossadj}{%
  \ifnum\c@people>1\relax
     \pluralpossadj
  \else
     \@get@firstperson{\@thisperson}%
     \personpossadj[\@thisperson]%
  \fi
}
\newcommand*{\Peoplepossadj}{%
  \ifnum\c@people>1\relax
     \expandafter\MakeUppercase\pluralpossadj
  \else
     \@get@firstperson{\@thisperson}%
     \Personpossadj[\@thisperson]%
  \fi
}
\newcommand*{\personposspronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender posspronoun\endcsname
  }%
}
\newcommand*{\Personposspronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter
    \MakeUppercase\csname\@gender posspronoun\endcsname
  }%
}
\newcommand*{\peopleposspronoun}{%
  \ifnum\c@people>1\relax
```

```
     \pluralpossproncun
  \else
     \@get@firstperson{\@thisperson}%
     \personpossproncun[\@thisperson]%
  \fi
}
\newcommand*{\Peoplepossproncun}{%
  \ifnum\c@people>1\relax
     \expandafter\MakeUppercase\pluralpossproncun
  \else
     \@get@firstperson{\@thisperson}%
     \Personpossproncun[\@thisperson]%
  \fi
}
\newcommand*{\personchild}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender child\endcsname
  }%
}
\newcommand*{\Personchild}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter\MakeUppercase
      \csname\@gender child\endcsname
  }%
}
\newcommand*{\peoplechild}{%
  \ifnum\c@people>1\relax
    \ifallmale
      {\malechildren}%
      {\ifallfemale{\femalechildren}{\pluralchild}}%
  \else
    \@get@firstperson{\@thisperson}%
    \personchild[\@thisperson]%
 \fi
}
\newcommand*{\Peoplechild}{%
  \ifnum\c@people>1\relax
    \ifallmale
      {\expandafter\MakeUppercase\malechildren}%
      {\ifallfemale
```

```
        {\expandafter\MakeUppercase\femalechildren}
        {\expandafter\MakeUppercase\pluralchild}}%
  \else
    \@get@firstperson{\@thisperson}%
    \Personchild[\@thisperson]%
 \fi
}
\newcommand*{\personparent}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender parent\endcsname
  }%
}
\newcommand*{\Personparent}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter\MakeUppercase
       \csname\@gender parent\endcsname
  }%
}
\newcommand*{\peopleparent}{%
  \ifnum\c@people>1\relax
     \pluralparent
  \else
     \@get@firstperson{\@thisperson}%
     \personparent[\@thisperson]%
  \fi
}
\newcommand*{\Peopleparent}{%
  \ifnum\c@people>1\relax
     \expandafter\MakeUppercase\pluralparent
  \else
     \@get@firstperson{\@thisperson}%
     \Personparent[\@thisperson]%
  \fi
}
\newcommand*{\personsibling}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
```

```
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender sibling\endcsname
  }%
}
\newcommand*{\Personsibling}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person `#1' has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter\MakeUppercase
      \csname\@gender sibling\endcsname
  }%
}
\newcommand*{\peoplesibling}{%
  \ifnum\c@people>1\relax
    \ifallmale
      {\malesiblings}%
      {\ifallfemale{\femalesiblings}{\pluralsibling}}%
  \else
    \@get@firstperson{\@thisperson}%
    \personsibling[\@thisperson]%
 \fi
}
\newcommand*{\persongender}[1]{%
   \ifmale{#1}{\malename}{\femalename}%
}
\newcommand*{\getpersongender}[2]{%
   \ifmale{#2}{\let#1\malename}{\let#1\femalename}%
}
\newcommand*{\getpersonname}[2]{%
   \ifpersonexists{#2}%
   {%
     \expandafter\let\expandafter#1\csname person@#2@name\endcsname
   }%
   {%
      \PackageError{person}{Person `#2' doesn't exist}{}%
   }%
}
\newcommand*{\getpersonfullname}[2]{%
   \ifpersonexists{#2}%
   {%
     \expandafter
       \let\expandafter#1\csname person@#2@fullname\endcsname
   }%
   {%
      \PackageError{person}{Person `#2' doesn't exist}{}%
   }%
}
```

Change History

1.01 – 2007 Aug 17
General: uses \@sDTLforeach instead of \DTLforeach . . . 927
\DTLaddall: removed extraneous space 201
\dtlcompare: replaces \compare (no longer using compare.tex) 233
\DTLforeach: added starred version 496
\DTLgetrowforkey: new 566
\DTLgetvalueforkey: new . . . 566
\dtlicompare: new 234
\DTLifclosedbetween: added starred version 247
\DTLifeq: added starred version . 243
\DTLifgt: added starred version . 241
\DTLiflastrow: fixed bug 512
\DTLiflt: added starred version . 240
\DTLifopenbetween: added starred version 249
\DTLifStartsWith: new 245
\DTLifstringclosedbetween: added starred version 246
\DTLifstringeq: added starred version 242
\DTLifstringgt: added starred version 241
\DTLifstringlt: added starred version 239
\DTLifstringopenbetween: added starred version 248
\DTLifSubString: new 244
\DTLinitialhyphen: new 219
\DTLinitials: now uses \DTLinitialhyphen 215
now works with unbreakable space symbol 215
\DTLisiclosedbetween: new . 255
\DTLisieq: new 252
\DTLisigt: new 251
\DTLisilt: new 251
\DTLisiopenbetween: new . . . 255
\DTLisiSubString: new 252
\DTLisPrefix: new 252
\DTLisSubString: new 252
\DTLisSuffix: new 253
\DTLmaxall: removed extraneous space 204
\DTLmeanforall: removed extraneous space 205
\DTLminall: removed extraneous space 204
\DTLplotstream: uses \@sDTLforeach instead of \DTLforeach 963
\DTLremovecurrentrow: fix bug caused by missing \fi and unrequired argument 508
\DTLsdforall: fixed bug 207
removed extraneous space 207
\DTLsort: added optional argument 586
added starred version 586
\DTLsplitstring: new 221
\DTLstoreinitials: now uses \DTLinitialhyphen 216
now works with unbreakable space symbol 216
\DTLsubstituteall: fixed bug caused when certain commands occur in the string 221
\DTLvarianceforall: fixed bug 206
removed extraneous space 206
1.01 – 2007/08/22
\DTLmultibibs: new 846
1.03 – 2009 January 27
\DTLnewbibitem: removed check if database exists 781
\DTLnewbibrow: removed check if database exists 781
\DTLplotlines: fixed error in solid line setting 935
2.0 – 2009 February 27
General: added etex as a required package 313
removed \@dtl@getidtype . 434
removed \@dtl@ifrowcontains . . 434
removed \@dtl@setidtype . 434
removed \@dtl@setkeys . . . 434

removed \dtl@getentryid . 434
removed \dtl@getentryvalue ... 434
\@DTLforeach: updated to use new database structure 501
\@DTLifdbempty: new 403
\@DTLremoverow: new 540
\@dtl@after: new 415
\@dtl@assign: updated to use new database structure 438
\@dtl@before: new 415
\@dtl@colhead: new 415
\@dtl@decrementrows: new .. 539
\@dtl@getcolumnindex: new . 406
\@dtl@getdatatype: new 409
\@dtl@getprops: new 410
\@dtl@list: new 586
\@dtl@setnull: modified to use new database structure 440
\@dtl@sortcriteria: updated to take account of new database structure 592
\@dtl@updatekeys: new 420
\@dtlgetdatatype: new 409
\@dtlifreadonly: new 502
\@sDTLforeach: updated to use new database structure 501
\@sDTLnewrow: new 404
\@sdtlgetdatatype: new 409
\dtl@compare: no longer used .. 592
\dtl@compare@: updated to use new database structure 592
\dtl@decrementrows: new ... 539
\dtl@gathervalues: updated to use new database structure ... 443
\dtl@sortdata: new 592
\dtladdalign: new 514
\DTLaddentryforrow: updated to use new database structure .. 509
\dtlaftercols: new 513
\DTLappendtorow: updated to use new database structure 502
\dtlbeforecols: new 513
\dtlbetweencols: new 513
\DTLcolumncount: new 402
\dtlcolumnindex: new 407
\dtlcolumnnum: new 404
\dtlcurrencyformat: new ... 517
\DTLcurrencytype: new 408
\dtldisplayafterhead: new . 518
\DTLdisplaydb: new 524
\dtldisplayendtab: new 518
\DTLdisplaylongdb: new 536
\dtldisplaystartrow: new .. 518
\dtldisplaystarttab: new .. 518
\DTLeverybarhook: new 865
\dtlforcolumnidx: new 492
\DTLforeachkeyinrow: updated to use new database structure .. 510
\DTLgetcolumnindex: new ... 406
\DTLgetdatatype: new 409
\dtlgetentryfromcurrentrow: new 453
\DTLgetrowforkey: update to use new database structure ... 566
\DTLgetvalueforkey: updated to use new database structure .. 566
\dtlheaderformat: new 516
\DTLiffirstrow: modified to have different definition depending on location 511
\DTLifhaskey: new 405
\DTLiflastrow: modified to have different definition depending on location 512
\DTLifoddrow: modified to have different definition depending on location 512
\dtlintformat: new 517
\DTLinttype: new 408
\DTLloaddb: added optional argument 641
removed checks to see if the database exists when adding to it 641
\DTLmaxforcolumn: new 560
\DTLmaxforkeys: added second optional argument 558
\DTLmeanforcolumn: new 547
\DTLmeanforkeys: added second optional argument 545
\DTLminforcolumn: new 557
\DTLminforkeys: added second optional argument 555
\DTLnewdb: Changed way database is stored 399
\dtlrealformat: new 517
\DTLrealtype: new 408
\DTLremovecurrentrow: updated to use new database structure 508

\DTLremoveentryfromrow: updated to use new database structure 504
\DTLremoverow: new 540
\DTLreplaceentryforrow: updated to use new database structure 506
\dtlrownum: new 404
\DTLsavedb: updated to use new database structure 624
\DTLsavetexdb: updated to use new database structure 624
\DTLsdforcolumn: new 553
\DTLsdforkeys: added second optional argument 552
\dtlshowdb: updated to use new database structure 643
\dtlshowdbkeys: updated to use new database structure 643
\dtlshowtype: updated to use new database structure 644
\DTLsort: updated to use new data structure 586
\dtlstringformat: new 516
\DTLstringtype: new 408
\DTLsumcolumn: new 543
\DTLsumforkeys: added second optional argument 541
\DTLunsettype: new 408
\DTLvarianceforcolumn: new 551
\DTLvarianceforkeys: added second optional argument 549
2.03 – 2009 November 15
\@dtl@assigncmd: modified to ignore spaces after commas . . . 440
\@dtl@storeandupdate: value can be expanded before adding to database 431
\DTLappendtorow: value expanded before storing 503
\DTLcleardb: new 399
\dtlcolumnindex: renamed \dtl@columnindex to \dtlcolumnindex 407
\DTLdeletedb: new 400
\DTLreplaceentryforrow: expand replacement entry 507
2.10 – 2012-07-18
\@dtl@construct@qlopoff: Added code to replace escaped delimiters 645
\@dtl@getkeyforcolumn: fixed bug 408
\dtl@domappings: replaced \DTLsubstitute with \DTLsubstituteall 642
\dtl@omitlines: new 599
\dtlappendentrytocurrentrow: new 454
\DTLassign: new 434
\DTLdisplaydb: added optional arg 524
\DTLdisplaylongdb: added omit option 536
\DTLifnumeq: changed \FPifeq to \dtlifnumeq 230
\DTLifnumgt: changed \FPifgt to \dtlifnumgt 230
\DTLifnumlt: changed \FPiflt to \dtlifnumlt 230
\dtlrecombineomitcurrent: new 449
\dtlremoveentryincurrentrow: new 452
\dtlreplaceentryincurrentrow: new 451
\DTLsettabseparator: changed tab character to ^^I 315
\DTLsubstituteall: added \long 221
\dtlswapentriesincurrentrow: new 453
\long@addto@envbody: new . . 269
\long@collect@@body: new . . 270
\long@push@begins: new 270
2.11 – 2012-09-25
General: remove unwanted space . . 424
\@dtl@updatekeys: remove unwanted space 420
\@dtlgetrowindex: new 462
\DTLaddcolumn: new 418
\dtldisplayvalign: new 518
\dtlgetrowforvalue: new . . . 446
\DTLgetrowindex: new 461
\dtlgetrowindex: new 462
\dtlupdateentryincurrentrow: new 455
2.12 – 2012-11-30
General: fixed bug in \DTLiflastrow 497

\dtlifnumclosedbetween: fixed bug causing premature expansion 306
\dtlifnumeq: fixed bug causing premature expansion 305
\dtlifnumgt: fixed bug causing premature expansion 305
\dtlifnumlt: fixed bug causing premature expansion 305
\dtlifnumopenbetween: fixed bug causing premature expansion 306
2.13 – 2013-01-15
General: added datagidx package . . 646
\@DTLnewrow: fixed typo in \PackageError 403
\@dtl@setheaderforindex: removed spurious space 429
\@dtlnumbernull: new 66
\@dtlstringnull: new 66
\DTLaddentryforrow: removed spurious space 509
\DTLforeachkeyinrow: changed to use \@dtlstringnull and \@dtlnumbernull 510
\DTLgcleardb: new 401
\DTLgdeletedb: new 401
\DTLgnewdb: new 401
\dtlparsewords: new 238
\DTLrawmap: removed spurious space 642
\DTLsaverawdb: new 624
\dtlsort: new 586
\ifDTLnewdbonload: new 595
2.14 – 2013-06-28
General: added \postnewtermhook 740
added calc library requirement . . 931
\@dtl@updatekeys: expand value before testing if it's empty 420
\@dtl@width: replaced \FPsub etc with \dtlsub etc 1020
\datagidx@style@gloss: removed spurious see also line . 694
\datagidxdb: new 747
\dtl@constructminorticklist: replaced \FPsub with \dtlsub etc 1031
\dtl@constructticklist: replaced \FPsub with \dtlsub etc 1027
\dtl@constructticklistwithgap: replaced \FPadd with \dtladd 1028
\dtl@constructticklistwithgapex: replaced \FPadd with \dtladd and changed third argument to minimum gap width (in data co-ordinates) 1032
\dtl@getbounds: changed \FPifgt to \dtlifnumgt 1024
\DTLgidxForeachEntry: Added optional argument when using \DTLforeach 777
\DTLpar: changed to \let 320
\ifnewtermfield: new 742
\newtermfield: new 741
\postnewtermhook: new 741
2.15 – 2013-07-10
General: added afterpage as a required package 660
\datagidx@write@usedentry: Added check for page counter . 758
\do@locrange: removed spurious space 706
\DTLaddtoplotlegend: Used \xdef instead of \edef as may be scoped. 1033
\dtlplothandlermark: new . . 964
\DTLprotectedsaverawdb: new 624
\loadgidx: new 708
2.16 – 2013-08-16
\@dtl@updatekeys: reverted to not expanding value (2.14 change causes an error with fragile commands). Fix now in \@dtl@checknumerical . 420
2.17 – 2013-08-29
\DTLfetch: new 446
\edtlgetrowforvalue: new . . 446
2.18 – 2013-09-06
\DTLpar: changed back to a robust command 320
2.19 – 2014-01-17
\@datagidx@use@entry: renamed \datagidx@use@entry to \@datagidx@use@entry and removed redundant field argument 756

\dtldisplaycr: new 519
\glsaddall: Fixed bug in database reference 766
2.20 – 2014-02-03
General: change \gdef to \xdef . 496
\@dtl@assign: Added check for empty argument 438
\DTLassignfirstmatch: new . 435
\DTLifnullorempty: new 441
\DTLloaddbtex: new 641
\xDTLassignfirstmatch: new 435
2.21 – 2014-03-08
\@gDTLforeachbibentry: new 788
\@sgDTLforeachbibentry: new 788
\datagidx@parse@location: replaced \ifstrequal with \ifdefequal 702
\dtl@g@gathervalues: new . . 444
\dtl@gathervalues: fixed bug that ignore row tok argument . . 443
\DTLforeachbibentry: fixed bug in starred version 786
\gDTLforeachbibentry: new . 787
\gDTLformatbibentry: new . . 784
2.22 – 2014-06-10
\DTLcustombibitem: new 814
\DTLformatbooktitle: new . . 839
\DTLformatthisbibentry: new 785
\DTLpcite: new 805
\ifdtlautokeys: new 595
2.23 – 2015-07-11
General: removed etex as a required package 313
\@DTLsort: bug fix: replaced \dtlicompare with \dtlcompare 586
\dtl@getfirst: changed \end to \end@dtl@getfirst . . 274
\dtl@testinlist: new 253
\DTLisinlist: new 253
\DTLmaketabspace: restores tab catcode to 10 315
\iffemale: bug fix: replaced \@thisperson with #1 . . . 1059
\ifmale: bug fix: replaced \@thisperson with #1 . . . 1058
\persongender: bug fix: replaced \ifpersonmale with \ifmale 1080
\PersonIfFemale: bug fix: replaced \@thisperson with #1 1058
2.24 – 2016-01-12
\@dtl@firsttonil: new 274
\dtl@getfirst@UTFviii: new 273
\dtl@if@two@octets: new . . . 273
\dtl@ifsingleorUTFviii: new 272
\dtldisableUTFviii: new . . . 269
\dtlenableUTFviii: new 269
\dtlsetcharcode: new 275
\dtlsetdefaultUTFviiicharcode: new 276
\dtlsetdefaultUTFviiilccharcode: new 279
\dtlsetlccharcode: new 275
\dtlsetUTFviiicharcode: new 276
\dtlsetUTFviiilccharcode: new 276
2.26 – 2016-07-20
General: removed truncation (caused minor gap between first and last segments) 921
replaced \ifthen 922
\@@dtl@set@off: replaced fp commands 929
\@@dtl@setoffr: replaced fp commands 929
\@dtl@gatherintfrac: new . . 308
\dtl@insertinto: fixed bug (incorrect inequality sign) 271
fixed bug (missing \dtl@sortresult) 271
\dtlifnumclosedbetween: added \number 306
\dtlifnumeq: added \number . 305
\dtlifnumgt: added \number . 305
\dtlifnumlt: added \number . 305
\dtlifnumopenbetween: added \number 306
\DTLpiechart: replaced fp commands 926
\dtlround: fixed bug cause by rounding 308

fixed bug caused by rounding errors . . . 308
\dtltrunc: fixed bug caused by rounding errors . . . 309
new . . . 308
2.27 – 2016-07-28
\dtl@insertinto: undone the incorrect change in v2.26 . . . 271
\dtlinsertinto: new . . . 47
\dtlsortlist: new . . . 46
\edtlinsertinto: new . . . 48
2.28 – 2017-11-10
General: renamed \toks@g... to \dtl@toks@g... . . . 50
\@dtl@formatlist@handler: new . . . 50
\@dtlformatlist: new . . . 50
\datagidxconvertchars: new 733
\datagidxextendedtoascii: new . . . 733
\DTLandname: new . . . 49
\DTLformatlist: new . . . 50
\DTLlistformatitem: new . . . 49
\DTLlistformatlastsep: new 49
\DTLlistformatoxford: new . 49
\DTLlistformatsep: new . . . 49
\ifdatagidxbalance: new . . . 707
\printterms: added paragraph break at the start . . . 775
\printterms@setupmulticol: new . . . 772
\printterms@setuptwocol: new . . . 773
\printtermsstartpar: new . . 772
\s@dtlformatlist: new . . . 50
\xdtlgetrowindex: new . . . 462
2.30 – 2018-04-16
\@dtl@construct@lopoff: removed spurious spaces . . . 644
2.31 – 2018-12-07
\@dtl@listelement@outofrange: new . . . 270
\DTLnumitemsinlist: made robust . . . 31
\ifDTLlistskipempty: new . . 23
2.32 – 2019-09-27
General: changed \ifx test to \ifblank . . . 499
removed \relax . . . 497
\datagidxstripaccents: added test for kernel version . . 734
\dtlgidx@checklocationchange: new . . . 777
\ifdtlcompareskipcs: new . . 23
3.0 – 2025-03-03
General: added datetime option . . . 12
added encoding files . . . 53
added lists option . . . 22
added non-binary check . . . 1048
added numeric option . . . 9
added thinspace, apostrophe, and underscore number group commands . . . 62
dropped xkeyval . . . 313, 779, 850, 908, 930
moved lower-level null commands from datatool to datatool-base 65
new . . . 273, 281–283, 291
removed afterpage as a required package . . . 660
removed textcase as a required package . . . 660
removed xfor as a required package . . . 660
\@@dtl@set@off: replaced with __datapie_set_offset:n . . . 929
\@@dtl@setoffr: replaced with __datapie_set_offset_range:nw . . . 929
\@@get@firstperson: removed . . . 1041
\@DTLaddcolumnwithheader: new . . . 419
\@DTLforeachbibentry: removed . . . 787
\@DTLgetrowindex: new . . . 462
\@DTLifEndsWith: new . . . 245
\@DTLifStartsWith: new . . . 245
\@DTLifSubString: new . . . 244
\@DTLreconstructdata: new . 626
\@DTLremoverow: obey global setting . . . 541
\@datagidx@db@col@id@w: removed . . . 655
\@dtl@assign: updated to LaTeX3 438
\@dtl@assigncmd: removed . . . 440
\@dtl@assigncmdnoop: removed . . . 440

\@dtl@assigntmpseq: new ... 30
\@dtl@bar: changed to __databar_index_int . 876
\@dtl@barcount: removed 856
\@dtl@cite@write: new 847
\@dtl@currencies: removed .. 185
\@dtl@decimal: replaced 56
\@dtl@def@write: new 624
\@dtl@gap: replaced with \l__dataplot_gap_fp .. 932
\@dtl@get@sortdirection: removed 592
\@dtl@getsortdirection: removed 592
\@dtl@ifsingle: removed 270
\@dtl@list: removed 586
\@dtl@mathprocessor: changed default processor to l3fp or lua ... 3
\@dtl@neggap: replaced with \l__dataplot_neg_gap_fp 932
\@dtl@numbergroupchar: replaced 56
\@dtl@posgap: replaced with \l__dataplot_pos_gap_fp 932
\@dtl@rowa: removed 591
\@dtl@rowb: removed 591
\@dtl@seg: replaced with \l__datapie_segment_int 916
\@dtl@set@off: replaced with __datapie_set_offset:nw 929
\@dtl@sortcriteria: removed 592
\@dtl@start: renamed \l__databar_start_fp . 853
replaced with \l__datapie_start_tl . 917
\@dtl@storeandupdate: new . 431
\@dtl@thislabel: replaced with \l__databib_label_tl . 841
\@dtl@useresult: new 332
\@dtl@widestlabel: replaced with \g__databib_widest_label_tl 841
\@dtl@width: replaced with \l__dataplot_width_fp 933
switched to l3fp 1020
\@dtl@year: replaced with \l__databib_year_tl .. 846
\@dtlgetfirstchar: new 274
\@dtlmaxforkeys: removed ... 560
\@dtlmeanforkeys: removed .. 547
\@dtlminforkeys: removed ... 557
\@dtlnovalue: replaced with \c_nullvalue_tl 65
\@dtlnumbernull: moved from datatool to datatool-base 66
\@dtlsdforkeys: removed 553
\@dtlstringnull: moved from datatool to datatool-base 66
\@dtlsumforkeys: removed ... 543
\@dtlvarianceforkeys: removed 551
\@female@label: replaced with \c_person_female_label_tl 1041
\@foreachperson: removed .. 1062
\@gDTLforeachbibentry: removed 788
\@get@firstperson: removed 1041
\@male@label: replaced with \c_person_male_label_tl 1041
\@people@list: replaced with \l__person_people_seq 1041
\@person@datatoolsty: new 1037
\@s@DTLnewdbentry: new 433
\@sDTLforeachbibentry: removed 787
\@sDTLifEndsWith: new 245
\@sDTLifStartsWith: new ... 245
\@sDTLifSubString: new 244
\@sgDTLforeachbibentry: removed 788
macro: made robust 238
\addfemalelabel: made robust 1042
\addmalelabel: deprecated .. 1042
\andname: removed 779, 1068
\datagidx@add@term: replaced with __datagidx_add_term:n 736
\datagidx@addchild: replaced with \ 744
\datagidx@anchorcount: replaced with \g__datagidx_anchor_count_int 751

\datagidx@aux@usedentry:
new 761
\datagidx@clearlocationformat:
removed 752
\datagidx@columns: replaced
with
\l__datagidx_columns_int
. 646
\datagidx@count: removed . . . 766
\datagidx@defaultdatabase:
replaced with
\l__datagidx_default_database_tl
. 707
\datagidx@doifsymlocwidth:
replaced with
__datagidx_do_ifsymlocwidth:nnn
. 670
\datagidx@escapelocation:
removed 752
\datagidx@escapelocationformat:
removed 752
\datagidx@formatanchor:
replaced with
__datagidx_format_anchor:n
. 751
\datagidx@formatlocation:
replaced with
__datagidx_formatlocation:nn
. 699
\datagidx@formatsymdesc:
removed 649
\datagidx@getgroup: removed 776
\datagidx@getlocdo: removed 700
\datagidx@multicols:
replaced with
\l__datagidx_multicols_tl
. 667
\datagidx@postheading:
replaced with
\l__datagidx_post_heading_tl
. 667
\datagidx@setchildsort:
removed 663
\datagidx@setchildstyle:
removed 647
\datagidx@setfieldvalues:
removed 735
\datagidx@setlocation:
removed 648
\datagidx@setnamecase:
removed 647
\datagidx@setpostdesc:
removed 647
\datagidx@setprelocation:
removed 647
\datagidx@setsee: removed . . 648
\datagidx@sort: replaced
\dtlsort with
\DTLsortdata 668
replaced with
\l__datagidx_sort_tl . 668
\datagidx@sort@foreachchild:
removed 662
\datagidx@sortchildren:
removed 662
\datagidx@unsort@foreachchild:
removed 662
\datagidx@write@usedentry:
Removed check for page counter 758
\datagidxdoseealso: changed
to new document command . . . 664
\datagidxmapdata: new 677
\datatoolasciiend: new 32
\datatoolasciistart: new . . 32
\datatoolctrlboundary: new 32
\datatoolcurrencysymbolprefixfmt:
new 9
\DataToolDateFmt: new 13
\DataToolDateTimeFmt: new . 15
\datatoolparen: new 237
\datatoolSetCurrencySort:
new 193
\DataToolTimeFmt: new 14
\DataToolTimeStampFmtSep:
new 15
\DataToolTimeStampNoZoneFmt:
new 16
\DataToolTimeStampWithZoneFmt:
new 16
\DataToolTimeZoneFmt: new . 15
\do@prevlocation: removed . . 705
\dtl@angle: replaced with
\l__datapie_angle_tl . 917
\dtl@barlabel: renamed
\l__databar_barlabel_tl
. 854
\dtl@bounds: replaced with
\l__dataplot_bounds_seq
. 935

\dtl@compare: removed 592
\dtl@compare@: removed 592
\dtl@constructminorticklist: removed 1031
\dtl@constructticklist: removed 1027
\dtl@constructticklistwithgap: removed 1028
\dtl@constructticklistwithgapex: removed 1032
\dtl@constructticklistwithgapz: removed 1031
\dtl@cutawayoffset: replaced with \l__datapie_cutaway_offset_dim 909
\dtl@cutlen: replaced with \l__datapie_cut_length_tl 917
\dtl@decx: replaced with \l__dataplot_decimal_x_tl 931
\dtl@decy: replaced with \l__dataplot_decimal_y_tl 931
\dtl@dx: replaced with \l__dataplot_x_extent_fp 933
\dtl@dy: replaced with \l__dataplot_y_extent_fp 933
\dtl@endangle: replaced with \l__datapie_end_tl . . . 917
\dtl@extent: renamed \l__databar_extent_fp 853 replaced with \l__datapie_extent_tl 917
\dtl@get@yearsuffix: replaced with __databib_get_year_suffix:n 845
\dtl@getbounds: removed . . . 1024
\dtl@getfirst: deprecated . . . 274
\dtl@ifsingle: now uses LaTeX3 270
\dtl@ifsingleorUTFviii: partially rewritten to use LaTeX3 272
\dtl@init@write@wrap: new . 608
\dtl@innerlabel: renamed \l__datatool_pie_inner_label_tl 910
\dtl@innernodeopt: replaced with \l__datapie_inner_node_opt_tl 917
\dtl@inneroffset: replaced with \l__datapie_inner_offset_dim 909
\dtl@legend: replaced with \l_dataplot_legend_tl 935
\dtl@legendlabels: converted to sequence \l__dataplot_legend_labels_seq 941
\dtl@legendsetting: replaced with \l__dataplot_legend_setting_int 941
\dtl@linestyle: replaced with \l__dataplot_current_line_tl 934
\dtl@mark: replaced with \l__dataplot_current_mark_tl 933
\dtl@maxx: replaced with \l__dataplot_max_x_tl 940
\dtl@maxy: replaced with \l__dataplot_max_y_tl 940
\dtl@midangle: replaced with \l__datapie_mid_tl . . . 917
\dtl@minx: replaced with \l__dataplot_min_x_tl 939
\dtl@miny: replaced with \l__dataplot_min_y_tl 940
\dtl@multibarlabels: changed to \l__databar_multibarlabels_seq 854
\dtl@offset@x: replaced with \l__dataplot_offset_x_fp 932
\dtl@offset@y: replaced with \l__dataplot_offset_y_fp 933
\dtl@outerlabel: renamed \l__datatool_pie_outer_label_tl 910

\dtl@outernodeopt: replaced with \l__datapie_outer_node_opt_tl 917
\dtl@outeroffset: replaced with \l__datapie_outer_offset_dim 909
\dtl@piecutaways: replaced with \l__datapie_cutaways_clist 910
\dtl@reconstruct@data: new 623
\dtl@scale@x: replaced with \l__dataplot_scale_x_fp 932
\dtl@scale@y: replaced with \l__dataplot_scale_y_fp 932
\dtl@sortdata: removed 592
\dtl@SortWordCommands: new 32
\dtl@SortWordCommands@hook: new 32
\dtl@stream: replaced with \l__dataplot_stream_tl 934
\dtl@thisplotlinecolor: replaced with \l__dataplot_current_line_color_tl 934
\dtl@thisplotmarkcolor: replaced with \l__dataplot_current_mark_color_tl 933
\dtl@ticklength: replaced with \ 933
\dtl@ticlabeloffset: replaced with \l__dataplot_tic_label_offset_fp 933
\dtl@unit: renamed \l__databar_unit_fp . . 853
\dtl@upperbarlabel: changed to \l__databar_upperbarlabel_tl 854
\dtl@uppermultibarlabels: changed to \l__databar_uppermultibarlabels_seq 854
\dtl@x: replaced with \l__dataplot_x_tl 931
\dtl@xkey: replaced \dtl@xkey with \l__dataplot_x_key_tl 931
\dtl@xlabel: replaced with \l__dataplot_xlabel_tl 940
\dtl@xminorticlist: replaced with \l__dataplot_x_minor_tic_seq 934
\dtl@xticgap: replaced with \l__dataplot_xtic_gap_tl 940
\dtl@xticlabelheight: replaced with \l__dataplot_x_tic_label_height_dim 933
\dtl@xticlabels: converted to sequence \l__dataplot_xtic_labels_seq 940
\dtl@xticlist: replaced with \l__dataplot_x_tic_seq 940
\dtl@y: replaced with \l__dataplot_y_tl 931
\dtl@ykey: replaced \dtl@ykey with \l__dataplot_y_key_tl 931
\dtl@ylabel: replaced with \l__dataplot_ylabel_tl 940
\dtl@yminorticlist: replaced with \l__dataplot_y_minor_tic_seq 934
\dtl@yticgap: replaced with \l__dataplot_ytic_gap_tl 940
\dtl@yticlabels: converted to sequence \l__dataplot_ytic_labels_seq 940
\dtl@yticlabelwidth: replaced with \l__ytic_label_width_dim 933
\dtl@yticlist: replaced with \l__dataplot_y_tic_seq 940
\DTLabs: made robust 202
\DTLaction: new 320

\DTLadd: made robust 200
\DTLaddall: made robust 201
\dtladdall: new 300, 306
\DTLaddcolumn: made robust . . 418
\DTLaddcolumnwithheader: new 418
\dtladdheaderalign: new . . . 516
\DTLaddperiod: bug fix: insert period before resetting conditional 791
\DTLaddtoplotlegend: switched to \l_dataplot_legend_tl instead of \dtl@legend . . 1033
\DTLaposinitialpunc: new . . 220
\dtlappendentrytocurrentrow: made robust 454
use new value setting instead of always expanding 454
\DTLappendtorow: made robust 502
\DTLassign: made robust 434
\DTLassignfirstmatch: made robust 435
\DTLassignfromcurrentrow: new 437
\DTLassignlettergroup: new 41
\dtlbar@groupgap: changed to \l__databar_groupgap_tl 854
\dtlbar@variables: changed to \l__databar_variables_seq 853
\dtlbar@ylabel: renamed \l__databar_ylabel_tl 854
\dtlbar@yticgap: renamed to \l__databar_yticgap_tl 854
\DTLbargroupindex: new 856
\DTLbargrouplabelalign: new 857
\DTLbarindex: new 856
\DTLbarsetupperlabelalign: new 865
\DTLBarStyle: new 858
\DTLbartotalvariables: new 852
\DTLbarXneglabelalign: new 851
\DTLbarXnegupperlabelalign: new 851
\DTLbarXupperlabelalign: new 851
\DTLbibaccessedname: new . . 807
\DTLbibdatefield: new 788
\DTLbibdoi: new 806
\DTLbibdoihome: new 805
\DTLbibdoitag: new 805
\DTLbibeprints: new 806
\DTLbibfieldlet: made robust 789
\DTLbibformatdigital: made new 807
\DTLbibgetlongestlabel: new 782
\DTLbibliography: made robust 783
switched to read-only loop 783
\DTLbibliographystyle: made robust 846
\DTLbibpubmed: new 807
\DTLbibpubmedhome: new 805
\DTLbibpubmedtag: new 806
\DTLbibsetlongestlabel: new 782
\DTLbibsortencap: new 849
\DTLbibsortname: new 849
\DTLbibsortnamesep: new . . . 850
\DTLbiburl: new 807
\DTLbiburldate: new 806
\DTLcheckendsperiod: made robust 790
\DTLclearbarcolors: new . . . 860
\DTLcleardb: changed to new document command 399
\DTLclearnegbarcolors: new 861
\DTLclip: made robust 210
\dtlcolumnheader: new 516
\dtlcolumnindex: expand to 0 if undefined 407
\DTLcomputebounds: changed to new document command 562
changed to use \DTLmapdata . 562
\DTLcomputewidestbibentry: made robust 785
switched to read-only loop 785
\DTLconverttodecimal: made robust 104
\DTLcurr: new 192
\DTLcurrChar: new 186
\DTLcurrCodeOrSymOrChar: new 10
\dtlcurrdefaultfmt: new . . . 190
\DTLcurrency: new 190
\DTLCurrencyCode: new 185
\dtlcurrencygroup: new 44

\DTLCurrencySymbol: new . . . 185
\DTLcurrencytype: changed to \newcommand 408
\DTLCurrentLocaleFormatDate: new 12
\DTLCurrentLocaleFormatTime: new 13
\DTLCurrentLocaleFormatTimeStampNoZone: new 18
\DTLCurrentLocaleFormatTimeStampWithZone: new 18
\DTLCurrentLocaleFormatTimeZone: new 14
\DTLCurrentLocaleGetGroupString: new 42
\DTLCurrentLocaleGetInitialLetter: new 210
\DTLCurrentLocaleIfpmTF: new 150
\DTLCurrentLocaleParseDate: new 145
\DTLCurrentLocaleParseTime: new 146
\DTLCurrentLocaleParseTimeStamp: new 144
\DTLCurrentLocaleTimeStampFmtSep: new 15
\DTLCurrentLocaleWordHandler: new 32
\DTLcurrEUR: new 193
\dtlcurrfmtsep: new 192
\dtlcurrfmtsymsep: new 191
\dtlcurrprefixfmt: new 190
\DTLcurrStr: new 186
\dtlcurrsuffixfmt: new 191
\DTLcurrSym: new 186
\DTLcurrXBT: new 192
\DTLcurrXXX: new 192
\DTLcustombibitem: made robust 814
\DTLcustomlegend: new 938
\DTLdatatypecurrencyname: new 64
\DTLdatatypedatename: new . 64
\DTLdatatypedatetimename: new 64
\DTLdatatypedecimalname: new 64
\DTLdatatypeintegername: new 64
\DTLdatatypeinvalidname: new 64
\DTLdatatypestringname: new 64
\DTLdatatypetimename: new . 64
\DTLdatatypeunsetname: new 64
\dtldateformat: new 517
\dtldategroup: new 44
\dtldatetimeformat: new . . . 517
\dtldatetimegroup: new 44
\DTLdatumcurrency: new 78
\DTLdatumtype: new 79
\DTLdatumvalue: new 78
\dtldbcolreconstruct: new . 621
\dtldbdatumreconstruct: new 622
\dtldbheaderreconstruct: new 622
\DTLdbLog: new 642
\DTLdbNewEntry: new 434
\DTLdbNewRow: new 434
\DTLdbProvideData: new 624
\dtldbreconstructkeyindex: new 623
\dtldbrowreconstruct: new . 621
\DTLdbSetHeader: new 434
\dtldbvaluereconstruct: new 622
\DTLdecimaltocurrency: added optional argument 106
\DTLdefaultEURcurrencyfmt: new 192
\DTLDefaultLocaleWordHandler: new 31
\DTLdefcurrency: new 187
\DTLdeletedb: changed to new document command 400
\DTLdisplaybargrouplabel: new 859
\DTLdisplaydb: changed to new document command 524
\DTLdisplaydbAddBegin: new 519
\DTLdisplaydbAddEnd: new . . 520
\DTLdisplaydbAddItem: new . 533
\dtldisplaydbenv: new 519
\DTLdisplaylongdb: made robust and reimplemented 536
\DTLdisplaylongdbAddEnd: new 535
\dtldisplaylongdbenv: new . 533

\DTLdisplayTBrowidxmap: new 519
\DTLdiv: made robust 202
DTLenvforeach: changed to xparse 495
DTLenvforeach*: changed to xparse 495
DTLenvmapdata: new 471
\DTLeverybargrouphook: new 865
\DTLeveryprebarhook: new . . 865
\dtlfallbackaction: new . . . 44
\DTLfetch: made robust 446
\DTLfetchlistelement: rewritten in LaTeX3 31
\DTLfmtcurr: new 190
\DTLfmtcurrency: new 190
\dtlforcolumn: changed to document command 491
\dtlforcolumnidx: changed to document command 492
\DTLforeach: made robust 496
\DTLforeachbibentry: made robust 786
only assign \DBIBcitekey and \DBIBentrytype locally . . 786
\DTLformatarticlecrossref: made robust 804
\DTLformatbibentry: made robust 784
\DTLformatbookcrossref: made robust 801
\DTLformatbvolume: made robust 799
\DTLformatchapterpages: made robust 799
\DTLformatcrossrefeditor: made robust 797
\DTLformatdate: made robust . 804
\DTLformatforenames: made robust 796
\DTLformatincollproccrossref: made robust 802
\DTLformatinedbooktitle: made robust 803
\DTLformatjr: made robust . . . 797
\DTLformatnumberseries: made robust 800
\DTLformatpages: made robust 800
\DTLformatsurname: made robust 797
\DTLformatsurnameonly: made robust 796
\DTLformatthisbibentry: made robust 785
\DTLformatvolnumpages: made robust 798
\DTLformatvon: made robust . . 796
\DTLgabs: made robust 202
\DTLgadd: made robust 200
\DTLgaddall: made robust 201
\DTLgcleardb: changed to new document command 401
\DTLgclip: made robust 210
\DTLgdeletedb: changed to new document command 401
\DTLgdiv: made robust 202
\DTLget: new 331
\DTLgetcolumnindex: made robust 406
\DTLgetdatatype: made robust 409
\DTLgetDataTypeName: new . . 63
\dtlgetentryfromcurrentrow: made robust 453
\dtlgetentryfromrow: made robust 453
\DTLGetInitialLetter: new . 210
\DTLgetkeydata: made robust . 442
\DTLgetkeyforcolumn: made robust 407
\DTLgetlocation: made robust 458
\DTLgetnegbarcolor: new . . . 862
\dtlgetrow: made robust 444
\dtlgetrowforvalue: made robust 446
\DTLgetrowindex: changed to new document command and added starred variant 461
\dtlgetrowindex: made robust 462
\DTLgetvalue: made robust . . . 457
\dtlgidx@checklocationchange: removed 777
\DTLgidxAddLocationType: deprecated 752
dtlgidxchildlist: 666
\DTLgidxDoSeeOrLocation: made robust 665
\DTLgidxForeachEntry: made robust 777
switched to \DTLmapdata . . . 777

\DTLgidxFormatSeeAlso: changed to new document command 663
\DTLgidxPostLocation: new . 647
\DTLgidxSeeList: added group 664
changed to new document command 664
\DTLgidxSetColumns: mad robust 646
\DTLgmax: made robust 204
\DTLgmaxall: made robust 205
\DTLgmeanforall: made robust 205
\DTLgmin: made robust 203
\DTLgminall: made robust 204
\DTLgneg: made robust 203
\DTLground: made robust 209
\DTLgsdforall: made robust . . 208
\DTLgsqrt: made robust 203
\DTLgsub: made robust 201
\DTLgtrunc: made robust 209
\DTLgvarianceforall: made robust 207
\dtlheaderformat: removed \hfil 516
\DTLidxFormatSeeItem: added group 665
changed to new document command 665
\DTLifaction: new 330
\DTLifAllLowerCase: made robust 220
\DTLifAllUpperCase: made robust 220
\DTLifanybibfieldexists: made robust 790
\DTLifcasedatatype: made robust 228
\DTLifclosedbetween: made robust 247
\DTLifcurrency: made robust . 227
\DTLifcurrencyunit: made robust 228
\DTLifdate: new 227
\DTLifdatetime: new 226
\DTLifEndsWith: new 245
\DTLifeq: made robust 243
\DTLifgt: made robust 241
\DTLifhaskey: made robust . . . 405
\DTLifinlist: rewritten in LaTeX3 31
\DTLifint: made robust 226
\dtlifintclosedbetween: switched to LaTeX3 27
\dtlifintopenbetween: switched to LaTeX3 26
\DTLiflt: made robust 240
\DTLifnumclosedbetween: made robust 230
\dtlifnumclosedbetween: made robust 299, 306
\DTLifnumeq: made robust 230
\dtlifnumeq: made robust 298, 305
\DTLifnumerical: made robust 225
\DTLifnumgt: made robust 230
\dtlifnumgt: made robust 298, 305
\DTLifnumlt: made robust 230
\dtlifnumlt: made robust 298, 305
\DTLifnumopenbetween: made robust 231
\dtlifnumopenbetween: made robust 299, 306
\DTLifopenbetween: made robust 249
\DTLifreal: made robust 226
\DTLifStartsWith: made robust 245
\DTLifstring: made robust . . . 227
\DTLifstringclosedbetween: made robust 246
\DTLifstringeq: made robust . 242
\DTLifstringgt: made robust . 241
\DTLifstringlt: made robust . 239
\DTLifstringopenbetween: made robust 248
\DTLifSubString: added starred version 244
made robust 244
\DTLiftemporal: new 225
\DTLiftime: new 227
\DTLinbooktitlefmt: new . . . 795
\DTLinitialpunc: new 219
\DTLinitials: changed to new document command 215
\DTLinseries: new 795
\dtlinsertinto: made robust . 47
\DTLinttype: changed to \newcommand 408
\DTLisFPopenbetween: made synonym of \DTLisnumopenbetween . 254
\DTLisiPrefix: new 253
\DTLisiSuffix: new 253

\DTLisnumeq: new 256
\DTLisnumgt: new 256
\DTLisnumgteq: new 257
\DTLisnumlt: new 256
\DTLisnumlteq: new 257
\DTLjournalfmt: new 795
\dtllettergroup: new 44
\DTLlistand: new 23
\DTLlistelement: rewritten in LaTeX3 31
\DTLloadbbl: add check for begin document hook 780
allow empty name to indicate default 781
made robust 780
\DTLloadmbbl: made robust . . . 848
\DTLmanualtitlefmt: new . . . 795
\DTLmapdata: new 467
\DTLmapdatabreak: new 471
\DTLmapget: new 482
\DTLmapgetvalues: new 485
\DTLmaprow: new 481
\DTLmaprowbreak: new 482
\DTLmax: made robust 204
\DTLmaxall: made robust 204
\dtlmaxall: new 295, 302, 310
\DTLmaxforcolumn: changed to new document command 560
changed to use \DTLmapdata . 560
\DTLmaxforkeys: changed to new document command 558
changed to use \DTLmapdata . 558
\DTLmbibliography: made robust 849
\DTLmeanforall: made robust . 205
\dtlmeanforall: new . . . 302, 310
\DTLmeanforcolumn: changed to new document command 547
switched to using \DTLmapdata 547
\DTLmeanforkeys: changed to new document command 545
changed to use \DTLmapdata . 545
\DTLmin: made robust 203
\DTLminall: made robust 204
\dtlminall: new 294, 301, 309
\DTLminforcolumn: changed to new document command 557
changed to use \DTLmapdata . 557
\DTLminforkeys: changed to new document command 555
changed to use \DTLmapdata . 555
\DTLmul: made robust 201
\DTLmultibibs: made robust . . 846
\DTLneg: made robust 203
\DTLnewbibitem: made robust . 781
\DTLnewbibliteralitem: new 782
\DTLnewbibrow: made robust . . 781
\DTLnewcurrencysymbol: changed to document command . 55
\DTLnewdb: changed to new document command 399
\DTLnewdbentry: switch to new document command 431
\DTLnewrow: made robust 403
\dtlnonlettergroup: new . . . 44
\dtlnovalue: moved from datatool to datatool-base 66
\dtlnumbergroup: new 44
\DTLnumbernull: moved from datatool to datatool-base 66
\DTLnumcompare: new 231
\dtlnumericformat: new 517
\DTLnumitemsinlist: rewritten in LaTeX3 31
\DTLofseries: new 795
\DTLofseriesfmt: new 795
\dtlpadleadingzeros: new . . 28
\dtlpadleadingzerosminus: new 28
\dtlpadleadingzerosplus: new 28
\DTLparse: new 86
\DTLpieatsegment: new 911
\DTLpiechart: switched to using \DTLmapdata 926
\DTLplotdisplayticklabel: new 939
\DTLplotdisplayXticklabel: new 939
\DTLplotdisplayYticklabel: new 939
\DTLplotlegendname: new . . 1035
\DTLplotlegendnamesep: new 1035
\DTLplotlegendsetname: new 1035
\DTLplotlegendsetxlabel: new 1036
\DTLplotlegendsetylabel: new 1036

\DTLplotlegendx: new 1036
\DTLplotlegendxy: new 1035
\DTLplotlegendxysep: new . 1035
\DTLplotlegendy: new 1036
\DTLpostchaptername: new . . 795
\DTLpostnumbername: new . . . 795
\DTLpostpagename: new 794
\DTLpostvolnum: new 794
\DTLpostvolumename: new . . . 795
\DTLpostvon: new 794
\DTLprecite: new 795
\DTLPreProcessCurrencyGroup: new 43
\DTLPreProcessDateGroup: new 44
\DTLPreProcessDateTimeGroup: new 43
\DTLPreProcessDecimalGroup: new 43
\DTLPreProcessIntegerGroup: new 43
\DTLPreProcessLetterGroup: new 43
\DTLPreProcessNonLetterGroup: new 43
\DTLPreProcessTimeGroup: new 44
\DTLproceedingstitlefmt: new 795
\DTLprotectedsaverawdb: rewritten to use \DTLwrite . 624
\DTLrealtype: changed to \newcommand 408
\dtlrecombine: made robust . . 449
\dtlrecombineomitcurrent: made robust 449
\DTLreconstructdata: new . . 625
\DTLreconstructdatabase: new 626
\DTLreconstructdbdata: new 626
\DTLremoveentryfromrow: made robust 504
\dtlremoveentryincurrentrow: made robust 452
\dtlreplaceentryincurrentrow: made robust 451
\DTLresetLanguage: new 261
\DTLresetpredefined: new . . 817
\DTLresetpredefinedabbrv: new 818
\DTLresetRegion: new 262
\DTLrmentry: new 480
\DTLrmrow: new 481
\dtlroot: bug fix: added missing third argument 307
\DTLround: made robust 209
\DTLrowincr: new 494
\DTLrowreset: new 494
\DTLsavedb: rewritten to use \DTLwrite 624
\DTLsaverawdb: rewritten to use \DTLwrite 624
\DTLsavetexdb: rewritten to use \DTLwrite 624
\DTLscinum: new 9
\DTLsdforall: made robust . . . 207
\dtlsdforall: new 303, 312
\DTLsdforcolumn: changed to new document command 553
changed to use \DTLmapdata . 553
\DTLsdforkeys: changed to new document command 552
changed to use \DTLmapdata . 552
\DTLsetcurrencydatum: new . 91
\DTLsetdecimaldatum: new . . 91
\DTLsetdefaultcurrency: changed to document command 186
\DTLsetdelimiter: changed to new document command and set up reg exp 315
\DTLsetentry: new 476
\DTLsetfpdatum: new 90
\DTLsetheader: made robust . . 428
\DTLsetintegerdatum: new . . 90
\DTLsetLocaleOptions: new . 265
\DTLsetseparator: changed to new document command and detokenize 314
\DTLsetstringdatum: new . . . 93
\DTLsettemporaldatum: new . 92
\DTLsetup: new 24
\DTLshufflelist: new 49
\dtlsort: rewritten in LaTeX3 . . . 586
\dtlsortdatavalue: new 583
\DTLsortedactual: new 40
\DTLsortedletter: new 40
\DTLsortedvalue: new 40
\DTLsortlettercasehandler: new 46
\DTLsortletterhandler: new 46

\DTLsortwordcasehandler: new 45
\dtlSortWordCommands: new . 32
\DTLsortwordhandler: new . . 45
\DTLsortwordlist: new 34
\dtlspecialvalue: new 592
\dtlsplitrow: made robust . . . 450
\DTLsplitstring: made robust 221
\DTLsqrt: made robust 203
\DTLStoreInitialGetLetter: new 219
\DTLstoreinitials: changed to new document command 216
\DTLstringnull: moved from datatool to datatool-base 66
\DTLstringtype: changed to \newcommand 408
\DTLsub: made robust 201
\DTLsubstitute: made robust . 220
rewritten in LaTeX3 220
\DTLsubstituteall: made robust 221
rewritten in LaTeX3 221
\DTLsumcolumn: changed to new document command 543
changed to use \DTLmapdata . 543
\DTLsumforkeys: changed to new document command 541
changed to use \DTLmapdata . 541
\dtlswapentriesincurrentrow: made robust 453
\dtlswaprows: made robust . . . 537
obey global setting 538
\DTLtemporalvalue: new 112
\dtltexorsort: new 32
\DTLtherow: new 494
\DTLthesistitlefmt: new . . . 795
\dtltimeformat: new 517
\dtltimegroup: new 44
\DTLtotalbargroups: new . . . 852
\DTLtotalbars: new 852
\DTLtrunc: made robust 209
\dtlunknowntag: new 747
\DTLunsettype: changed to \newcommand 408
\dtlupdateentryincurrentrow: check global setting 456
made robust 455
\DTLuse: new 332
\DTLusedatum: new 78
\DTLvarianceforall: made robust 206
\dtlvarianceforall: new 302, 311
\DTLvarianceforcolumn: changed to new document command 551
switched to using \DTLmapdata 551
\DTLvarianceforkeys: changed to new document command . . . 549
changed to use \DTLmapdata . 549
\DTLwrite: new 606
\DTLxparse: new 88
\DTLxsetcurrencydatum: new 92
\DTLxsetdecimaldatum: new . 91
\DTLxsetintegerdatum: new . 90
\DTLxsetstringdatum: new . . 93
\DTLxsettemporaldatum: new 92
\DTLxsplitstring: new 221
\DTLyAxisLabelStyle: new . . 854
\edtlgetrowforvalue: made robust 446
\femalelabels: replaced with \g_person_female_label_clist 1042
\forallpeople: new 1062
\gDTLformatbibentry: made robust 784
\getpersonforenames: new . 1081
\getpersonfullname: made robust 1081
\getpersongender: made robust 1080
\getpersongenderlabel: new 1081
\getpersonname: made robust 1081
\getpersonsurname: new . . . 1081
\getpersontitle: made robust 1082
\if@datagidx@warn: replaced with \l__datagidx_warn_bool 653
\if@datagidxsymbolleft: replaced with \l__datagidx_symbol_left_bool 649
\if@dtl@sequential: replaced with \l__datagidx_sequential_bool 700
\ifallfemale: deprecated . . . 1059
\ifallmale: deprecated 1058

\iffemale: deprecated 1059
\iffemalelabel: deprecated . 1044
\ifmale: deprecated 1058
\ifmalelabel: deprecated . . . 1043
\loadgidx: made robust 708
\malelabels: replaced with \g_person_male_label_clist 1042
\newperson: added starred form 1044
made robust 1044
\newterm@database: replaced with \l__datagidx_term_database_tl 721
\newterm@defaultshook: replaced with \g__datagidx_term_defaults_tl 725
\newterm@description: replaced with \l__datagidx_term_description_tl 721
\newterm@extrafields: replaced with \g__datagidx_extra_fields_tl 725
\newterm@label: replaced with \l__datagidx_term_label_tl 721
\newterm@long: replaced with \l__datagidx_term_long_tl 721
\newterm@longplural: replaced with \l__datagidx_term_longplural_tl 721
\newterm@name: replaced with \l__datagidx_term_name_tl 721
\newterm@parent: replaced with \l__datagidx_term_parent_tl 721
\newterm@plural: replaced with \l__datagidx_term_plural_tl 721
\newterm@see: replaced with \l__datagidx_term_see_tl 721
\newterm@seealso: replaced with \l__datagidx_term_seealso_tl 721
\newterm@short: replaced with \l__datagidx_term_short_tl 721
\newterm@shortplural: replaced with \l__datagidx_term_shortplural_tl 721
\newterm@sort: replaced with \l__datagidx_term_sort_tl 721
\newterm@symbol: replaced with \l__datagidx_term_symbol_tl 721
\newterm@text: replaced with \l__datagidx_term_text_tl 721
\newtermaddfield: check for column existence 727
\newtermsorthook: new 729
\Peoplechild: added warning if no people defined 1079
made robust 1079
switched to LaTeX3 for case change 1079
\peoplechild: added warning if no people defined 1079
made robust 1079
\peopleforenames: new 1070
\peoplefullname: added warning if no people 1070
made robust 1070
\peoplename: made robust . . . 1071
\Peopleobjpronoun: added warning if no people defined . 1075
made robust 1075
\peopleobjpronoun: added warning if no people defined . 1075
made robust 1075
\Peopleobjpronounii: new . 1077
\peopleobjpronounii: new . 1077
\Peopleparent: added warning if no people defined 1079
made robust 1079
switched to LaTeX3 for case change 1079
\peopleparent: added warning if no people defined 1079
made robust 1079

\Peoplepossadj: added warning if no people defined 1076
made robust 1076
switched to LaTeX3 for case change 1076
\peoplepossadj: added warning if no people defined 1076
made robust 1076
\Peoplepossadjii: new 1078
\peoplepossadjii: new 1078
\Peopleposspronoun: added warning if no people defined . 1076
made robust 1076
switched to LaTeX3 for case change 1076
\peopleposspronoun: added warning if no people defined . 1076
made robust 1076
\Peopleposspronounii: new 1078
\peopleposspronounii: new 1078
\Peoplepronoun: added warning if no people defined 1075
made robust 1075
switched to LaTeX3 for case change 1075
\peoplepronoun: added warning if no people defined 1075
made robust 1075
\Peoplepronounii: new 1077
\peoplepronounii: new 1077
\Peoplesibling: new 1080
\peoplesibling: added warning if no people defined 1080
made robust 1080
\peoplesurname: new 1071
\peopletitlesurname: new . 1072
person: removed 1040
\PersonAddFemaleLabel: made robust 1042
\PersonAddMaleLabel: made robust 1042
\PersonAddNonBinaryLabel: new 1043
\Personchild: made robust . . 1079
switched to LaTeX3 for case change 1079
\personchild: made robust . . 1079
\PersonFemaleCount: new . . 1040
\personforenames: new 1070
\personfullname: made robust 1070
\Persongender: new 1080
\PersonIfAllFemale: new . . 1059
\PersonIfAllMale: new 1058
\PersonIfAllNonBinary: new 1060
\PersonIfAllUnknownGender: new 1061
\PersonIfFemale: made robust 1058
\PersonIfFemaleLabel: made robust 1043
\PersonIfMale: new 1058
\PersonIfMaleLabel: new . . 1043
\PersonIfNonBinary: new . . 1059
\PersonIfNonBinaryLabel: new 1044
\PersonIfUnknownGender: new 1060
\personlastsep: use datatool-base separators 1069
\PersonMaleCount: new 1040
\personname: made robust . . . 1071
\PersonNonBinaryCount: new 1040
\Personobjpronoun: made robust 1075
switched to LaTeX3 for case change 1075
\personobjpronoun: made robust 1075
\Personobjpronounii: new . 1077
\personobjpronounii: new . 1077
\Personparent: made robust . 1079
switched to LaTeX3 for case change 1079
\personparent: new 1079
\Personpossadj: made robust 1076
switched to LaTeX3 for case change 1076
\personpossadj: made robust 1076
\Personpossadjii: new 1078
\personpossadjii: new 1078
\Personposspronoun: made robust 1076
switched to LaTeX3 for case change 1076
\personposspronoun: made robust 1076
\Personposspronounii: new 1078
\personposspronounii: new 1078
\Personpronoun: made robust 1075

switched to LaTeX3 for case change 1075
\personpronoun: made robust 1075
\Personpronounii: new 1077
\personpronounii: new 1077
\PersonSetFemaleLabels: new 1042
\PersonSetLocalisation: new 1063
\PersonSetMaleLabels: new 1042
\PersonSetNonBinaryLabels: new 1043
\Personsibling: made robust 1080
switched to LaTeX3 for case change 1080
\personsibling: made robust 1080
\personsurname: new 1071
\persontitlesurname: new . 1071
\persontitlesurnamesep: new 1072
\PersonTotalCount: new . . . 1039
\PersonUnknownCount: new . 1040
\removeallpeople: made robust 1053
\removepeople: made robust . 1053
\removeperson: made robust . 1052
moved existence check 1052
\RequireDatatoolDialect: new 263
\s@dtlformatlist: rewritten to (partially) use LaTeX3 50
\seealsoname: added check for \alsoname 663
dtl@createalphabiblabels: switched to read-only loop . . . 841
dtlbar@yticlabels: replaced with \l__databar_yticlabels_seq 854
dtlbar@yticlist: replaced with \l__databar_yticpoints_seq 854
\twopeoplesep: use datatool-base separators 1069
\xDTLassignfirstmatch: made robust 435
\xDTLinitials: new 215
3.1 – 2025-03-03
\DTLsortlettercasehandler: changed \tl_set:Nx to \tl_set:Ne 46
\DTLsortletterhandler: changed \tl_set:Nx to \tl_set:Ne 46
moved case-change 46
\DTLsortwordcasehandler: changed \tl_set:Nx to \tl_set:Ne 45
\DTLsortwordhandler: changed \tl_set:Nx to \tl_set:Ne 45
moved case-change 45
3.1 – 2025-03-10
\DTLmeanforall: changed \tl_set:Nx to \tl_set:Ne 205
\DTLmeanforcolumn: expand result 548
\DTLmeanforkeys: expand result 545
\DTLsdforall: changed \tl_set:Nx to \tl_set:Ne 208
\DTLsdforcolumn: expand result 554
\DTLsdforkeys: expand result . 553
\DTLsumcolumn: expand result . 544
\DTLsumforkeys: expand result 543
\DTLvarianceforall: changed \tl_set:Nx to \tl_set:Ne 207
\DTLvarianceforcolumn: expand result 551
\DTLvarianceforkeys: expand result 550
3.3 – 2025-03-15
General: bug fix: multiple separators in delimited cell 640
3.4.1 – 2025-04-25
\dtl@get@yearsuffix: bug fix: incorrect parameters 845
3.4.2 – 2025-11-06
General: switched to \csedef and \csdef 187
\dtllastloadeddb: corrected spelling 599
3.4.3 – 2025-12-04
General: clear or defined 314
3.4.4 – 2026-07-31
General: corrected misspelt command 282

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.

Symbols

\" 1225, 1286
\# 33, 54, 608, 734, 1176, 1290
\$ 33, 53, 185, 187, 734, 1176, 1290, 1301, 1305, 1320
\% 33, 608, 628, 734, 782, 1139, 1176, 1281–1284, 1289, 1290
\& 23, 33, 49, 261, 665, 733, 734, 1142, 1176, 1290, 1298
\' 53, 283
\(..... 259–261, 1205, 1357, 1358
\) 259–261, 1205, 1357, 1358
\+ 122, 123, 150
\, ... 62, 166
\- 46, 56, 57, 59–61, 93, 122, 123, 150–156, 213
\. 56, 57, 93, 152, 154, 156, 157, 790, 792, 1102
\/ 150–155
\: 150, 157
\@ 273
\@@ 487–491, 1237, 1238, 1245–1247, 1275
\@@dtl@set@off 929, 1204, 1205
\@@dtl@set@offr 1205
\@@dtl@setnull . 440, 440, 1237, 1238
\@@dtl@setoffr 929
\@@dtl@strip@numgrpchar ... 1302
\@@get@firstperson 1041, 1364
\@Alph 1183, 1184, 1195
\@DTLaddcolumn 418, 1232
\@DTLaddcolumnwithheader 418, 419
\@DTLforeach 495, 496, 501, 786, 787, 1100, 1101, 1249, 1250
\@DTLforeachbibentry .. 787, 1100
\@DTLgetrowindex 461, 462
\@DTLifEndsWith 245, 245, 253
\@DTLifStartsWith ... 245, 245, 252
\@DTLifSubString 244, 244, 252
\@DTLifclosedbetween 247, 247, 254, 1350, 1351
\@DTLifdbempty 392, 402, 403, 403, 497, 529, 1228, 1250, 1253
\@DTLifeq 243, 243, 251, 1346
\@DTLifgt 242, 242, 251, 1345
\@DTLifhaskey 405, 405, 491, 505, 507, 629, 1229, 1247, 1256, 1257
\@DTLiflt 240, 240, 250, 1344
\@DTLifopenbetween 249, 249, 255, 1352
\@DTLifstringclosedbetween 246, 246, 247, 1350, 1351
\@DTLifstringeq . 242, 243, 244, 1346
\@DTLifstringgt . 241, 241, 242, 1345
\@DTLifstringlt . 239, 239, 240, 1344
\@DTLifstringopenbetween 248, 249, 1351, 1352
\@DTLnewdbentry 432, 432, 1099, 1235
\@DTLnewrow 403, 403, 1099, 1229
\@DTLreconstructdata ... 626, 626
\@DTLremoverow 540, 540, 1267
\@DTLsetheader 428, 428, 1234
\@DTLsort 586, 586, 1274
\@Roman 730, 1174, 1183, 1184
\@TrackLangAddToHook 264
\@after 1293
\@afterheading 675, 689, 693, 1150, 1159, 1161
\@alph 1183, 1184
\@arabic 1183, 1184
\@auxout 653, 758, 781, 814, 1099, 1112, 1138, 1187, 1196
\@before 1293
\@begindocumenthook 780
\@bsphack 848, 1131
\@checkend 270, 1293
\@cite 847, 1131
\@cite@ofmt 848, 1131
\@citea 847, 1131
\@citeb 847, 848, 1131, 1132
\@cur@location 1197
\@currenvir 269, 1293
\@data@rerun@warn 654, 1138, 1196, 1197
\@data@rerun@warn@sort 653, 654, 1138

\@datagidx@db@col@id@w 655, 1138, 1139
\@datagidx@do@xusedentry .. 1187
\@datagidx@dorerun@warn 653, 654, 1138, 1196, 1197
\@datagidx@dorerun@warn@sort 653, 654, 654, 1138
\@datagidx@dowriteaux 1184
\@datagidx@escloc . 752, 1183, 1184
\@datagidx@fieldkey@Description 1173
\@datagidx@fieldkey@Label . 1173
\@datagidx@fieldkey@Long .. 1173
\@datagidx@fieldkey@Name .. 1173
\@datagidx@fieldkey@Parent 1173
\@datagidx@fieldkey@Plural 1173
\@datagidx@fieldkey@See ... 1173
\@datagidx@fieldkey@SeeAlso 1173
\@datagidx@fieldkey@Short . 1173
\@datagidx@fieldkey@Sort .. 1173
\@datagidx@fieldkey@Symbol 1173
\@datagidx@fieldkey@Text .. 1173
\@datagidx@parse@location 703, 703, 1167
\@datagidx@rerun@warn 654
\@datagidx@rerun@warn@sort . 654
\@datagidx@target . 661, 1184, 1185
\@datagidx@thisval 1142, 1143
\@datagidx@use@entry 756, 1186, 1190
\@datagidx@warnfalse 1139
\@datagidx@warntrue ... 1138, 1139
\@datagidxlink 1146
\@datagidxsymbolleftfalse 1136, 1137
\@datagidxsymbollefttrue 1136, 1137
\@datagidxtarget 1146
\@do@dtlenvforeach 1249
\@do@dtlenvgforint 1358
\@do@dtlnewentry 1288
\@do@foreachperson 1367
\@do@sdtlenvforeach 1250
\@doendpe 668, 776
\@dtl@A@after 1338, 1340
\@dtl@A@before 1338–1340
\@dtl@A@chargroup 1339, 1340
\@dtl@A@comma 1338, 1340
\@dtl@B@after 1339, 1340
\@dtl@B@before 1339, 1340
\@dtl@B@chargroup 1339, 1340
\@dtl@B@comma 1339, 1340
\@dtl@a 1276–1279
\@dtl@activatebraces 1290
\@dtl@af 1130, 1131
\@dtl@after 415, 420, 422, 429, 430, 443, 500, 507, 508, 1232, 1233, 1235, 1239, 1252, 1258
\@dtl@after@cs 1241, 1242
\@dtl@afterpart 1318, 1319, 1338, 1339
\@dtl@arg 1291, 1292
\@dtl@argA 1324, 1325, 1334, 1335, 1337, 1338, 1347, 1348
\@dtl@argB 1324, 1325, 1334, 1335, 1337–1339, 1347, 1348
\@dtl@argi 1102, 1130, 1301
\@dtl@argii 1102, 1130, 1301
\@dtl@argiii 1130
\@dtl@asg@value 1237
\@dtl@assign 435, 436, 437, 499, 1236, 1237, 1251, 1254
\@dtl@assigncmd ... 440, 1237, 1238
\@dtl@assigncmdnoop ... 440, 1238
\@dtl@assigntmpseq 30, 31, 50, 198, 199, 206, 207
\@dtl@author 1102–1106, 1129
\@dtl@authorcount . 1102–1105, 1129
\@dtl@b 1276, 1277, 1279
\@dtl@bar 876, 1088–1090
\@dtl@barcount 856, 1083, 1094–1096
\@dtl@before 415, 420, 422, 429, 430, 443, 500, 507, 508, 1232, 1233, 1235, 1239, 1252, 1258
\@dtl@before@cs 1241, 1242
\@dtl@beforepart 1318, 1319, 1338, 1339
\@dtl@c 1281
\@dtl@cap 1262, 1263
\@dtl@chars 645, 1226
\@dtl@chcknumnext 1321, 1322
\@dtl@checknumerical 225, 225, 421, 423, 1233, 1234, 1319, 1322, 1323, 1351–1353
\@dtl@checknumericalloop 1321, 1322
\@dtl@checknumericalnoop 1321, 1322
\@dtl@checknumericalstart 1320, 1321

\@dtl@chop@trailingzeroes . 1301
\@dtl@chopexcessfrac 1304
\@dtl@choptrailingzeroes ...
..... 1300, 1301
\@dtl@chopzeroesnext 1301
\@dtl@cite@write 847, 847, 848
\@dtl@col 400,
401, 443, 444, 566, 1227, 1228,
1239, 1267–1269, 1271, 1272,
1274, 1276, 1277, 1279, 1283, 1284
\@dtl@colhead 415, 420, 422,
429, 430, 443, 1232, 1233, 1235, 1239
\@dtl@colnum 429, 430, 1235
\@dtl@comparecs 1275, 1278
\@dtl@cond 1267–1272
\@dtl@conditionfalse .....
..... 226, 250–259, 808–812,
1109–1112, 1286–1288, 1316,
1318, 1324, 1347–1349, 1353–1357
\@dtl@conditiontrue 226,
250–259, 808–812, 1109–1111,
1287, 1288, 1315–1318,
1324, 1347, 1348, 1353–1357
\@dtl@construct@getintfrac .
..... 1299, 1300
\@dtl@construct@getnums ....
..... 1299, 1300, 1305
\@dtl@construct@lop@ff .....
..... 645, 645, 1226
\@dtl@construct@lopoff .....
..... 644, 645, 1225, 1226
\@dtl@construct@lopoffs ....
..... 645, 1225, 1226
\@dtl@construct@qlopoff ....
..... 645, 645, 1225, 1226
\@dtl@construct@stripnumgrpchar
..... 1299, 1302, 1305
\@dtl@contcap 1262–1264
\@dtl@countdigits .....
... 269, 308, 309, 1303, 1362, 1363
\@dtl@countnext 269, 1303
\@dtl@curi 495, 1249
\@dtl@curii 495, 1249
\@dtl@curiii 495, 1249
\@dtl@currencies 185, 1305
\@dtl@currency 10,
102, 108, 185, 185, 186, 189, 190,
228, 325, 328, 1300, 1304, 1305, 1320
\@dtl@currentrow .. 271, 1295–1297
\@dtl@datatype .....
... 6, 37, 38, 41, 63, 70, 75, 77,
86, 87, 94, 96, 97, 100, 102–104,
116, 117, 119, 147, 194–200, 206,
207, 225–231, 363–369, 376, 377,
386, 421–424, 429–431, 452, 454,
456, 478, 541–548, 550–552, 554,
556, 557, 559, 561, 563, 564,
571, 590, 634–639, 1233, 1234,
1261, 1264, 1319–1323, 1351–1353
\@dtl@db@name 1267–1269, 1271, 1272
\@dtl@dbname 314, 438–440,
496, 498, 512, 775, 1194, 1237,
1238, 1250, 1251, 1253, 1275–1278
\@dtl@decimal 56, 1299,
1300, 1302, 1304, 1305, 1321, 1322
\@dtl@decimal@to@localeint 1303
\@dtl@decimaltolocale . 1302, 1304
\@dtl@decimaltolocalefrac . 1303
\@dtl@decimaltolocaleint ...
..... 1302, 1303
\@dtl@decrementrows .....
..... 539, 539, 1265, 1266
\@dtl@decvals 1311, 1312
\@dtl@def@write 624
\@dtl@delimiter 315, 315,
610, 628, 645, 1225, 1226, 1279, 1280
\@dtl@dict@compare 1341
\@dtl@diff 1311, 1312
\@dtl@digitcount 1303, 1304
\@dtl@digitcountnext 1304
\@dtl@dir 1278
\@dtl@do@compare 1341
\@dtl@do@getentry 1242
\@dtl@do@splitrow 1241, 1242
\@dtl@do@stripnumgrpchar .. 1302
\@dtl@doamp 1261–1264
\@dtl@dobegintab 1261, 1263
\@dtl@docompare ... 1110, 1111, 1277
\@dtl@doforcol 1247, 1248
\@dtl@dogetdata 409, 1231
\@dtl@dogetentry 1237
\@dtl@dogetkeydata 1239
\@dtl@dogetkeyforcolumn 408, 1230
\@dtl@dogetprops 1233, 1235
\@dtl@dogetrow 1240
\@dtl@dogetrowforvalue 1240
\@dtl@dogetval 566, 1274
\@dtl@doifinlist 1293
\@dtl@dolabel 1204

\@dtl@dollar 1320, 1321
\@dtl@donext 1102, 1130, 1347
\@dtl@donextdec 539, 1266
\@dtl@donextinitials 1314
\@dtl@doreadline . 1286, 1288, 1289
\@dtl@doreadraw 1290
\@dtl@dosetheader 1289
\@dtl@dosplit 537, 538, 1265
\@dtl@dosplitrow 505, 507, 1256, 1257
\@dtl@dostartrow 1262, 1264
\@dtl@dosubs . . 645, 1225, 1226, 1305
\@dtl@dosubstitute 1319
\@dtl@dosubstitutenoop 1319
\@dtl@dosubstnext 1319
\@dtl@dotestifsubstring . . . 1347
\@dtl@dovalue 299, 300, 1350–1352, 1359, 1360
\@dtl@element 1294, 1295
\@dtl@elements 544, 1268–1270
\@dtl@endgrabword 1341, 1342
\@dtl@endloophook . . 259, 260, 1357
\@dtl@endpt 1095
\@dtl@entryI 453, 1242
\@dtl@entryII 453, 1242
\@dtl@falsepart 305, 306, 1361, 1362
\@dtl@firstpart 537, 538, 1265
\@dtl@firsttonil . . . 273, 274, 1327
\@dtl@firsttype 1323
\@dtl@foot 1262, 1263
\@dtl@forcolnext 493, 1248
\@dtl@forcolnoop 493, 493, 1248, 1249
\@dtl@forcolumn 493, 1248
\@dtl@foreach@level 260, 260, 261, 487, 488, 490, 491, 1245–1247, 1358
\@dtl@foreachkey 489, 490, 1246, 1247
\@dtl@foreachnext 487, 488, 490, 491, 1246, 1247
\@dtl@foreachnoop 487, 488, 490, 1246, 1247
\@dtl@foreachrow 486, 487, 1245, 1246, 1275
\@dtl@formatlist@handler . . . 50, 50, 1298, 1299
\@dtl@formatlist@item 1299
\@dtl@formatlist@itemsep . . . 50, 1298, 1299
\@dtl@formatlist@lastitem .. 50, 1298, 1299
\@dtl@formatlist@prelastitem 50, 1298, 1299
\@dtl@formatlist@prelastitemsep 50, 1298, 1299
\@dtl@fracpart 308, 309, 1300, 1302, 1304, 1305, 1362, 1363
\@dtl@gap 932, 1222, 1223
\@dtl@gatherintfrac 308, 308, 309, 1362, 1363
\@dtl@get@int@part 1300, 1301
\@dtl@get@intpart 1300
\@dtl@get@keydata . 410, 1231, 1232
\@dtl@get@keyforcolumn 408, 1230, 1231
\@dtl@get@next@intpart 1301
\@dtl@get@nextintpart 1301
\@dtl@get@sortdirection 592, 1278
\@dtl@getcolumnindex 406, 406, 505, 507, 1229, 1256, 1257
\@dtl@getdatatype .. 409, 409, 1231
\@dtl@getfracpart 1300
\@dtl@getintfrac 1299, 1300
\@dtl@getkeyforcolumn 329, 389, 390, 408, 408, 1230
\@dtl@getlocation 1244
\@dtl@getprops 410, 1232, 1233, 1235, 1239
\@dtl@getrow 1240
\@dtl@getrowindex 1245
\@dtl@getsortdirection 592, 1274, 1276, 1278
\@dtl@getvalue 1244
\@dtl@gobbletonil . . . 269, 539, 1266, 1301, 1302, 1304, 1315–1318
\@dtl@h .. 1087, 1088, 1093, 1094, 1281
\@dtl@head 400, 401, 443, 444, 1227, 1228, 1239, 1261–1264, 1279, 1283–1285, 1289
\@dtl@header 1279, 1280
\@dtl@idx 1261–1264
\@dtl@ifDigitOrDecimalSep .. 1321, 1322
\@dtl@ifsingle 270, 1291, 1292
\@dtl@initials 1314
\@dtl@insertdonefalse 47, 271, 1275, 1296, 1297
\@dtl@insertdonetrue 47, 271, 1276, 1296, 1297
\@dtl@intpart 308, 309, 1300–1305, 1362, 1363

\@dtl@k 1281
\@dtl@key 400, 401, 420, 429, 430, 443, 444, 1227, 1228, 1233, 1235, 1239, 1241, 1261–1264, 1267–1269, 1271, 1272, 1274–1280, 1283–1285, 1287
\@dtl@key@Author 794, 841, 1102, 1103, 1127
\@dtl@key@CrossRef 842, 1128
\@dtl@key@Editor 794, 797, 841, 1103–1105, 1127
\@dtl@key@EprintType 808
\@dtl@key@Key 841, 1128
\@dtl@key@Organization 841, 1128
\@dtl@key@Year 842, 1128
\@dtl@keya 1276, 1277
\@dtl@keyb 1276, 1277
\@dtl@keys 489, 1246
\@dtl@label 1262, 1263
\@dtl@lastfoot 1262, 1263
\@dtl@level 1274–1276
\@dtl@lin@ 1287, 1288
\@dtl@line 1286–1288
\@dtl@list 586, 1274
\@dtl@listelement@outofrange 270, 1294
\@dtl@localeintnext 1303
\@dtl@loop@body 511, 1259
\@dtl@loopbody 487, 489, 490, 1245–1247
\@dtl@lop@ff . . . 644, 645, 1225, 1226
\@dtl@lopoff . . 644, 1225, 1287, 1288
\@dtl@map 642, 1290
\@dtl@mathprocessor 3, 8, 24, 76, 1226, 1291, 1359, 1361
\@dtl@max 1310
\@dtl@maxcols 1261, 1262, 1289
\@dtl@mean 303, 304, 311, 312, 1310–1312
\@dtl@midpt 1095, 1096
\@dtl@min 539, 1265, 1266, 1309
\@dtl@minus 1320
\@dtl@n 1268–1270, 1310–1312
\@dtl@neggap 932, 1221, 1222
\@dtl@newlist 539, 1265, 1266, 1275, 1276
\@dtl@newsortedlist 271, 272, 1296–1298
\@dtl@newstuff . 271, 272, 1296, 1297
\@dtl@next . . . 1201–1203, 1238, 1302
\@dtl@nexti 495, 1249
\@dtl@nextii 495, 1249
\@dtl@nextiii 495, 1249
\@dtl@notdone . . 509, 510, 1258, 1259
\@dtl@num 1306, 1309–1312
\@dtl@numbergroupchar 56, 1299, 1302, 1303, 1305, 1322
\@dtl@numgrpsepcount 1299, 1303, 1320, 1322
\@dtl@numgrpsepfalse 1319
\@dtl@numgrpseptrue 1322
\@dtl@numi 230, 231, 1305–1309, 1312, 1313, 1324, 1345, 1346, 1349–1351
\@dtl@numii . 230, 231, 1305–1307, 1309, 1324, 1345, 1346, 1350, 1351
\@dtl@numiii . . . 230, 231, 1350, 1351
\@dtl@oldbreak 493, 1248, 1249
\@dtl@oldtype 420, 421, 1233
\@dtl@omitlist 1262–1264
\@dtl@org@currency 1300, 1305, 1320, 1323
\@dtl@orgbreak . 259, 260, 1357, 1358
\@dtl@pages 1106, 1107
\@dtl@parse@words 1344
\@dtl@period 1102
\@dtl@plus 1320
\@dtl@posgap 932, 1222
\@dtl@previ 495, 1249
\@dtl@previi 495, 1249
\@dtl@previii 495, 1249
\@dtl@protect 1320
\@dtl@qlopoff 644, 645, 1225
\@dtl@rawmappings 641, 642, 642, 1290
\@dtl@rawread 1286, 1289
\@dtl@read 1285–1289
\@dtl@readline 1285, 1286
\@dtl@readrawline 1286, 1289
\@dtl@reconstruct@data 623
\@dtl@replaced 221, 1305–1313, 1318, 1319, 1338, 1339
\@dtl@replacementkeys . 1274, 1277
\@dtl@resetdoamp 1261–1264
\@dtl@resetdostartrow . 1262–1264
\@dtl@row 566, 1273, 1274, 1280
\@dtl@rowAcontents 1275
\@dtl@rowAidx . . 537, 538, 1264, 1265
\@dtl@rowAnum 1275
\@dtl@rowBcontents 1275
\@dtl@rowBidx 537, 538, 1265
\@dtl@rowBnum 1275

\@dtl@rowa 591, 1275, 1276
\@dtl@rowb 591, 1275, 1276
\@dtl@s@thislabel 1128
\@dtl@sd 1312
\@dtl@secondpart 538, 1265
\@dtl@seg . 916, 1198, 1200, 1201, 1205
\@dtl@separator 314, 314, 609, 610, 628, 639, 640, 645, 1224–1226, 1279, 1280, 1287, 1288
\@dtl@sequentialfalse . 1167, 1168
\@dtl@sequentialtrue .. 1167, 1168
\@dtl@set@off 929, 1200, 1204
\@dtl@set@options 1226
\@dtl@setheaderforindex 429, 429, 627, 1235, 1289
\@dtl@setnewvalue 317, 318, 318, 430, 630, 1235, 1236
\@dtl@setnull 440, 443, 444, 1238, 1239, 1276, 1277
\@dtl@setwordbreaks 1342
\@dtl@setwordbreaks@next .. 1342
\@dtl@shift 1201
\@dtl@shortcap 1262, 1263
\@dtl@sortcriteria 592, 1275, 1276
\@dtl@sortdirection ... 1277, 1278
\@dtl@sortedlist 271, 1275, 1276, 1295–1297
\@dtl@sortorder 1274–1276
\@dtl@splitsubstr 1319
\@dtl@standardize@currency 1300, 1305, 1320
\@dtl@start 853, 917, 1088–1090, 1094–1097, 1200, 1204
\@dtl@storeandupdate ... 431, 432
\@dtl@string 1313, 1314
\@dtl@strip@numgrpchar 1301, 1302
\@dtl@stripeol 1286, 1290
\@dtl@stripped 1302
\@dtl@strippedline 1286, 1290
\@dtl@stuff 1241, 1242, 1299
\@dtl@subnobrsp 1313, 1314
\@dtl@subnobrspnext ... 1313, 1314
\@dtl@subs@argA 1347
\@dtl@subs@argB 1347
\@dtl@sum 1306
\@dtl@t 1281
\@dtl@tabargs 1261–1263
\@dtl@testifsubstring 1347
\@dtl@teststartswith .. 1347–1349
\@dtl@thirdpart 538, 1265
\@dtl@thiscurrency 1305
\@dtl@thisdb 502–508, 510, 1255–1259
\@dtl@thisdialect 850, 1082
\@dtl@thisentry 1288
\@dtl@thiskey 1288
\@dtl@thislabel ... 841, 1127, 1128
\@dtl@thisreplaced 1306, 1307
\@dtl@thisrow 539, 1266
\@dtl@thistick 1222–1224
\@dtl@tmp 269, 300–304, 317, 408, 443, 444, 503, 505, 507, 538, 642, 796, 1102, 1104, 1105, 1108–1112, 1116, 1118, 1119, 1122, 1128, 1129, 1231, 1235, 1239, 1243, 1251, 1252, 1255–1258, 1261, 1262, 1265, 1290, 1293, 1300, 1302, 1305–1309, 1312, 1313, 1319–1323, 1348, 1351–1353
\@dtl@tmpcmp 1278, 1344–1346, 1350–1352
\@dtl@tmpcount 25, 269, 308, 309, 495, 539, 809–812, 1087, 1092–1094, 1103–1105, 1110–1113, 1129, 1211, 1212, 1249, 1251, 1266, 1278, 1291, 1294, 1295, 1300, 1301, 1303, 1304, 1320, 1322, 1344–1346, 1350–1352, 1362, 1363
\@dtl@tmpcpz 1301
\@dtl@tmpdtl 1302, 1304
\@dtl@tmpi 1306, 1308–1312
\@dtl@tmpii .. 1306–1310, 1312, 1313
\@dtl@toks 25, 271, 272, 317, 318, 430, 431, 433, 503, 505, 507, 508, 538, 539, 630, 641, 642, 1110, 1111, 1128, 1129, 1218, 1224, 1235, 1236, 1239, 1243, 1255, 1256, 1258, 1265, 1266, 1287–1291, 1295–1298, 1304, 1305, 1313–1319
\@dtl@toks@gconcat@middle@cx 51, 1233, 1235, 1236, 1240, 1252, 1258, 1299
\@dtl@toks@gput@right@cx 50, 1229, 1234, 1241, 1243, 1255, 1299
\@dtl@toks@name 1299
\@dtl@toksA 537, 538, 1265, 1277, 1278, 1364

\@dtl@toksB 537, 538, 1265, 1277, 1278, 1364
\@dtl@truepart . 305, 306, 1361, 1362
\@dtl@truncatedecimal 1302
\@dtl@type 400, 401, 420–423, 429, 440, 441, 443, 444, 644, 1227, 1228, 1233–1235, 1238, 1239, 1261–1264, 1279, 1283, 1284, 1291
\@dtl@typeA 1278
\@dtl@typeB 1278
\@dtl@updatefkcs . . . 489, 490, 1246
\@dtl@updatekeys 420, 503, 1232, 1235, 1241–1243, 1255
\@dtl@useresult 332, 332
\@dtl@val 566, 1261, 1262, 1264, 1274, 1280, 1281
\@dtl@var 1311
\@dtl@vonpart 1129
\@dtl@widestlabel . 841, 1127, 1128
\@dtl@width 933, 1221–1223
\@dtl@wordbreak 237, 1324, 1328, 1334, 1335, 1337, 1342, 1347, 1348
\@dtl@write 624, 624, 1279–1284
\@dtl@year 846, 1128, 1130
\@dtldictcompare . . . 237, 237, 1337
\@dtlenvforeach@args 1249
\@dtlenvgforint@arg 1358
\@dtlforcolumn 491, 491, 1247
\@dtlforcolumnidx . . 492, 492, 1248
\@dtlforeachrow 486, 1245, 1254, 1275
\@dtlformatlist 50, 50, 1299
\@dtlgetdatatype . . . 409, 409, 1231
\@dtlgetfirstchar 274
\@dtlgetkeydata 442, 442, 1239
\@dtlgetkeyforcolumn 407, 407, 1230
\@dtlgetrow 447, 468, 530, 538, 1240, 1265
\@dtlgetrowindex 462, 1245
\@dtlifreadonly 502, 502, 504, 506, 508, 1255–1258
\@dtlloaddb 1286, 1289
\@dtlmaxforkeys 560, 1272
\@dtlmeanforkeys . . 547, 1268, 1269
\@dtlminforkeys 557, 1271
\@dtlnovalue 65, 1238, 1242, 1244, 1245
\@dtlnumbernull 66, 66, 510, 1238, 1259
\@dtlplothandlermark 964, 975, 1207, 1213, 1221
\@dtlsdforkeys 553, 1271
\@dtlsplitrow 1241
\@dtlstringnull 66, 66, 510, 1238, 1259
\@dtlsumforkeys 543, 1267
\@dtlvarianceforkeys 551, 1269, 1271
\@dtlwordindexcompare . 1340, 1341
\@eha 848, 1132
\@empty . . 221, 270, 566, 847, 1100, 1104, 1105, 1129, 1131, 1147, 1148, 1152, 1153, 1156–1158, 1164, 1193, 1212, 1213, 1217, 1218, 1223, 1224, 1274, 1293, 1295, 1300, 1302, 1306, 1307, 1309–1311, 1318, 1365, 1367
\@emptytoks 269, 1293
\@endforfalse 271, 272, 1102, 1143, 1277, 1295–1297, 1305
\@endfortrue . . 1101, 1103, 1104, 1106, 1129, 1143, 1165, 1197, 1277, 1294, 1295, 1305, 1366, 1367
\@endparse@formatlabel@ 755, 756, 1185, 1186
\@envbody 269, 1293
\@esphack 848, 1132
\@female@label . . . 1041, 1367, 1368
\@firstofone 813, 847, 848, 1131, 1141, 1184, 1325, 1335
\@firstoftwo 1142, 1201–1203
\@for 271, 300–304, 642, 847, 848, 1087, 1088, 1090, 1092–1094, 1097, 1101–1105, 1129–1131, 1141–1144, 1165, 1173, 1183, 1197, 1200, 1212–1217, 1267–1269, 1271–1274, 1276, 1277, 1285, 1290, 1294–1297, 1299, 1305, 1306, 1309–1312, 1365–1367, 1369
\@foreachperson 1062, 1367
\@forremainder 1143, 1295
\@gDTLforeachbibentry . 788, 1100
\@gender 1366, 1367, 1370–1375
\@get@firstperson 1041, 1364, 1370–1375
\@glsadd 1188
\@gobble 33, 269, 270, 723, 734, 1102, 1139, 1141, 1145, 1146, 1174, 1176–1178, 1185, 1281–1284, 1293, 1301, 1303, 1315, 1325, 1335, 1344
\@gobbletwo 734, 1176, 1342

\@idxitem 668, 668, 674, 676, 677, 698, 1146, 1149–1151, 1165
\@ifl@t@r 1176
\@ifnextchar . . . 756, 847, 1131, 1186
\@ifstar 50, 244, 245, 407, 409, 418, 428, 442, 496, 586, 1100, 1229–1232, 1234, 1235, 1239, 1247, 1248, 1250, 1274, 1299, 1344–1346, 1350–1352
\@ifundefined 848, 1099, 1101, 1128, 1130–1132, 1229, 1236, 1287–1289, 1305, 1364, 1366, 1368–1375
\@input@ 781, 849, 1099, 1132
\@latex@error 848, 1132
\@latex@warning 1131, 1132
\@m 847, 1131
\@male@label 1041, 1366, 1368
\@namedef 1365
\@ne 302–304, 494, 1304
\@nil 221, 269, 273, 274, 408, 486, 487, 493, 499, 509, 539, 1102, 1129, 1130, 1211, 1221, 1231, 1237, 1245, 1249, 1258, 1262, 1265, 1266, 1275, 1291–1293, 1301, 1302, 1304, 1313–1315, 1317–1322, 1327, 1342–1344, 1364, 1365
\@nnil 487, 493, 539, 1102, 1130, 1141, 1238, 1246, 1248, 1251, 1263, 1266, 1277, 1293, 1313, 1315–1317, 1321, 1342, 1344
\@nx 269, 1293
\@onelevel@sanitize 593, 1128, 1197, 1279, 1281, 1284
\@onlypreamble 708, 709, 735, 847, 848, 1131, 1170, 1171, 1176, 1184
\@org@dtl@paren@start . 1340, 1341
\@org@dtl@person@comma 1340
\@org@dtl@place@comma 1340
\@org@dtl@subject@comma . . . 1340
\@people@list 1041, 1364–1367, 1369
\@percentchar 294
\@person@datatoolsty 1037, 1038, 1039
\@person@label 1365
\@prev@location 1197
\@prsn@post 1365
\@prsn@pre 1365
\@remove@person 1365
\@removeperson 1365
\@roman 1183, 1184
\@s@DTLnewdbentry . . . 368, 433, 433
\@sDTLaddcolumn 418, 1232
\@sDTLforeach 496, 501, 566, 786, 787, 813, 818, 1086, 1088, 1092, 1094, 1100, 1101, 1112, 1113, 1200, 1201, 1206, 1218, 1250, 1253, 1261, 1264, 1267–1269, 1271–1274, 1280, 1281
\@sDTLforeachbibentry . 787, 1100
\@sDTLifEndsWith 245, 245, 253
\@sDTLifStartsWith . . 245, 245, 253
\@sDTLifSubString . . . 244, 244, 252
\@sDTLifclosedbetween 247, 247, 255, 1350, 1351, 1354
\@sDTLifeq . . 243, 244, 252, 1346, 1354
\@sDTLifgt . . 242, 242, 251, 1345, 1353
\@sDTLifhaskey 369, 373, 405, 405, 406, 409, 418–420, 423, 428, 439, 440, 442, 597, 627, 1229–1232, 1234, 1237–1239, 1268–1271, 1273
\@sDTLiflt . . 240, 240, 251, 1344, 1353
\@sDTLifopenbetween 249, 250, 255, 1352, 1353, 1355
\@sDTLifstringclosedbetween 246, 246, 248, 1350, 1351
\@sDTLifstringeq 242, 243, 244, 1346
\@sDTLifstringgt 241, 241, 242, 1345
\@sDTLifstringlt . . . 239, 240, 1344
\@sDTLifstringopenbetween 248, 248, 250, 1351–1353
\@sDTLnewdbentry 432, 432, 434, 781, 1235, 1288
\@sDTLnewrow 403, 404, 434, 781, 1229, 1288
\@sDTLsetheader 428, 429, 434, 628, 1234
\@sDTLsort 586, 586, 1274
\@sdtl@getcolumnindex 370, 406, 443, 566, 1229, 1230, 1239, 1267–1269, 1271, 1272, 1274, 1276, 1277
\@sdtlforcolumn 491, 492, 1247, 1268–1271, 1273
\@sdtlforcolumnidx . 492, 492, 1248
\@sdtlgetdatatype 409, 409, 440, 441, 1231, 1238
\@sdtlgetkeydata . . . 442, 443, 1239

\@sdtlgetkeyforcolumn 407, 407, 1230, 1241
\@secondoftwo 33, 734, 1143, 1146, 1176, 1177, 1201–1203
\@sgDTLforeachbibentry 788, 1100, 1101
\@tabacckludge 734, 1176
\@tempswafalse 847, 1131, 1162
\@tempswatrue 847, 1131, 1162
\@text@composite@x 734, 1176
\@thisperson . 1365–1367, 1369–1375
\@undefined 780
\@xfor@nextelement 1141
\@xp 269, 270, 1293
\[..... 213
\\ 83–86, 316, 593, 608, 1033, 1224, 1261
\␣ 33, 212, 213, 244, 608, 642, 643, 792, 795, 818, 847, 1072, 1102, 1106–1109, 1127, 1131, 1313, 1314, 1328, 1334, 1342, 1343
\] 213
\^ 213, 315, 1225
_ 4–6, 9, 26, 28–30, 33–41, 44, 46, 47, 49, 51–55, 57, 58, 62, 66, 69–81, 86–110, 113, 119–133, 135, 138–148, 187–189, 194–200, 205–210, 213, 214, 216–219, 222–231, 233–237, 239–241, 243, 245, 246, 248, 249, 263, 264, 268, 272–275, 314, 315, 321, 325–327, 329–351, 353–366, 368–374, 376–402, 404, 410–431, 433, 437–441, 443, 445–486, 488, 489, 491, 492, 496–501, 503, 507–509, 511–513, 519, 520, 522, 525, 526, 528–531, 533, 536–539, 541–557, 559, 561–563, 565, 568–585, 587–599, 602–623, 625, 627–636, 638–643, 647, 649–651, 653, 655, 656, 658, 659, 662, 668–672, 674, 675, 677, 698–703, 705–713, 717, 718, 720, 722–729, 731, 732, 734–744, 746–770, 774, 777–779, 781, 782, 786, 787, 790, 793, 797–799, 812, 813, 815, 841–847, 857, 860–866, 870, 872, 876–882, 884–889, 893, 895–904, 906, 907, 910–912, 915–919, 922–929, 941–943, 960, 961, 963–976, 980, 982, 985, 988, 990, 993, 996, 998, 999, 1001, 1003–1007, 1009, 1016–1018, 1020–1025, 1027–1032, 1038–1047, 1052, 1053, 1058–1068, 1070–1074, 1080, 1176, 1290
\| 69, 213, 239
\~ 212, 213, 244, 245, 608

Numbers

\1 ..... 97, 98, 100, 105–108, 214, 215, 593, 594, 631, 640
\2 ... 97, 98, 100, 105–108, 214, 215, 239
\3 ..... 97, 98, 100, 105–108, 214, 215
\4 ..... 97, 98, 100, 105–108, 214, 215
\5 ..... 97, 98, 100, 105, 106, 108, 214
\6 ..... 98, 105, 108

A

\A .. 56, 57, 59–61, 93, 122, 150–157, 166–169, 210, 238, 267, 314, 315
\AA 733, 1175
\aa 733, 1175
\Acr 772, 1191
\acr 771, 772, 1191
\acronymfont 722, 723, 745, 746, 1177, 1181
\Acrpl 772, 1191
\acrpl 771, 772, 1191
\active 1285, 1286, 1289, 1290
\add@accent@ 734, 1176
\addfemalelabel 1042, 1364
\addmalelabel 1042, 1364
\addtocounter 1365
\AddToHook 9
\addtolength 1087, 1088, 1091, 1093, 1094, 1096, 1098, 1147, 1148, 1151, 1152, 1154–1156, 1160, 1161, 1214, 1216
\AddTrackedRegion 267
\advance 259–261, 269, 302–304, 308, 309, 487, 488, 490, 491, 495, 1094, 1096, 1102–1105, 1112, 1113, 1129, 1144, 1151, 1163, 1165, 1184, 1186, 1189, 1195, 1205, 1229, 1234, 1245–1247, 1249, 1250, 1252–1254, 1266–1270, 1275, 1276, 1285–1288, 1294, 1295, 1303, 1304, 1322, 1357, 1358, 1362, 1363

\AE 733, 1175
\ae 733, 1175
\aftergroup 614, 1281, 1283
\afterpage 1187
\alph 1128
\alpha 732, 1174
\alsoname 663
\and 1218, 1267–1269, 1271, 1272
\andname 49, 733, 779, 1068, 1099, 1102, 1106, 1176, 1298, 1368
\AnyTrackedLanguages 268, 850, 1082
\appto 31, 1188, 1192, 1193, 1342
\arabic 494, 499, 661, 788, 1101, 1133, 1249, 1251, 1254
\AtBeginDocument ... 752, 781, 1305
\AtEndDocument 654, 1138

B

\baselineskip 855, 988, 998, 1087, 1088, 1091, 1093, 1094, 1098, 1215, 1216
\begin 270, 519, 533, 666, 670–673, 679–682, 684, 686–689, 691, 693, 772, 783, 813, 819, 840, 849, 886, 890, 904, 919, 920, 938, 975, 976, 1008, 1033, 1088, 1094, 1095, 1099, 1112, 1113, 1127, 1132, 1147, 1148, 1152, 1153, 1155–1158, 1160, 1162, 1193, 1200, 1201, 1213, 1219–1221, 1261, 1263, 1293
\begin@stack 269, 270, 1293
\begingroup 32, 269, 315, 1127, 1225, 1285, 1286, 1289, 1290, 1293, 1294
\beta 732, 1174
\bfseries 848, 1131
\bgroup .. 614, 625, 709, 710, 1138, 1144, 1146, 1149–1151, 1154, 1156, 1159, 1161–1166, 1170, 1171, 1177, 1184, 1185, 1188, 1189, 1193, 1200, 1211, 1282, 1283
\bibcite 814, 1112
\bibdata 781, 848, 1099, 1132
\bibitem 814, 819, 840, 1112, 1114, 1127
\bibliographystyle 780, 1098
\bibstyle 848, 1132
\bool 3–7, 9–12, 19–23, 27, 29, 30, 33, 35, 36, 39, 45, 46, 64–68, 72, 95, 96, 103, 116, 124, 143–146, 191, 194, 195, 210, 235, 236, 317, 318, 320–322, 324, 326–329, 337, 362, 364, 366, 391, 392, 394, 399, 400, 404, 416, 417, 419, 421–427, 429, 431, 439–441, 449, 454, 456, 467–472, 498, 499, 518, 520, 528, 534, 535, 541, 543–545, 548, 550, 552–554, 556, 558, 560, 561, 567, 569–572, 577, 579, 580, 582, 588–590, 593–596, 599, 600, 602–605, 627, 628, 632, 634, 635, 637–640, 649–651, 653, 654, 662, 665, 666, 670, 672, 679, 680, 686, 687, 694, 700, 703, 704, 712, 745, 773, 779, 780, 849, 876, 877, 879, 882, 891, 893, 899, 901, 933, 934, 944–946, 964, 969, 977, 979, 983–985, 988–990, 993–995, 999, 1000, 1004–1006, 1009, 1019, 1020, 1039, 1047–1052, 1054
\boolean 495, 496, 501, 783, 787, 812, 818, 840, 849, 1085, 1091, 1099–1101, 1112, 1113, 1127, 1132, 1193, 1200, 1206, 1210, 1249, 1250, 1253, 1267–1269, 1271–1273
\boolfalse 269, 1291
\booltrue 269, 1291
\box 25, 26

C

\c 6, 7, 14, 15, 27–29, 36–39, 41, 42, 54, 56, 57, 62–70, 72, 75, 79–84, 86–91, 93–100, 102, 103, 105–124, 131–135, 139, 143, 144, 146, 147, 149–157, 178, 184, 193–195, 198, 206, 207, 210, 212–215, 222, 226–231, 234–236, 244, 316, 320–324, 326, 327, 329, 333–338, 342, 344, 349, 355, 365, 367, 368, 370, 375–377, 379, 381, 383, 385, 386, 392, 407, 411–414, 421–427, 429, 430, 433, 437, 438, 444, 445, 450, 452–461, 463–467, 469, 474, 475, 477, 479, 482, 484, 485, 489, 496, 498, 501–504, 506, 508, 509, 511, 512, 514, 515, 518–521, 527, 528, 530–532, 534–536, 541, 543, 545, 547, 550–552, 554, 555, 557, 558, 560, 562, 569, 572, 577, 579, 581–584,

589–591, 593, 601, 609–620, 622, 625, 629, 630, 634–638, 642, 643, 646, 652, 674, 675, 678, 679, 681, 683, 685, 686, 688, 690, 691, 696, 700, 710, 711, 724–729, 732, 734–737, 739, 741, 743, 749, 752, 755, 756, 765, 774, 775, 777, 780, 781, 793, 794, 797, 798, 801–805, 810, 811, 842, 843, 845, 846, 855–858, 860–864, 869, 874, 875, 877–883, 888–894, 900, 903, 907, 910, 912, 916, 918, 920–923, 925–928, 932, 943, 944, 948, 957–959, 965, 970, 971, 973, 984–986, 990, 991, 994, 996, 997, 1001, 1002, 1009–1015, 1019, 1020, 1024, 1026, 1028, 1029, 1031, 1032, 1034, 1035, 1040, 1041, 1043–1050, 1055–1061, 1069
\c@DTLbarroundvar 859, 867, 889, 1087, 1091, 1093, 1097
\c@DTLbibrow 1101
\c@DTLmaxauthors 794, 1103
\c@DTLmaxeditors .. 794, 1103, 1104
\c@DTLpieroundvar .. 914, 918, 1200
\c@DTLplotroundXvar 948, 971, 1214, 1216
\c@DTLplotroundYvar 948, 974, 1213
\c@DTLrow 1250, 1253
\c@people 1051–1053, 1057–1061, 1369–1375
\c@person 1369, 1370
\capitalisewords 745, 1181
\caption 534, 535, 1263, 1264
\catcode 273, 315, 1225, 1285, 1286, 1289, 1290
\cB 210
\centering 673, 1148
\chapter 667, 695, 1145, 1163
\chaptername 800, 1106
\char 32, 53, 66, 69, 187, 628, 782
\chi 733, 1175
\Children 662, 666, 667, 726, 1038, 1142, 1144, 1145, 1172
\children 1038
\citation 847, 848, 1131
\cite 805, 1109
\clist 2, 4, 8, 9, 39, 41, 49, 265–267, 285, 288–290, 307, 310–312, 320–322, 351–353, 361, 371, 372, 375, 383, 390, 394, 395, 542, 545, 546, 555, 559, 562, 567, 568, 570, 571, 587, 600, 660–662, 698, 699, 712, 718–720, 726, 727, 729, 735, 744, 760, 762, 763, 773, 778, 790, 792, 793, 797, 798, 843, 844, 847, 849, 854, 868, 879, 880, 885, 901, 910, 927, 928, 931, 940, 941, 945, 952, 953, 957–959, 969, 972, 1017, 1018, 1021–1023, 1037, 1042–1044, 1049, 1053, 1058–1063
\closein 1289
\closeout 1280, 1281, 1283, 1284
\cM 239
\cO 314–316, 631
\color 863, 912, 1084, 1088, 1095, 1199, 1217, 1218
\count@ 302–304, 1186, 1195, 1303, 1304, 1343
counters:
DTLbarroundvar 859
DTLbibrow 788
DTLgidxChildCount 661
DTLmaxauthors 792
DTLmaxeditors 794
DTLpieroundvar 910
DTLplotroundXvar 937
DTLplotroundYvar 937
people 1039
person 1040
\cS 631
\cs 3, 4, 8, 10–15, 17, 18, 21, 25–30, 33–36, 39–41, 44, 45, 51–54, 57–63, 67, 69, 70, 73, 74, 76–80, 86–90, 93, 95, 100–104, 107, 110, 112, 114, 115, 119, 122–140, 147–149, 157, 159–161, 163, 164, 166–172, 174–203, 210–212, 214, 216, 223–225, 229, 234–237, 264–268, 273–275, 314, 321, 325–327, 329–341, 343, 345–351, 353–357, 359–362, 365, 369–374, 382, 386, 387, 389, 391–394, 396–402, 410–417, 419, 420, 423, 425, 428, 430, 431, 438, 440, 441, 444–452, 454, 458–461, 463, 465, 466, 471–476, 478–486, 488, 489, 491, 492, 496, 497, 501, 511–513,

519, 520, 522, 523, 525, 526, 529–531, 533, 536, 541, 542, 545, 548, 550, 552, 553, 555, 559, 562, 565, 568, 570–576, 578, 580–583, 585, 586, 588–599, 602–610, 612, 613, 616–621, 625, 627–632, 634, 636, 639–641, 647, 649–651, 653, 655, 659, 667–674, 676, 695, 699–701, 705, 709, 711, 712, 720, 722–724, 726, 728, 729, 731, 735, 736, 740, 741, 744–747, 751–753, 756–759, 766, 776, 777, 779, 782, 783, 786, 787, 790, 791, 798, 806, 807, 815, 843–846, 857, 860–863, 865, 866, 876–882, 884–888, 895, 896, 898–904, 906, 907, 910–912, 915–919, 922–929, 938, 941, 943, 960–964, 966–976, 980, 982, 988, 993, 998, 999, 1003, 1009, 1016–1018, 1020–1025, 1027, 1028, 1030–1032, 1034, 1037, 1038, 1040, 1041, 1044–1047, 1051–1053, 1055–1064, 1069–1074
\csdef 187, 190, 470, 1180
\csedef 187, 420, 422, 425, 1287
\csgdef 469, 623
\csgundef 1052–1054
\cslet 1178, 1180
\csname 3, 260, 261, 269, 400, 407–409, 486, 487, 489, 490, 492, 495, 498, 502, 503, 505, 507, 510, 540, 614–616, 623, 643, 645, 734, 848, 1083, 1084, 1088, 1094, 1099–1101, 1109–1111, 1127, 1128, 1130–1132, 1139, 1173, 1174, 1176, 1177, 1183, 1184, 1188, 1197–1201, 1204, 1205, 1226–1231, 1233–1236, 1239–1241, 1244–1259, 1261, 1262, 1265–1267, 1276, 1282–1285, 1287–1290, 1293, 1299, 1313, 1314, 1358, 1364–1375
\csundef 1052–1054
\csuse 186, 192, 264, 494, 655, 747, 757, 764, 775, 1052, 1053, 1138, 1141, 1148, 1165, 1179, 1181, 1186, 1188, 1194, 1195
\csxdef 419, 422, 425, 1171
\CurrentLocation 653, 654, 726, 1172, 1196, 1197
\CurrentOption 319, 660, 780, 852, 908, 930, 1039
\CurrentTrackedRegion .. 263, 264
\CurrentTrackedTag 264

D

\d 56, 57, 59–61, 122, 123, 150–156, 238, 700, 846
\databib 783, 784, 791, 822, 825, 826, 831, 844, 845
\datagidx 668, 675, 682, 689, 692, 696, 745
\datagidx@add@term 736, 1178, 1180
\datagidx@addchild 744, 1179, 1181
\datagidx@anchorcount 751, 1183, 1184, 1186, 1187, 1189
\datagidx@aux@usedentry 758, 761
\datagidx@bothoftwo 722, 731, 1174, 1177
\datagidx@catsep 697, 1164
\datagidx@child 660, 1181
\datagidx@childsep 694, 1162, 1163
\datagidx@clearlocationformat 752, 1166, 1184
\datagidx@columns 646, 1132, 1193, 1194
\datagidx@compositor 651, 1137, 1166
\datagidx@count 766, 1189
\datagidx@current@format 699, 701–703, 705, 1165–1168
\datagidx@current@location 701–703, 705, 1166–1168
\datagidx@current@locationformat 703–705, 1166–1168
\datagidx@current@locationstring 699, 701, 702, 705, 1165, 1166, 1168
\datagidx@current@prefix 701–703, 705, 1166–1168
\datagidx@current@target 698, 701, 705, 1165, 1166, 1168
\datagidx@defaultdatabase 707, 1169, 1171, 1177, 1193
\datagidx@dispentryval 1185
\datagidx@displaychild 666, 667, 1144, 1145
\datagidx@do@highopt@optimize 654, 659, 1138, 1139
\datagidx@do@highopt@update 656, 758, 1139, 1187

\datagidx@do@optimize@sort 652, 653, 1137, 1138
\datagidx@do@sort 652, 654, 659, 775, 1137–1139, 1194
\datagidx@do@usedentry 1187
\datagidx@docomplist . . 1137, 1166
\datagidx@doforeachentry . . 1196
\datagidx@doifdisplayed 678, 683, 685, 690, 692, 777, 777, 1151, 1155, 1156, 1160, 1161, 1196, 1197
\datagidx@doiflocwidth 671, 1147, 1149
\datagidx@doifsymlocwidth 670, 1147, 1149
\datagidx@doifsymwidth 672, 1148
\datagidx@dowrite 1196
\datagidx@endgetgroup . 1195, 1196
\datagidx@escapelocation 752, 1183, 1184, 1188
\datagidx@escapelocationformat 752, 1166, 1184
\datagidx@foreachchild 662, 666, 1133, 1144
\datagidx@format . . 755, 1186, 1188
\datagidx@formatanchor 751, 1183, 1184, 1186, 1187, 1189
\datagidx@formatlocation 699, 1165, 1168, 1169
\datagidx@formatsymdesc 649, 1136, 1140, 1192
\datagidx@getgroup 776, 1195, 1196
\datagidx@getlocation 701, 1165, 1166
\datagidx@getlocdo 700, 1166
\datagidx@heading 667, 1145, 1170, 1171, 1193, 1194
\datagidx@highopt@newgidx 655, 708, 1138, 1170
\datagidx@highopt@newterm 655, 743, 1139, 1180
\datagidx@id . . 661, 1186, 1187, 1189
\datagidx@indexfilename 708, 1170
\datagidx@invert 722, 732, 776, 1174, 1177, 1195
\datagidx@item@body 674–676, 679–682, 686–690, 1149, 1150, 1152–1154, 1156–1159
\datagidx@label 1182, 1183, 1186, 1188–1190
\datagidx@labellist 1194
\datagidx@level 666, 676, 677, 683, 696–698, 777, 778, 1144, 1150, 1151, 1154, 1163–1165, 1196, 1197
\datagidx@list 661, 1166, 1182, 1183
\datagidx@loc 661, 1188
\datagidx@location@format 704, 706, 1168, 1169
\datagidx@location@sep 699, 705, 706, 1166, 1168, 1169
\datagidx@location@start 701, 1166, 1168, 1169
\datagidx@location@startval 704, 706, 1168, 1169
\datagidx@location@target 704, 706, 1168, 1169
\datagidx@mac . . 723, 730, 1174, 1178
\datagidx@markparent 757, 1186, 1187
\datagidx@multicols 667, 1145, 1170, 1171, 1193, 1194
\datagidx@namenum 724, 730, 1174, 1178
\datagidx@newgidx 707, 708, 710, 1170, 1171
\datagidx@newgidx@update 708, 709, 711, 711, 1170, 1171
\datagidx@newsortedlist 1142, 1143
\datagidx@newterm 734, 742, 1176, 1180
\datagidx@optimize@sort 652, 659, 1137, 1139
\datagidx@orgfield 1177
\datagidx@paren 723, 732, 1174, 1178
\datagidx@parent . . 661, 1186, 1187
\datagidx@parentdatabase . . . 661
\datagidx@parse@format@label@ 755, 1185
\datagidx@parse@formatlabel 755, 764, 767–770, 1185, 1188, 1190
\datagidx@parse@formatlabel@ 756, 1186
\datagidx@parse@location 699, 702, 1166
\datagidx@particle 724, 731, 1174, 1178
\datagidx@person 723, 731, 1174, 1177
\datagidx@place 723, 732, 1174, 1177
\datagidx@postend 696, 773, 775, 776, 1163, 1193–1195

\datagidx@postheading 667, 1145, 1170, 1171, 1193, 1194
\datagidx@prestart 772, 1163, 1193, 1194
\datagidx@prev@format 699, 702–707, 1166–1169
\datagidx@prev@location 705, 1166–1169
\datagidx@prev@locationformat 699, 703–705, 1166–1168
\datagidx@prev@locationstring 699, 702, 704–707, 1166, 1168, 1169
\datagidx@prev@prefix 699, 703, 705, 1166–1168
\datagidx@prev@target 699, 704–707, 1166, 1168, 1169
\datagidx@rank . 724, 731, 1174, 1178
\datagidx@rowcount 1189
\datagidx@saint 723, 731, 1174, 1178
\datagidx@save@loc 764, 1188, 1196
\datagidx@sep 661, 1141
\datagidx@setchildsort 663, 1133, 1192
\datagidx@setchildstyle 647, 1133, 1140, 1192
\datagidx@setfieldvalues 735, 1176, 1180
\datagidx@setlocation 648, 1134, 1140, 1191
\datagidx@setnamecase 647, 1133, 1140, 1192
\datagidx@setpostdesc 647, 1134, 1140, 1191
\datagidx@setprelocation 647, 1134, 1140, 1191
\datagidx@setsee 648, 1135, 1140, 1192
\datagidx@showgroups . . . 707, 712, 719, 720, 775, 1170, 1171, 1194
\datagidx@sort . 668, 1137–1139, 1145, 1170, 1171, 1180, 1194
\datagidx@sort@foreachchild 662, 1133, 1143
\datagidx@sortchildren 662, 1142, 1143
\datagidx@sortedlist . . 1142, 1143
\datagidx@style 707, 712, 719, 775, 1170, 1171, 1194
\datagidx@style@align . 685, 1156
\datagidx@style@dict . . 695, 1163
\datagidx@style@gloss . 692, 1160
\datagidx@style@index 673, 1149, 1151
\datagidx@style@indexalign 677, 1151
\datagidx@subcapsep . . . 697, 1164
\datagidx@subcatsep 697, 698, 1164
\datagidx@subject 723, 732, 1174, 1177
\datagidx@target 752, 1184, 1186, 1188, 1189
\datagidx@thisidx 1142, 1173
\datagidx@thislabel 1183
\datagidx@title . . . 661, 1193, 1194
\datagidx@tmp 1188
\datagidx@unsort@foreachchild 662, 1133, 1144
\datagidx@usedentry 759, 759, 1187
\datagidx@value 1142, 1173, 1182, 1190
\datagidx@write@usedentry 753, 758, 758, 1184, 1187
\datagidx@xusedentry . . 758, 1187
\datagidxbalancefalse . 719, 1170
\datagidxbalancetrue 707, 719, 1170
\datagidxchildend . . . 666, 669, 676, 684, 691, 694, 697, 1145, 1146, 1150, 1155, 1160, 1162, 1164
\datagidxchilditem . . 667, 669, 676, 684, 691, 694, 697, 1145, 1146, 1150, 1155, 1160, 1162, 1164
\datagidxchildstart . 666, 669, 676, 683, 690, 694, 697, 1144, 1146, 1150, 1154, 1159, 1162, 1164
\datagidxconvertchars 722, 723, 733, 1176, 1177
\datagidxcurrentgroup 675, 676, 682, 689, 690, 693, 695, 696, 726, 777, 1150, 1154, 1159, 1161, 1163, 1164, 1195, 1196
\datagidxdb 747, 1181
\datagidxdescwidth . . 678–682, 684, 684, 685–689, 691–693, 1151–1153, 1155–1158, 1160–1162
\datagidxdictindent 695, 696, 1163
\datagidxdictparshape 696, 697, 1163, 1164
\datagidxdoseealso . 663, 664, 1141
\datagidxend 668, 675, 682, 689, 692, 696, 776, 1146, 1149, 1154, 1159, 1161, 1163, 1195

\datagidxextendedtoascii ... 733, 1175, 1177
\datagidxgetchildfields 666, 667, 729, 1145, 1173
\datagidxgroupheader 669, 675, 676, 682, 689, 690, 693, 696, 697, 1146, 1150, 1154, 1159, 1161, 1163, 1164
\datagidxgroupsep 669, 675, 676, 682, 689, 690, 692, 693, 696, 697, 1146, 1149, 1150, 1154, 1159, 1161, 1163, 1164
\datagidxhighoptfilename ... 655, 656, 708, 1138, 1139, 1170
\datagidxindent 676, 677, 683, 684, 698, 1150, 1151, 1154, 1156, 1165
\datagidxitem 669, 675, 682, 689, 693, 696, 776, 1146, 1150, 1154, 1159, 1161, 1164, 1195
\datagidxlastlabel 740, 744, 1179, 1180
\datagidxlink 665, 669, 669, 698, 705–707, 767–769, 1142, 1146, 1165, 1168, 1169, 1190
\datagidxlocalign 671, 672, 673, 679–681, 687, 689, 1147, 1148, 1152, 1153, 1157, 1158
\datagidxlocationwidth . 663, 670, 671, 674, 675, 679–681, 686–689, 713, 1136, 1147–1149, 1152, 1153, 1156–1158, 1192
\datagidxmapdata 677, 678, 683, 685, 690, 692
\datagidxnamewidth 678, 682–684, 684, 685, 690–693, 1151, 1154–1156, 1159–1162
\datagidxnewstyle 673, 673, 678, 685, 692, 695
\datagidxprevgroup 676, 682, 689, 690, 693, 696, 777, 1150, 1154, 1159, 1161, 1164, 1196
\datagidxseealsoend 664, 670, 677, 698, 1141, 1147, 1151, 1165
\datagidxseealsostart .. 664, 670, 677, 698, 1141, 1146, 1151, 1165
\datagidxsetstyle 673, 677, 775, 1148, 1194
\datagidxshowifdraft 659, 753, 1141, 1185
\datagidxstart ... 668, 674, 678, 685, 692, 695, 696, 775, 1146, 1149, 1151, 1156, 1160, 1163, 1195
\datagidxstripaccents 734, 1176, 1177
\datagidxsymalign ... 670–673, 673, 679–681, 686–688, 1147, 1148, 1152, 1153, 1156–1158
\datagidxsymbolwidth ... 663, 670–674, 678–681, 685–688, 712, 713, 1136, 1147–1149, 1151–1153, 1156–1158, 1192
\datagidxtarget 668, 669, 676, 682, 684, 690, 691, 693, 694, 697, 753, 1146, 1150, 1154, 1155, 1159, 1160, 1162, 1164, 1185
\datagidxtermkeys 726, 726, 735, 1172, 1173
\datagidxwordifygreek 722, 724, 732, 1174, 1177
\dataplot 938, 943, 961–964, 975, 1007, 1009, 1020, 1024, 1027, 1028, 1034, 1037
\datatool 3, 5, 8, 12–15, 17, 18, 20, 21, 25–30, 33, 34, 37–39, 41, 45, 46, 53–55, 57–68, 70–79, 86, 88–90, 94, 98, 104, 106, 109–114, 117, 118, 123–142, 148, 149, 157–205, 210–212, 214, 216, 225, 234, 235, 262–268, 282, 283, 286, 287, 329, 336, 376, 377, 386, 390, 392, 393, 402, 405, 430, 431, 433, 436, 439, 441, 443–445, 447, 454, 456–458, 462, 488, 503, 505, 507, 518, 531–533, 542, 544, 546, 548, 549, 556, 557, 559, 561, 563, 564, 569, 571–576, 584, 586, 587, 590, 609, 611, 613, 631, 632, 635, 637, 638, 648, 657, 658, 662, 665, 666, 668, 670–672, 678, 679, 683, 685, 686, 690–692, 699, 711, 715, 716, 727, 729, 740, 741, 743, 744, 749, 753, 754, 756, 757, 759, 761, 762, 765–770, 777, 778, 785, 789, 843, 884, 888, 889, 917, 918, 941, 942, 957–959, 964–968, 971, 972, 974, 1007, 1045, 1046
\datatool@do@load@locales .. 268, 269

\datatool@load@locales 4, 8, 268, 850, 1082
\datatoolasciiend 32, 33, 850
\datatoolasciistart . . . 32, 33, 740
\datatoolctrlboundary 31, 32, 33, 740
\datatoolcurrencysymbolprefixfmt 9, 11, 186
\DataToolDateFmt 13, 16, 20, 21, 113, 120, 133
\DataToolDateTimeFmt . 15, 119, 132
\datatoolparen 33, 237
\datatoolparenstart 33, 237, 732, 1174, 1338–1341
\datatoolpersoncomma 33, 237, 731, 850, 1174, 1338, 1340, 1341
\datatoolplacecomma 33, 237, 732, 1174, 1338–1341
\datatoolSetCurrencySort 33, 193
\datatoolsubjectcomma 33, 237, 732, 1174, 1338–1341
\DataToolTimeFmt 14, 15, 20, 21, 113, 121, 133
\DataToolTimeStampFmtSep 15, 17, 20
\DataToolTimeStampNoZoneFmt 16, 16, 20–22
\DataToolTimeStampWithZoneFmt 16, 16, 20, 21
\DataToolTimeZoneFmt 15, 16, 20, 21
\db@col@elt@end@ 398, 454, 458–461, 463–467, 481, 482, 493, 565, 618, 621, 1139, 1236, 1241–1245, 1248, 1249, 1255, 1258, 1282
\db@col@elt@w 398, 454, 458–460, 463–466, 481, 482, 493, 565, 618, 621, 1139, 1236, 1241–1245, 1248, 1249, 1255, 1258, 1282
\db@col@id@end@ . . 397–399, 408, 410–415, 450, 451, 454, 458–461, 463–467, 477, 480–482, 488–490, 493, 565, 618, 619, 621–623, 1139, 1231–1236, 1241–1246, 1248, 1249, 1255, 1258, 1282
\db@col@id@w 397, 398, 408, 410–415, 450, 451, 454, 458–461, 463–467, 477, 480–482, 488–490, 493, 565, 618, 619, 621–623, 1138, 1139, 1231–1236, 1241–1246, 1248, 1249, 1255, 1258, 1282
\db@header@id@end@ 397, 408, 410–415, 488–490, 619, 621, 623, 1231–1235, 1246, 1282
\db@header@id@w 397, 408, 410–415, 488–490, 619, 621, 623, 1231–1235, 1246, 1282
\db@key@id@end@ 397, 408, 410–415, 488–490, 619, 621, 622, 1231–1235, 1246, 1282
\db@key@id@w 397, 408, 410–415, 488–490, 619, 621, 622, 1231–1235, 1246, 1282
\db@plist@elt@end@ 397, 408, 410–415, 488–490, 619, 621, 623, 1231–1235, 1246, 1247, 1282
\db@plist@elt@w 397, 408, 410–415, 488–490, 619, 621, 622, 1231–1235, 1246, 1282
\db@row@elt@end@ 398, 448, 468, 471, 487, 539, 618, 621, 1229, 1236, 1240, 1241, 1245, 1252, 1265, 1266, 1275, 1276, 1281
\db@row@elt@w 397, 448, 468, 471, 486, 487, 539, 617, 621, 1229, 1236, 1240, 1245, 1252, 1265, 1266, 1275, 1276, 1281
\db@row@id@end@ 398, 448, 458–461, 463–468, 471, 486, 487, 539, 618, 621, 1229, 1236, 1240, 1241, 1244, 1245, 1252, 1265, 1266, 1275, 1276, 1282
\db@row@id@w 398, 448, 458–461, 463–468, 471, 486, 487, 539, 618, 621, 1229, 1236, 1240, 1241, 1244, 1245, 1252, 1265, 1266, 1275, 1276, 1281
\db@type@id@end@ 397, 408, 410–415, 488–490, 619, 621, 623, 1231–1235, 1246, 1282
\db@type@id@w 397, 408, 410–415, 488–490, 619, 621, 622, 1231–1235, 1246, 1282
\DBIBcitekey 784, 785, 787, 788, 814, 819, 840, 841, 843, 1099–1101, 1112, 1114, 1127, 1128
\DBIBentrytype 784, 785, 787, 788, 842, 1099–1101, 1128
\DBIBname 785–787, 1100, 1101

\DeclareCurrentRelease 1, 304, 313, 646, 778, 850, 908, 930, 1037
\DeclareDocumentCommand 24
\DeclareListParser 1137
\DeclareOption ... 319, 660, 780, 852, 908, 930, 1039, 1082, 1083, 1198
\DeclareOptionX 1141
\DeclareRelease 1, 304, 313, 646, 778, 850, 907, 930, 1037
\DeclareRobustCommand 1185, 1188, 1190, 1191, 1226
\def 33, 50, 221, 259, 260, 269–271, 273, 274, 299, 300, 302–305, 308, 309, 317, 318, 408, 410, 421, 454, 486–490, 493–495, 509, 511, 537–539, 608, 612, 616, 621, 625, 642, 644, 645, 655, 661, 705, 706, 732–735, 756, 788, 847, 1083–1086, 1088, 1090, 1092, 1094, 1096, 1098, 1101, 1102, 1104, 1105, 1129–1133, 1137–1139, 1141, 1142, 1149, 1152, 1153, 1156–1158, 1162, 1166, 1169, 1172–1177, 1182–1184, 1186, 1188, 1193–1196, 1198–1200, 1204, 1205, 1208, 1210, 1211, 1218, 1221, 1222, 1224–1226, 1230–1233, 1235, 1237, 1238, 1240–1242, 1244–1246, 1248–1250, 1258, 1259, 1261, 1262, 1265–1272, 1274, 1275, 1278–1284, 1290–1297, 1299–1306, 1309–1323, 1327, 1338–1340, 1342, 1344, 1347, 1348, 1350–1353, 1357–1367
\define@boolkey 1083, 1198, 1209, 1226, 1285, 1291
\define@choicekey 1084, 1098, 1139–1141, 1170, 1191, 1192, 1208–1210, 1226, 1291, 1359
\define@key ... 1084, 1085, 1140, 1170–1173, 1191–1193, 1199, 1200, 1208, 1210, 1226, 1262, 1285
\Delta 733, 1175
\delta 732, 1174
\Description . 648–651, 666, 694, 725, 1136, 1137, 1145, 1162, 1172
\detokenize 44, 187, 262, 782
\DH 733, 1175
\dh 733, 1175
\dim 2, 25, 26, 668, 670–672, 674–679, 681, 683–686, 688, 690–693, 696, 698, 785, 786, 841, 843, 853, 855, 858, 871, 872, 884, 885, 889, 891, 894–899, 904, 905, 907, 909, 913, 914, 921, 925, 929, 930, 933, 957, 961, 970, 972, 974, 982, 988, 989, 993, 998, 999
\dimen@ 1146–1148, 1150–1152, 1154–1156, 1160, 1161, 1163, 1165, 1185
\directlua 3, 291–297
\disable@keys 1086, 1092
\discretionary 753, 1185
\divide 1087, 1093, 1211, 1303
\dlt@parsewordshandler 1344
\do 259–261, 271, 300–304, 400, 401, 443, 444, 486, 489, 498, 510, 609, 611–613, 616, 617, 642, 847, 848, 1087, 1088, 1090, 1092–1094, 1097, 1101–1105, 1129–1131, 1141–1144, 1165, 1166, 1173, 1183, 1197, 1200, 1212–1217, 1227, 1228, 1239, 1245, 1246, 1251, 1254, 1259, 1261–1264, 1267–1269, 1271–1277, 1279, 1281, 1283–1285, 1289, 1290, 1294–1297, 1299, 1305, 1306, 1309–1312, 1357, 1358, 1365–1367, 1369
\do@dtlreplaceincurrentrow 1244
\do@getrow 1142, 1144, 1182, 1185–1189, 1193, 1194
\do@locrange 706, 706, 1169
\do@prevlocation .. 705, 1166, 1168
\do@update 1181
\document 848, 1131
\dotfill 656, 714, 715, 1134
\draw 895, 896, 921, 925, 977, 979–992, 994–1003, 1015, 1089, 1090, 1095, 1097, 1201, 1204, 1213–1217, 1219–1221
\dtl@A@first 1341
\dtl@A@remain 1341
\dtl@abbrvmonthname 813, 840, 1112, 1127
\dtl@addtolegend 1218
\dtl@angle 917, 1201, 1204, 1205
\dtl@argi 1315, 1317

\dtl@argii 1315, 1317
\dtl@asg@rowidx 436, 1237
\dtl@assignfirstmatch 435, 436, 1236, 1237
\dtl@authorcount 792
\dtl@B@first 1341
\dtl@B@remain 1341
\dtl@barlabel . 854, 1085, 1089, 1097
\dtl@barvar 1086, 1092
\dtl@bounds 935, 1210, 1211
\dtl@checkendsperiod 1102
\dtl@chopfirst 1091, 1095, 1097, 1214, 1216–1218, 1295
\dtl@choplast 1295
\dtl@citex 847, 847, 1131
\dtl@code 1316, 1318
\dtl@codeA 270, 1325, 1326, 1328, 1335, 1336, 1349
\dtl@codeB 270, 1325, 1326, 1328, 1335, 1336, 1349
\dtl@compare 592, 1278
\dtl@compare@ 592, 1277, 1278
\dtl@computeangles 1200, 1204
\dtl@construct@subnobrsp 1313, 1314
\dtl@constructminorticklist 1031, 1212, 1213, 1223
\dtl@constructticklist 1027, 1087, 1093, 1212, 1221
\dtl@constructticklistwithgap 1028, 1212, 1213, 1222
\dtl@constructticklistwithgapex 1032, 1223
\dtl@constructticklistwithgapz 1031, 1087, 1093, 1212, 1213, 1222
\dtl@createalphabiblabels 840, 841, 841, 1127
\dtl@cutawayoffset 909, 1198–1200, 1205
\dtl@cutlen 917, 1201
\dtl@db@col@reconstruct 625
\dtl@db@datum@reconstruct . . 625
\dtl@db@header@reconstruct . 625
\dtl@db@reconstruct@keyindex 626
\dtl@db@row@reconstruct 625
\dtl@db@value@reconstruct . . 625
\dtl@decrementrows 449, 499, 538, 540, 541, 1241, 1251, 1252, 1265, 1267
\dtl@decx . 931, 1206, 1218–1221, 1273
\dtl@decy . 931, 1206, 1218–1221, 1273
\dtl@diff 1270
\dtl@do@setkeys 1194
\dtl@do@setwordbreaks 1342
\dtl@dogetentry 510, 1236, 1242, 1243, 1255–1257, 1259
\dtl@dogetrow 540, 1180–1182, 1235, 1236, 1251, 1267
\dtl@dogetrowindex 1245
\dtl@dogetvalue 1244
\dtl@domappings 631, 642, 1286, 1290
\dtl@donext 1325–1327, 1335–1337, 1340, 1343, 1344, 1347–1349
\dtl@donextcompare 1325, 1335
\dtl@dotestiflowernext 1317
\dtl@dotestifuppernext 1316
\dtl@dx . . . 933, 1211, 1215, 1219, 1220
\dtl@dy 933, 1212, 1216, 1220
\dtl@end@if@two@octets 273, 1292, 1327, 1328, 1334, 1342
\dtl@end@ifcasechargroup 1339, 1342
\dtl@endangle 917, 1201
\dtl@endhyp 1314, 1315
\dtl@entry 433, 454, 456, 503, 505, 507, 1236, 1243, 1255–1257
\dtl@entrycr 1285, 1287–1289
\dtl@extent 853, 917, 1087, 1092, 1093, 1201
\dtl@first 273, 274, 1324, 1327, 1335, 1343, 1347, 1348
\dtl@firstA 1324–1327, 1335–1337, 1348, 1349
\dtl@firstB 1324–1327, 1335–1337, 1348, 1349
\dtl@forcolumn . 491, 493, 1247, 1248
\dtl@g@gathervalues 443, 788, 1101, 1239
\dtl@gathervalues 443, 785, 787, 1100, 1239
\dtl@get@firstthree 845, 1128, 1130
\dtl@get@yearsuffix 845, 1128, 1130
\dtl@getauthorinitial . 844, 1129
\dtl@getbounds . . . 1024, 1211, 1221
\dtl@getentryfromrow . . . 376, 377, 386, 453, 484, 882, 903, 1242
\dtl@getfirst 274, 1324, 1327, 1335, 1347, 1348

\dtl@getfirst@UTFviii 273, 1292, 1327, 1343
\dtl@getfirstthree 1129, 1130
\dtl@getsinglealphalabel .. 1129
\dtl@getvalue 457, 1244
\dtl@getyearsuffix 1130
\dtl@grabword 1341, 1342
\dtl@if@two@octets 273, 1292, 1327, 1328, 1334, 1342
\dtl@ifcasechargroup .. 1339, 1342
\dtl@ifsingle 270, 644, 1130, 1225, 1291, 1292, 1300, 1309, 1310, 1314, 1319, 1323, 1327, 1347, 1348, 1351–1353
\dtl@ifsingleorUTFviii 272, 1292, 1324, 1335
\dtl@init@write@wrap ... 608, 608
\dtl@inithyphen 1314, 1315
\dtl@initials 1313–1315
\dtl@initialscmd 1313–1315
\dtl@initialshyphen ... 1314, 1315
\dtl@initialsnext 1315
\dtl@innerlabel 910, 1198, 1200, 1204
\dtl@innernodeopt 917, 1201, 1202, 1204
\dtl@inneroffset 909, 1198, 1199, 1204
\dtl@insertinto 271, 271, 1295, 1296
\dtl@int@round 308, 1362, 1363
\dtl@int@trunc 309, 1363
\dtl@labelangle 1201, 1202
\dtl@lccode 1318
\dtl@legend 935, 1009, 1211, 1219–1221, 1224
\dtl@legendlabels 941, 1210, 1211, 1218
\dtl@legendline 1224
\dtl@legendsetting 941, 1207, 1210, 1218, 1219
\dtl@linestyle 934, 1218, 1219
\dtl@listgetalphalabel 843, 1127, 1129
\dtl@mark 933, 1217–1219
\dtl@maxx 940, 1210, 1211
\dtl@maxy 940, 1210, 1211
\dtl@mean 1269, 1270
\dtl@message 24, 267, 268, 399, 404, 433, 450, 452, 455–457, 503, 505, 508, 576, 577, 589, 595, 596, 607, 614, 626, 627, 655, 742, 784, 841, 943, 1099, 1127, 1226, 1227, 1229, 1236, 1241–1243, 1255, 1257, 1258, 1279, 1282, 1283, 1286, 1291
\dtl@midangle 917, 1201–1204
\dtl@mingap 1087, 1093, 1212
\dtl@minx 939, 1210, 1211
\dtl@miny 940, 1210, 1211
\dtl@monthname 813, 813, 819, 1112, 1114
\dtl@multibar@labels .. 1094, 1095
\dtl@multibarlabels 854, 1085, 1094
\dtl@multimidpt .. 1094, 1096, 1097
\dtl@n 1094
\dtl@nexttick 1212, 1213
\dtl@offset@x 932, 1211–1213, 1216–1221
\dtl@offset@y 933, 1212–1215, 1219–1221
\dtl@old 1086, 1092, 1200, 1201, 1204, 1205
\dtl@oldtotal 1200
\dtl@omitlines 599, 605, 632, 1285, 1286
\dtl@outerlabel 910, 1198, 1200, 1204
\dtl@outernodeopt . 917, 1202–1204
\dtl@outeroffset 909, 1198–1200, 1204
\dtl@parsewordshandler 1343
\dtl@piecutaways .. 910, 1198, 1200
\dtl@plotlinecolorlist 1211, 1217, 1218
\dtl@plotlinelist 1211, 1218
\dtl@plotmarkcolorlist 1211, 1217
\dtl@plotmarklist 1211, 1217
\dtl@post 505, 507, 1256–1258
\dtl@pre 505, 507, 1256–1258
\dtl@prevtick 1212, 1213
\dtl@rawwrite@db 1284
\dtl@rawwrite@keys 1284
\dtl@reconstruct@data .. 623, 626
\dtl@refvalue 566, 1274
\dtl@remainder 1217, 1218
\dtl@rest 273, 274, 1091, 1095, 1097, 1214, 1216, 1218, 1292, 1324, 1327, 1335, 1347, 1348
\dtl@restA 1324–1327, 1335–1337, 1348, 1349
\dtl@restB 1324–1327, 1335–1337, 1348, 1349

\dtl@row 1200
\dtl@rowi 495, 1249
\dtl@rowidx 1180, 1240
\dtl@rowii 495, 1249
\dtl@rowiii 495, 1249
\dtl@sang 1205
\dtl@saverawdbhook 1138, 1281, 1282
\dtl@scale@x .....
. 932, 1208, 1211–1213, 1216–1221
\dtl@scale@y .....
. 932, 1208, 1212–1215, 1219–1221
\dtl@segang 1205
\dtl@setcharcode .....
..... 275, 1195, 1325, 1328, 1349
\dtl@setlccharcode 1334, 1335
\dtl@setwordbreaks 1341, 1342
\dtl@setwordbreaksnohyphens
..... 1324, 1335, 1342, 1347, 1348
\dtl@sg@arg 1291, 1292
\dtl@sortdata 591, 1275
\dtl@sortlist 270, 1295
\dtl@sortresult 36,
38, 39, 46, 47, 48, 271, 579, 580,
589, 1275, 1277–1279, 1296–1298
\dtl@SortWordCommands 32, 32
\dtl@SortWordCommands@hook .
..... 32, 32, 33–35, 237, 576
\dtl@split@str 221, 1319
\dtl@splitstr 221, 1318, 1319
\dtl@srtelement ... 271, 1296–1298
\dtl@start 1200, 1201
\dtl@stream 934, 1218, 1219
\dtl@string 1313, 1314,
1324, 1335, 1341–1344, 1347, 1348
\dtl@subnobrsp 1313, 1314, 1324,
1334, 1337, 1342, 1343, 1347, 1348
\dtl@tc@arg 1315, 1317
\dtl@tc@rest 1315–1318
\dtl@test@iflowercase 1317
\dtl@test@ifuppercase . 1315, 1316
\dtl@testbibfieldcontains ..
..... 812, 812, 1111
\dtl@testbibfieldexists ....
..... 808, 808, 1109
\dtl@testbibfieldiseq .....
..... 809, 809, 1109
\dtl@testbibfieldisge .....
..... 811, 811, 1111
\dtl@testbibfieldisgt .....
..... 811, 811, 1111
\dtl@testbibfieldisle .....
..... 810, 810, 1110
\dtl@testbibfieldislt .....
..... 809, 809, 1110
\dtl@testbothnumerical .....
.. 225, 240, 242–244, 247, 249,
250, 1323, 1344–1346, 1351–1353
\dtl@testclosedbetween .....
..... 254, 254, 1354
\dtl@testcurrency .. 258, 258, 1357
\dtl@testcurrencyunit .....
..... 259, 259, 1357
\dtl@testendswith 253, 253
\dtl@testeq 251, 251, 1353
\dtl@testFPiseq . 256, 256, 257, 1356
\dtl@testFPisgt 256, 256, 1355, 1356
\dtl@testFPisgteq .. 257, 258, 1356
\dtl@testFPislt 255, 256, 1355
\dtl@testFPislteq .. 257, 257, 1356
\dtl@testFPopenbetween 1355
\dtl@testgt 251, 251, 1353
\dtl@testiceq 252, 252, 1354
\dtl@testicgt 251, 251, 1353
\dtl@testiclosedbetween ....
..... 255, 255, 1354, 1355
\dtl@testiclt 251, 251, 1353
\dtl@testiendswith 253, 253
\dtl@testifisubstring .. 252, 252
\dtl@testiflowercase .. 1317, 1318
\dtl@testiflowernext .. 1317, 1318
\dtl@testifsubstring .....
... 252, 252, 812, 1112, 1347, 1354
\dtl@testifuppercase .. 1315, 1316
\dtl@testifuppernext .. 1315, 1316
\dtl@testinlist 253, 253, 1354
\dtl@testint 258, 258, 1357
\dtl@testiopenbetween .....
..... 255, 255, 1355
\dtl@testistartswith ... 253, 253
\dtl@testlt 250, 251, 1353
\dtl@testnumclosedbetween ..
..... 254, 254, 1354
\dtl@testnumerical . 258, 258, 1357
\dtl@testnumopenbetween ....
..... 254, 254, 1354
\dtl@testopenbetween 255, 255, 1355
\dtl@testreal 258, 258, 1357
\dtl@teststartswith .....
..... 252, 252, 1348, 1354
\dtl@teststring 258, 258, 1357

\dtl@textopt . 1089–1091, 1095–1098
\dtl@this 1094
\dtl@thisbarlabel 1095
\dtl@thisdb 1217, 1218, 1273
\dtl@thisfield 1101
\dtl@thisidx 1254
\dtl@thislabel 1087, 1088, 1091, 1093, 1094, 1097, 1098, 1141, 1197, 1213–1216, 1218
\dtl@thisloc 1165
\dtl@thisplotline 1218
\dtl@thisplotlinecolor 934, 1218
\dtl@thisplotmark 1217
\dtl@thisplotmarkcolor 933, 1217
\dtl@thisrow 1254, 1267–1269, 1271, 1272
\dtl@thistick 1086, 1087, 1090–1092, 1097, 1098, 1214–1217
\dtl@thisticklabel 1214, 1216
\dtl@thisupperbarlabel 1095, 1096
\dtl@thisval 1306, 1309–1311
\dtl@ticklength . . . 933, 1214–1217
\dtl@ticlabeloffset 933, 1214–1216
\dtl@tlc 1317
\dtl@tmp 486, 1200, 1223, 1237, 1238, 1245, 1267, 1283, 1284, 1290, 1315, 1342, 1344
\dtl@tmpi 1094
\dtl@tmpii 1094
\dtl@tmplength 25, 785, 786, 843, 971, 972, 974, 1087, 1088, 1091, 1093, 1094, 1096–1098, 1100, 1128, 1147, 1148, 1154, 1155, 1163, 1213, 1216, 1223, 1291
\dtl@tmpshift . . 308, 309, 1362, 1363
\dtl@total 1200, 1204
\dtl@truncatedecimal 1201, 1202, 1302, 1304
\dtl@tuc 1315
\dtl@uccode 1316
\dtl@unit . 853, 1087, 1088, 1093, 1094
\dtl@upperbarlabel 854, 1085, 1090
\dtl@uppermultibar@labels 1094, 1095
\dtl@uppermultibarlabels 854, 1085, 1094
\dtl@variable 1088–1090, 1095, 1096, 1200
\dtl@widest . . . 785, 1100, 1127, 1128
\dtl@x 931, 1206, 1215, 1218
\dtl@xkey 931, 1208, 1211, 1218
\dtl@xlabel 940, 1210, 1215
\dtl@xminorticlist 934, 1212, 1214, 1215
\dtl@xticgap 940, 1210, 1212
\dtl@xticlabelheight 933, 1094, 1096, 1207, 1212, 1214
\dtl@xticlabels 940, 1210, 1212, 1214
\dtl@xticlist . 940, 1210, 1212, 1214
\dtl@y 931, 1206, 1216, 1218
\dtl@ykey 931, 1208, 1211, 1218
\dtl@ylabel . . . 940, 1210, 1213, 1216
\dtl@yminorticlist 934, 1213, 1214, 1217
\dtl@yticgap . . 940, 1210, 1212, 1213
\dtl@yticlabels 940, 1210, 1213, 1216
\dtl@yticlabelwidth 933, 1087, 1088, 1091, 1093, 1094, 1098, 1207, 1212, 1213, 1216
\dtl@yticlist 940, 1210, 1212–1214, 1216
\DTLabs 202, 202, 1308
\dtlabs 202, 288, 295, 302, 310, 1308, 1361, 1363
\DTLacmcs 816, 817, 818, 1113, 1114, 1127
\DTLacta 816, 817, 818, 1113, 1114, 1127
\DTLaction 320, 333, 337–339, 342, 345, 348, 349, 355, 356, 359, 365, 369, 371, 372, 374, 375, 383, 388, 389, 432, 653, 666, 740, 901, 1021–1023, 1036
\DTLadd 200, 200, 1267–1270, 1305, 1306
\dtladd 200, 201, 285, 292, 300, 306, 1086, 1092, 1094–1097, 1200, 1201, 1204, 1205, 1215, 1216, 1218–1224, 1305, 1306, 1310–1312, 1360, 1362
\dtladdalign . . . 514, 530, 1260–1262
\DTLaddall 201, 201, 1306
\dtladdall 285, 292, 300, 306
\DTLaddcolumn . 418, 1171, 1173, 1232
\DTLaddcolumnwithheader 418
\DTLaddcomma 791, 800, 803, 807, 808, 820–839, 1102, 1106, 1108, 1114–1126
\DTLaddentryforrow 509, 1258
\dtladdheaderalign 515, 530
\DTLaddperiod 791, 819–839, 1102, 1114–1126

\DTLaddtoplotlegend 1007, 1033, 1218, 1224
\dtlaftercols 513, 515, 516, 522, 1260
\DTLafterinitialbeforehyphen 217, 218, 219, 843, 1129, 1315
\DTLafterinitials 216, 219, 219, 843, 1129, 1315
\dtlafterrow 433, 444, 448, 449, 498, 538, 541, 1236, 1240, 1241, 1251, 1265, 1267
\DTLandlast 792, 793, 1102–1104
\DTLandname 23, 49, 261, 1298
\DTLandnotlast . 792, 793, 1102–1104
\DTLaposinitialpunc . 217–219, 220
\dtlappendentrytocurrentrow 454, 751, 759, 761, 763, 1186–1189, 1242
\DTLappendrow 502, 1255
\DTLappendtorow 502, 509, 1255, 1259
\DTLassign 434, 1236
\DTLassignfirstmatch 435, 436, 1236, 1237
\DTLassignfromcurrentrow . . . 437
\DTLassignlettergroup . 41, 41, 585
\dtlautokeysfalse . . 595, 600, 1285
\dtlbar@groupgap 854, 1085, 1094, 1097
\dtlbar@variables 853, 1085, 1091, 1092, 1094
\dtlbar@ylabel 854, 1085, 1087, 1091, 1093, 1098
\dtlbar@yticgap 854, 1085–1087, 1092, 1093
\dtlbar@yticlabels 854, 1085, 1087, 1088, 1091, 1093, 1097
\dtlbar@yticlist 854, 1085, 1087, 1090, 1093, 1097
\DTLbaratbegintikz 859, 886, 904, 1083, 1088, 1094
\DTLbaratendtikz 859, 887, 906, 1083, 1091, 1098
\DTLbarchart 860, 877, 880, 1084–1086
\DTLbarchartlength 855, 857, 872, 884, 1083, 1084, 1087, 1091, 1092, 1098
\DTLbarchartwidth 853, 878, 879, 895, 896, 903, 1088, 1090, 1094, 1097
\DTLbardisplayYticklabel 859, 898, 1083, 1087, 1088, 1091, 1093, 1098
\DTLbargroupindex 856, 877, 879, 906
\DTLbargrouplabelalign 857, 875, 905
\DTLbargroupwidth 902, 903, 905, 1094
\DTLbarindex 856, 887, 906
\DTLbarlabeloffset 858, 858, 871, 872, 875, 905, 1083, 1089, 1090, 1095, 1096
\DTLbarmax 857, 857, 882–885, 896, 1084, 1086, 1087, 1090–1093, 1097
\DTLbaroutlinecolor 864, 875, 891, 1084, 1088, 1095
\DTLbaroutlinewidth 864, 875, 891, 1084, 1088, 1095
DTLbarroundvar (counter) 859
\DTLbarsetupperlabelalign 865, 866
\DTLBarStyle 858, 891
\DTLbartotalvariables 852, 900–902
\DTLbarvalue 863, 864, 888, 889
\DTLbarvariable 853, 866, 877–882, 887, 899, 1085, 1086, 1088, 1089, 1092, 1094–1096
\DTLbarwidth 857, 867, 886, 889, 904, 905, 1083, 1085, 1088, 1094
\DTLbarxaxisfalse 873, 1085
\DTLBarXAxisStyle 858, 874, 895, 1083, 1090, 1097
\DTLbarxaxistrue . . 873, 1084, 1085
\DTLbarXlabelalign . . 851, 857, 870–872, 892, 1083, 1089, 1094–1097
\DTLbarXneglabelalign 851, 870–872, 892
\DTLbarXnegupperlabelalign 851, 865, 870, 894
\DTLbarXupperlabelalign 851, 865, 870, 894, 895
\DTLbaryaxisfalse 873, 1085
\DTLBarYAxisStyle 858, 874, 896, 1083, 1090, 1097
\DTLbaryaxistrue 868, 869, 873, 1084, 1085
\DTLbarYticklabelalign 851, 875, 897, 1083, 1091, 1098
\DTLbaryticsfalse 873, 1085
\DTLbaryticstrue 868, 869, 873, 1084, 1085
\dtlbeforecols 513, 514, 516, 522, 1260

\dtlbeforerow 433, 444, 448, 449, 498, 537, 538, 541, 1236, 1240, 1241, 1251, 1265, 1267
\dtlbetweencols 513, 514, 516, 522, 1260
\DTLbetweeninitials 216, 217, 219, 843, 1129, 1315
\dtlbib@style 779, 1098, 1130
\DTLbibaccessedname 806, 807
\DTLbibdatefield 788, 804, 808
\DTLBIBdbname 781, 782, 849, 1099, 1132
\DTLbibdoi 806, 807
\DTLbibdoihome 805, 806
\DTLbibdoitag 805, 806
\DTLbibeprints 806, 808
\DTLbibfield 788, 789, 790, 798–805, 807, 808, 819–839, 1101, 1102, 1106–1109, 1114–1126
\DTLbibfieldcontains ... 812, 1111
\DTLbibfieldexists 808, 1109
\DTLbibfieldiseq 809, 1109
\DTLbibfieldisge 811, 1111
\DTLbibfieldisgt 811, 1111
\DTLbibfieldisle 810, 1110
\DTLbibfieldislt 809, 1110
\DTLbibfieldlet 789, 1101
\DTLbibformatdigital 807
\DTLbibgetlongestlabel 782
\DTLbibitem 783, 814, 819, 840, 1099, 1112, 1114, 1127
\DTLbibliography 783, 1099
\DTLbibliographystyle 846, 846, 1130
\DTLbibpubmed 807, 807
\DTLbibpubmedhome 805, 807
\DTLbibpubmedtag 806, 807
DTLbibrow (counter) 788
\DTLbibsetlongestlabel 782
\DTLbibsortencap 849
\DTLbibsortname 849, 849
\DTLbibsortnamesep 849, 850
\DTLbiburl 806, 807
\DTLbiburldate 806, 808
\DTLboxfalse 938, 1209
\dtlbreak 259, 259, 260, 261, 487, 490, 493, 510, 566, 1246–1249, 1259, 1274, 1357, 1358
\dtlbst@abbrv 839, 1126
\dtlbst@alpha 840, 1127
\dtlbst@plain 818, 839, 840, 1113, 1126, 1127
\DTLcacm 816, 817, 818, 1113, 1114, 1127
\DTLcheckbibfieldendsperiod 790, 799–801, 803, 804, 819–839, 1102, 1106–1109, 1114–1126
\DTLcheckendsperiod 790, 790, 791, 793, 794, 796–798, 1102–1106, 1116, 1118, 1119, 1122
\DTLcite 847, 1131
\DTLclearbarcolors 860
\DTLcleardb 361, 399, 1227
\DTLclearnegbarcolors 861
\DTLclip 209, 210, 1313
\dtlclip 210, 287, 294, 301, 309, 1261, 1262, 1313, 1361, 1363
\dtlcol 510, 1259
\DTLcolorbarchartfalse 874, 1083
\DTLcolorbarcharttrue 851, 874, 1082
\DTLcolorpiechartfalse 915, 1198
\DTLcolorpiecharttrue 908, 915, 1198
\DTLcolumncount 329, 343, 354, 356, 357, 364, 373, 384, 390, 395, 402, 425–427, 473, 581, 582, 616, 617, 643, 1228
\dtlcolumnheader 516, 530
\dtlcolumnindex 407, 409, 420, 423, 429, 432, 436, 439, 446, 454, 456, 492, 503, 587, 590, 598, 627, 662, 666, 667, 729, 743, 744, 747, 749, 750, 752, 755–762, 764–766, 774, 775, 785, 849, 1100, 1142, 1144, 1145, 1174, 1180–1182, 1185–1189, 1193, 1194, 1230–1232, 1234, 1235, 1237, 1240, 1242, 1243, 1247, 1255
\dtlcolumnnum 365, 368, 369, 372, 387–389, 404, 419, 420, 422–425, 429, 432–434, 454–457, 482, 503, 505, 507, 508, 527, 529, 530, 598, 627, 632, 1229, 1232–1236, 1242–1244, 1255–1258
\dtlcompare 231, 238, 586, 809–811, 1110, 1111, 1274, 1324–1326, 1340, 1342, 1344–1346, 1350–1352
\dtlcompareskipcsfalse . 23, 1324
\dtlcomparewords .. 238, 1340, 1342

\DTLcomputebounds . 562, 1211, 1273
\DTLcomputewidestbibentry .. 785, 1100
\DTLconverttodecimal 104, 196–199, 206, 208–210, 963, 1086, 1088, 1092, 1095, 1200, 1206, 1218, 1273, 1300, 1305–1313, 1324, 1345, 1346, 1349–1351
\DTLcurr .. 103, 109, 188–190, 192, 639
\DTLcurrChar 186, 187
\DTLcurrCodeOrSymOrChar 10, 12, 186, 187, 192
\dtlcurrdefaultfmt 187, 190, 190, 192
\DTLcurrency .. 95, 102, 190, 190, 639
\dtlcurrencyalign 513, 514, 522, 1260
\DTLCurrencyCode 10, 108, 185, 186, 188, 189, 639
\dtlcurrencyformat 517, 532, 1260, 1262, 1264
\dtlcurrencygroup 42, 44, 262
\DTLCurrencySymbol 185
\DTLcurrencytype 408, 1231
\DTLcurrentindex 314, 499, 1251, 1254
\DTLCurrentLocaleCurrencyDP 107, 262
\DTLCurrentLocaleFormatDate 12, 13, 20, 263
\DTLCurrentLocaleFormatTime 13, 14, 20, 263
\DTLCurrentLocaleFormatTimeStampNoZone 17, 18, 20, 263
\DTLCurrentLocaleFormatTimeStampWithZone 16, 18, 20, 263
\DTLCurrentLocaleFormatTimeZone 14, 15, 20, 263
\DTLCurrentLocaleGetGroupString 41, 42, 262
\DTLCurrentLocaleGetInitialLetter 41, 210, 211, 262
\DTLCurrentLocaleGetMonthNameMap 149, 178, 262
\DTLCurrentLocaleGetTimeZoneMap 149, 178, 263
\DTLCurrentLocaleIfpmTF 150, 158, 159, 161–163, 165, 170, 172–176, 184, 262
\DTLCurrentLocaleParseDate . 145, 145, 263
\DTLCurrentLocaleParseTime . 146, 146, 263
\DTLCurrentLocaleParseTimeStamp 144, 144, 262
\DTLCurrentLocaleTimeStampFmtSep 15, 15, 20, 263
\DTLCurrentLocaleWordHandler 31, 32, 262
\dtlcurrentrow 382, 433, 444, 448, 449, 451–453, 455–457, 468, 498, 503, 505, 507, 510, 538, 565, 566, 632, 634, 785, 787, 788, 1100, 1101, 1236, 1239–1242, 1251, 1254–1257, 1259, 1265, 1267–1269, 1271, 1272, 1274
\dtlcurrentvalue 446, 1240
\DTLcurrEUR 193
\dtlcurrfmtsep 191, 192
\dtlcurrfmtsymsep ... 191, 192, 262
\dtlcurrprefixfmt 190, 190
\DTLcurrStr 186, 187
\dtlcurrsuffixfmt 191
\DTLcurrSym 186, 187
\DTLcurrXBT 192
\DTLcurrXXX 192
\DTLcustombibitem 814, 1112
\DTLcustomlegend 938, 1015
\DTLcutawayratio 909, 909, 914, 1198, 1199
\DTLdatatypecurrencyname ... 63, 64, 262
\DTLdatatypedatename .. 63, 64, 262
\DTLdatatypedatetimename ... 63, 64, 262
\DTLdatatypedecimalname 63, 64, 262
\DTLdatatypeintegername 63, 64, 262
\DTLdatatypeinvalidname 64, 64, 262
\DTLdatatypestringname 63, 64, 262
\DTLdatatypetimename .. 64, 64, 262
\DTLdatatypeunsetname . 63, 64, 262
\dtldatealign 513, 515
\dtldateformat 517, 532
\dtldategroup 42, 44, 262
\dtldatetimealign 513, 515
\dtldatetimeformat 517, 532
\dtldatetimegroup 42, 44, 262
\DTLdatumcurrency 78
\DTLdatumtoDTM 115
\DTLdatumtype 79, 116, 635

\DTLdatumvalue 78, 114, 115, 327, 376, 378, 387, 546, 549, 563, 564, 863
\dtldbcolreconstruct ... 620, 621
\dtldbdatumreconstruct 620, 622, 622, 625
\dtldbheaderreconstruct 620, 622, 625
\DTLdbLog 642
\dtldbname 333, 334, 371, 372, 386, 390, 437, 445, 447, 449, 450, 452–457, 467–471, 473, 475, 476, 484–486, 525–527, 529, 530, 536, 587, 590, 1240–1243
\DTLdbNewEntry 434, 613
\DTLdbNewRow 434, 605, 612
\DTLdbProvideData 612, 617, 624, 627
\dtldbreconstructkeyindex 617, 623, 626
\dtldbrowreconstruct 619, 621, 625
\DTLdbSetHeader 434, 613
\dtldbvaluereconstruct 622, 622, 625
\DTLdecimaltocurrency 106, 635, 638, 1304, 1306–1313
\DTLdecimaltodate 111
\DTLdecimaltodatetime 110
\DTLdecimaltolocale 105, 196, 326, 328, 635, 637, 638, 889, 971, 1302, 1306–1313
\DTLdecimaltotime 111
\DTLdefaultEURcurrencyfmt 192, 193
\dtldefaultkey ... 366, 373, 426, 479, 595, 629, 633, 1285, 1287, 1288
\DTLDefaultLocaleWordHandler 31, 45, 46
\DTLdefcurrency 11, 186
\DTLdeletedb 360, 400, 1227
\dtldisableUTFviii 269, 1291
\DTLdisplay 21
\dtldisplayafterhead 518, 520, 522, 534, 535, 1260, 1261, 1263, 1264
\DTLdisplaybargrouplabel 859, 905
\dtldisplaycr 519, 520, 531, 534, 535, 1260–1264
\DTLdisplaydb 524, 1261
\DTLdisplaydbAddBegin .. 519, 525
\DTLdisplaydbAddEnd 520, 526
\DTLdisplaydbAddItem 533
\dtldisplaydbenv 519, 519, 520, 522, 523
\dtldisplayendtab 518, 520, 536, 1260, 1262, 1264
\DTLdisplayinnerlabel 910, 925, 1198, 1204
\DTLdisplaylongdb 536, 1262
\DTLdisplaylongdbAddBegin 533, 536
\DTLdisplaylongdbAddEnd 535, 537
\dtldisplaylongdbenv 523, 533, 533, 536
\DTLdisplaylowerbarlabel 859, 887, 1083, 1089, 1094, 1097
\DTLdisplaylowermultibarlabel 859, 906, 1083, 1095
\DTLdisplayouterlabel 910, 925, 1198, 1204
\dtldisplaystartrow 518, 531, 1260–1262
\dtldisplaystarttab . 518, 519, 522, 534, 535, 1260, 1261, 1263, 1264
\DTLdisplayTBrowidxmap 519
\DTLdisplayupperbarlabel 859, 888, 1083, 1090
\DTLdisplayuppermultibarlabel 859, 907, 1083, 1096
\dtldisplayvalign 518, 519, 522, 523, 1260, 1261
\DTLdiv 202, 202, 1269, 1270, 1307, 1308
\dtldiv ... 202, 286, 293, 300, 307, 1087, 1093, 1200, 1204, 1211, 1212, 1214–1217, 1221–1223, 1307, 1310–1312, 1360, 1362
\DTLdobarcolor . 863, 864, 1083, 1084
\DTLdocurrentbarcolor 863, 1084, 1088
\DTLdocurrentpiesegmentcolor 912, 1199
\dtldofirstlocation 657, 698, 715, 1134, 1165
\dtldolocationlist 647, 657, 699, 715, 1134, 1165
\DTLdopiesegmentcolor 911, 912, 1199
\dtlenableUTFviii 269, 1291
\DTLencapbibfield 789
\DTLendbibitem 783, 785, 849, 1099, 1100, 1132
\DTLendpt 890, 1089, 1090, 1096

DTLenvforeach (env.) 495
DTLenvforeach* (env.) 495
dtlenvgforint (env.) 261
dtlenvgforint* (env.) 261
DTLenvmapdata (env.) 471
\DTLeverybargrouphook . . 865, 905
\DTLeverybarhook 865, 895, 1084, 1090, 1096
\DTLeveryprebarhook 865, 890
\dtlexpandnewvalue 317, 318, 602, 712, 1171, 1178, 1235
\dtlfallbackaction 44
\DTLfetch 446, 1240
\DTLfetchlistelement . . . 31, 1294
\DTLfmtcurr . . . 95, 103, 108–110, 190
\DTLfmtcurrency 10, 95, 103, 108, 109, 186, 190, 190, 328
\dtlforcolumn 491, 709, 727, 750, 751, 1171, 1173, 1182, 1183, 1247
\dtlforcolumnidx 492, 1247
\DTLforeach 496, 496, 502–504, 506, 508–512, 1151, 1155, 1156, 1159, 1161, 1196, 1250, 1253, 1255–1260
\DTLforeachbibentry . 783, 785, 786, 841, 849, 1099, 1100, 1127, 1132
\dtlforeachkey 400, 401, 443, 444, 489, 510, 608, 609, 611–613, 616, 617, 1227, 1228, 1239, 1246, 1259, 1261–1264, 1279, 1281, 1283, 1284
\DTLforeachkeyinrow 510, 1259, 1261, 1264, 1280, 1281
\dtlforeachlevel 494, 494, 495–513, 864, 912, 1084, 1088, 1198–1201, 1249–1259
\dtlforint 259, 1357
\DTLformatabbrvforenames 796, 839, 840, 1105, 1126
\DTLformatarticle 815, 819, 1113, 1114
\DTLformatarticlecrossref 804, 819, 1109, 1114
\DTLformatauthor 794, 814, 819, 839, 1103, 1112, 1114, 1126
\DTLformatauthorlist 794, 819, 820, 822, 823, 825, 827, 829, 831, 833, 834, 837, 839, 1102, 1114, 1115, 1117–1119, 1121–1123, 1125, 1126
\DTLformatbibentry 783, 784, 785, 849, 1099, 1100, 1132
\DTLformatbibnamelist . . 792, 794
\DTLformatbook . 815, 820, 1113, 1115
\DTLformatbookcrossref 801, 821, 824, 1107, 1115, 1117
\DTLformatbooklet 815, 822, 1113, 1117
\DTLformatbooktitle 803, 821, 839, 1108, 1115, 1126
\DTLformatbvolume 799, 821, 824, 826, 828, 835, 1106, 1116–1118, 1120, 1124
\DTLformatchapterpages 799, 824–826, 1106, 1117–1119
\DTLformatcrossrefeditor 797, 802, 1105, 1107, 1108
\DTLformatdate 804, 820, 822, 823, 825, 827–829, 831–834, 836–839, 1108, 1115–1126
\DTLformatedition 815, 819, 822, 825, 827, 831, 1113, 1114, 1116, 1118, 1119, 1122
\DTLformateditor 794, 815, 819, 840, 1104, 1113, 1114, 1126
\DTLformateditorlist 794, 803, 820, 823, 835, 1103, 1108, 1115, 1117, 1124
\DTLformatforenames 796, 814, 815, 819, 1105, 1112–1114
\DTLformatinbook 815, 823, 1113, 1117
\DTLformatincollection 815, 825, 1113, 1118
\DTLformatincollproccrossref . . . 802, 825, 827, 1108, 1118, 1119
\DTLformatinedbooktitle 803, 826, 827, 1108, 1118, 1120
\DTLformatinproceedings 815, 827, 1113, 1119
\DTLformatjr 797, 814, 815, 819, 839, 840, 1105, 1113, 1114, 1126, 1127
\DTLformatlegend 938, 938, 1015, 1207, 1219–1221
\DTLformatlist 50, 1299
\DTLformatmanual 815, 829, 1113, 1121
\DTLformatmastersthesis 815, 831, 1113, 1122
\DTLformatmisc . 815, 833, 1113, 1123

\DTLformatnumberseries 800, 821, 826, 828, 835, 1107, 1116, 1119, 1120, 1124
\DTLformatpages 800, 800, 820, 827, 828, 1106, 1114, 1120
\DTLformatphdthesis 815, 834, 1113, 1123
\DTLformatproceedings 816, 835, 1113, 1124
\DTLformatsurname 797, 814, 815, 819, 839, 840, 1105, 1113, 1114, 1126, 1127
\DTLformatsurnameonly 795, 797, 798, 1104–1106
\DTLformattechreport 816, 837, 1113, 1125
\DTLformatthisbibentry 785, 1100
\DTLformatunpublished 816, 839, 1113, 1126
\DTLformatvolnumpages 798, 820, 1106, 1115
\DTLformatvon 796, 814, 815, 819, 839, 840, 1105, 1113, 1114, 1126
\DTLgabs 202, 1308
\DTLgadd 200, 1306
\DTLgaddall 201, 1306
\DTLgcleardb 399, 401, 597, 1228
\DTLgclip 210, 1313
\DTLgdeletedb 400, 401, 1228
\DTLgdiv 202, 1308
\DTLget 331
\DTLgetbarcolor 862, 863, 864, 891, 1083, 1095
\DTLgetcolumnindex 406, 1229
\DTLgetdatatype 409, 644, 1231, 1278, 1291
\DTLgetDataTypeName 63
\dtlgetentryfromcurrentrow . 390, 433, 439, 446, 453, 453, 454, 456, 503, 505, 507, 510, 531, 566, 609, 611, 612, 666, 667, 729, 744, 749, 751, 755, 756, 758, 759, 761, 762, 766, 774, 775, 1144, 1145, 1174, 1181, 1182, 1185–1189, 1194, 1236, 1237, 1240, 1242, 1243, 1255–1257, 1259, 1274
\dtlgetentryfromrow 443, 444, 453, 453, 1239, 1242, 1267–1269, 1271, 1272, 1276, 1277
\dtlgetfirstchar 274
\DTLGetInitialLetter . . . 210, 219
\DTLgetkeydata 441, 1238
\DTLgetkeyforcolumn . . . 407, 1230
\DTLgetlocation 458, 1244
\DTLgetnegbarcolor 862, 863, 864, 890
\DTLgetpiesegmentcolor 911, 921, 1198, 1201
\dtlgetrow 389, 394, 433, 435–437, 444, 444, 445, 498, 537, 540, 609, 611, 612, 1189, 1235–1237, 1240, 1251, 1265, 1267
\DTLgetrowforkey 566, 566, 1273, 1274
\dtlgetrowforvalue 446, 1142, 1144, 1181, 1182, 1185–1187, 1189, 1193, 1240
\DTLgetrowindex 461, 1245
\dtlgetrowindex 436, 461, 462, 1180, 1237, 1240, 1245
\DTLgetvalue 457, 1244
\DTLgetvalueforkey 565, 842, 1128, 1273
\dtlgforint 260, 261, 498, 1251, 1289, 1358
\dtlgidx@checklocationchange 777, 1196, 1197
\DTLgidxAcrStyle 745, 746, 771, 1181, 1191
\DTLgidxAddLocationType 752, 1184
\DTLgidxAssignList 677, 725, 729, 777, 1172, 1173, 1196
\DTLgidxCategoryNameFont 695, 697, 1163, 1164
\DTLgidxCategorySep 695, 697, 1163, 1164
DTLgidxChildCount (counter) 661
\DTLgidxChildCountLabel 647, 661, 698, 1133, 1165
dtlgidxchildlist (env.) 666
\DTLgidxChildren 648, 666, 1135, 1144
\DTLgidxChildrenSeeAlso 648, 676, 677, 683, 684, 687–689, 691, 694, 697, 698, 1135, 1150, 1151, 1154, 1155, 1157, 1158, 1160, 1162–1165
\DTLgidxChildSep . . . 694, 694, 1163
\DTLgidxChildStyle . . 647, 647, 676, 683, 684, 690, 691, 694, 697, 1133, 1150, 1155, 1160, 1162, 1164
\DTLgidxCounter 651, 651, 757, 764, 1186–1188

\DTLgidxCurrentdb 655, 662, 666–668, 677, 729, 774, 775, 777, 1138, 1139, 1142, 1144–1146, 1151, 1155, 1156, 1159, 1161, 1174, 1193, 1194, 1196
\DTLgidxDictHead . . . 695, 696, 1163
\DTLgidxDictPostItem 695, 697, 1163, 1164
\DTLgidxDisableHyper . . 669, 1146
\DTLgidxDoSeeOrLocation 665, 674, 675, 677, 679–682, 684, 687–689, 691, 694, 698, 1142, 1149, 1151–1155, 1157, 1158, 1160, 1162, 1163, 1165
\DTLgidxEnableHyper . . . 669, 1146
\DTLgidxEndItem 670, 676, 683, 690, 694, 695
\DTLgidxFetchEntry 665, 754, 754, 768–770, 1142, 1185, 1190
\DTLgidxForeachEntry 776, 776, 1195, 1196
\DTLgidxFormatAcr 771, 771, 772, 1191
\DTLgidxFormatAcrUC 771, 772, 1191
\DTLgidxFormatDesc 648, 649–651, 1136, 1137
\DTLgidxFormatSee . . . 648, 657, 658, 663, 715, 716, 1134–1136, 1141
\DTLgidxFormatSeeAlso 648, 663, 1134, 1141
\DTLgidxGobble 722, 729, 1174
\DTLgidxGroupHeaderTitle 675, 689, 693, 695, 776, 1150, 1159, 1161, 1163, 1195
\DTLgidxIgnore 723, 730, 1138, 1174, 1178
\DTLgidxLocation 647, 657, 665, 715, 1134, 1142
\DTLgidxLocationF 705, 706, 1168, 1169
\DTLgidxLocationFF 705, 706, 1168, 1169
\DTLgidxLocationSep 705, 705, 706, 1168, 1169
\DTLgidxMac 723, 730, 1138, 1174, 1178
\DTLgidxName 722, 723, 729, 776, 1138, 1174, 1177, 1195
\DTLgidxNameCase 647, 656, 674–678, 682–685, 690–694, 697, 713, 714, 1133, 1149–1151, 1154–1156, 1159–1162, 1164
\DTLgidxNameFont 647, 656, 674–678, 682–685, 690–694, 697, 714, 1134, 1140, 1149–1151, 1154–1156, 1159–1162, 1164, 1192
\DTLgidxNameNum 724, 730, 1138, 1174, 1178
\DTLgidxNoFormat 722, 723, 729, 1174, 1177
\DTLgidxNoHeading 667, 1145
\DTLgidxOffice 722, 723, 730, 776, 1138, 1174, 1177, 1195
\DTLgidxParen 722, 723, 732, 1138, 1174, 1177, 1178
\DTLgidxParticle 722, 724, 731, 1138, 1174, 1177, 1178
\DTLgidxPlace 722, 723, 730, 776, 1138, 1174, 1177, 1195
\DTLgidxPostChild 694, 694, 1162, 1163
\DTLgidxPostChildName 647, 677, 684, 691, 694, 697, 698, 1133, 1151, 1155, 1160, 1162, 1164, 1165
\DTLgidxPostDescription 647, 649–651, 656, 714, 1134, 1136, 1137
\DTLgidxPostLocation . . . 647, 665
\DTLgidxPostName 647, 647, 656, 676, 678, 682, 685, 690, 692, 693, 697, 714, 1133, 1140, 1150, 1151, 1154, 1156, 1159, 1161, 1162, 1164, 1192
\DTLgidxPreLocation 647, 648, 656, 658, 665, 670–672, 679–681, 686–689, 714–716, 1134, 1136, 1142, 1147, 1148, 1152, 1153, 1156–1158
\DTLgidxRank 724, 731, 1138, 1174, 1178
\DTLgidxSaint 723, 731, 1138, 1174, 1178
\DTLgidxSee 648, 657, 658, 665, 715, 716, 1134, 1135, 1142
\DTLgidxSeeAlso 648, 648, 1134, 1135
\DTLgidxSeeList 663, 664, 1141
\DTLgidxSeeTagFont . 663, 663, 1141
\DTLgidxSetColumns 646, 656, 712, 1132, 1140, 1192
\DTLgidxSetCompositor 651, 651, 1137, 1140
\DTLgidxSetDefaultDB . . 707, 1169
\DTLgidxStripBackslash 729, 1174

\DTLgidxSubCategorySep 695, 698, 1163, 1164
\DTLgidxSubject 722, 723, 730, 776, 1138, 1174, 1177, 1195
\DTLgidxSymbolDescLeft 648–650, 670–673, 679–681, 686–688, 1136, 1137, 1147, 1148, 1152, 1153, 1157, 1158
\DTLgidxSymbolDescRight 648–651, 670–672, 679–681, 686–688, 1136, 1137, 1147, 1148, 1152, 1153, 1157, 1158
\DTLgidxSymbolDescription .. 648, 671, 675, 677, 681, 682, 684, 688, 689, 691, 694, 697, 698, 1136, 1148, 1149, 1151, 1153–1155, 1158, 1160, 1162–1165
\DTLgidxSymDescSep 648, 648, 649–651, 670–673, 678–681, 686–688, 1136, 1137, 1147, 1148, 1152, 1153, 1156–1158
\DTLgmax 204, 1310
\DTLgmaxall 205, 1310
\DTLgmeanforall 205, 1311
\DTLgmin 203, 1309
\DTLgminall 204, 1309
\DTLgmul 202, 1307
\DTLgneg 203, 1308
\DTLgnewdb 401, 710, 1171, 1227
\DTLground 209, 1312
\DTLgsdforall 208, 1312
\DTLgsqrt 203, 1308
\DTLgsub 201, 1306
\DTLgtrunc 209, 1313
\DTLgvarianceforall ... 207, 1311
\dtlheader 510, 1259
\dtlheaderformat 516, 516, 1260, 1261, 1263, 1264
\DTLibmjrd 816, 817, 818, 1113, 1114, 1127
\DTLibmsj 816, 817, 818, 1113, 1114, 1127
\dtlicompare 233, 238, 586, 1274, 1334–1337, 1342, 1344–1346, 1350, 1352
\dtlicomparewords . 238, 1340, 1342
\DTLidxFormatSeeItem 664, 664, 1141
\DTLidxSeeLastSep 664, 665, 1141, 1142
\DTLidxSeeSep .. 664, 665, 1141, 1142
\DTLieeese 816, 817, 818, 1113, 1114, 1127
\DTLieeetc 816, 817, 818, 1113, 1114, 1127
\DTLieeetcad 816, 817, 818, 1113, 1114, 1127
\DTLifaction 330
\DTLifAllLowerCase 220, 1317
\DTLifAllUpperCase 220, 1315
\DTLifanybibfieldexists 789, 820–839, 1101, 1106, 1115–1126
\DTLifbibfieldexists 789, 790, 794, 797–805, 807–812, 819–839, 841, 842, 1101–1103, 1105–1111, 1114–1128
\dtlifcasechargroup ... 238, 1343
\DTLifcasedatatype 228, 1323
\DTLifclosedbetween 247, 1350, 1354
\DTLifcurrency . 227, 258, 1323, 1357
\DTLifcurrencyunit 228, 259, 1323, 1357
\DTLifdate 227
\DTLifdatetime 226
\DTLifdbempty 402, 1228
\DTLifdbexists ... 333, 334, 359, 399–403, 405–407, 409, 418, 419, 428, 432, 434, 435–437, 442, 468, 491, 492, 496, 509, 525, 536, 540, 547, 551, 554, 557, 560, 586, 596, 597, 607, 614, 625, 626, 742, 743, 766, 774, 1016, 1021, 1180, 1189, 1193, 1226–1232, 1234–1237, 1239, 1247, 1248, 1250, 1253, 1258, 1266, 1268–1271, 1273, 1274, 1279, 1281, 1283
\DTLifEndsWith 245
\DTLifeq 243, 1346, 1353
\DTLiffirstrow 497, 501, 511, 1250, 1252–1254, 1259, 1261
\DTLifFPclosedbetween 230, 250, 1350, 1353
\DTLifFPopenbetween 231, 250, 1212, 1221, 1351, 1353, 1355
\DTLifgt 241, 1345, 1353
\DTLifhaskey 404, 1229, 1274
\DTLifinlist 31, 253, 1261–1264, 1293, 1354, 1364
\DTLifint 226, 258, 700, 1132, 1323, 1343, 1357

\dtlifintclosedbetween 27, 1195, 1292, 1343
\dtlifintopenbetween ... 26, 1292
\DTLiflastrow 497, 501, 511, 1250, 1252–1254, 1259
\DTLiflt 240, 1344, 1353
\DTLifnull 441, 566, 778, 1101, 1134, 1135, 1142, 1144, 1151, 1155, 1156, 1159, 1161, 1166, 1186, 1187, 1189, 1196, 1197, 1238, 1274, 1277, 1281
\DTLifnullorempty 441, 1238
\DTLifnumclosedbetween 230, 247, 254, 1349, 1351, 1354
\dtlifnumclosedbetween . 250, 284, 292, 299, 306, 1353, 1360, 1362
\DTLifnumeq 230, 243, 244, 1346
\dtlifnumeq 230, 256, 284, 291, 298, 299, 300, 305, 1202, 1203, 1223, 1301, 1302, 1304, 1346, 1356, 1359–1361
\DTLifnumerical 225, 226, 247, 249, 250, 258, 799, 800, 1106, 1107, 1322, 1357
\DTLifnumgt 230, 242, 1279, 1345
\dtlifnumgt 230, 235, 256, 257, 284, 291, 298, 299, 305, 306, 1202, 1204, 1221, 1222, 1309, 1345, 1355, 1356, 1359–1361
\DTLifnumlt 230, 240, 1089, 1096, 1279, 1324, 1344
\dtlifnumlt 230, 235, 255, 257, 284, 291, 298, 299, 305, 306, 1201–1204, 1221, 1222, 1309, 1324, 1355, 1356, 1359–1361
\DTLifnumopenbetween 231, 249, 250, 254, 1351–1354
\dtlifnumopenbetween 250, 284, 292, 299, 306, 1201–1203, 1353, 1359, 1361
\DTLifoddrow 497, 501, 512, 1251, 1253, 1254, 1260
\DTLifopenbetween . 249, 1352, 1355
\DTLifreal 226, 258, 1323, 1357
\DTLifStartsWith 244, 1348
\DTLifstring ... 227, 258, 1323, 1357
\DTLifstringclosedbetween .. 245, 1350
\DTLifstringeq 242, 1346
\DTLifstringgt 44, 241, 1345
\DTLifstringlt 239, 1344
\DTLifstringopenbetween 248, 1351
\DTLifSubString 244, 1129, 1297, 1298, 1346
\DTLiftemporal 225
\DTLiftime 227
\DTLiftimestamp 226
\DTLinbooktitlefmt 795, 823
\DTLinitialhyphen 217, 218, 219, 843, 1129, 1315
\DTLinitialpunc 216–219, 219
\DTLinitials 215, 215, 1313
\DTLinnerratio 909, 909, 913, 914, 1198, 1199
\DTLinseries 795, 801
\dtlinsertinto ... 47, 48, 1296–1298
\dtlintalign 513, 514, 522, 1260
\dtlintformat 517, 532, 1260, 1262, 1264
\DTLinttype 408, 1231
\DTLipl . 816, 817, 818, 1113, 1114, 1127
\DTLisclosedbetween 254, 1218, 1354
\DTLiscurrency 258, 1357
\DTLiscurrencyunit 259, 1357
\DTLiseq 251, 1353
\DTLisFPclosedbetween 254, 1087, 1092, 1355
\DTLisFPeq 257, 1356
\DTLisFPgt 256, 1356
\DTLisFPgteq 258, 1356
\DTLisFPlt 256, 1087, 1092, 1355
\DTLisFPlteq 257, 1356
\DTLisFPopenbetween 254, 1086, 1092, 1222–1224, 1355
\DTLisgt 251, 1353
\DTLisiclosedbetween .. 255, 1354
\DTLisieq 252, 1354
\DTLisigt 251, 1353
\DTLisilt 251, 1353
\DTLisinlist 253, 1354
\DTLisint 258, 1357
\DTLisiopenbetween 255, 1355
\DTLisiPrefix 253
\DTLisiSubString 252
\DTLisiSuffix 253
\DTLislt 251, 1353
\DTLisnumclosedbetween 254, 254, 1354, 1355
\DTLisnumeq 256, 257

\DTLisnumerical 258, 1267–1269, 1271, 1272, 1357
\DTLisnumgt 256, 256
\DTLisnumgteq 257, 258
\DTLisnumlt 256, 256
\DTLisnumlteq 257, 257
\DTLisnumopenbetween 254, 254, 1354
\DTLisopenbetween 255, 1355
\DTLisPrefix 252, 1354
\DTLisreal 258, 1357
\DTLisstring 258, 1357
\DTLisSubString 252, 1354
\DTLisSuffix 253
\DTLjacm 816, 817, 818, 1113, 1114, 1127
\DTLjcss 816, 817, 818, 1113, 1114, 1127
\DTLjournalfmt 795, 805, 820
\dtlkey . . . 510, 1259, 1261, 1264, 1281
\dtllastloadeddb 599, 612, 616, 623–629, 641, 709, 1171, 1281, 1283–1285
\DTLlegendxoffset 937, 957, 1010–1015, 1207, 1219–1221
\DTLlegendyoffset 938, 957, 1009–1014, 1207, 1219–1221
\dtllettergroup 41, 44, 262
\dtlletterindexcompare 237, 1337
\DTLlistand 23, 49, 792
\DTLlistelement 31, 1294
\DTLlistformatitem 49, 50, 1298, 1299
\DTLlistformatlastsep 49, 50, 1069, 1298
\DTLlistformatoxford 49, 50, 1069, 1298
\DTLlistformatsep 49, 50, 1069, 1298
\DTLlistskipemptytrue . . 23, 1294
\DTLloadbbl 780, 1098
\DTLloaddb 641, 1286
\DTLloaddbtex 641, 1285
\DTLloadmbbl 848, 1132
\DTLloadrawdb 641, 1289
\DTLmajorgridstyle 937, 947, 981, 982, 1207, 1214
\DTLmaketabspace 315, 1225
\DTLmanualtitlefmt 795, 830
\dtlmapcurrentrow 565
\DTLmapdata 376, 382, 467, 471, 474, 482, 485, 542, 543, 546, 548, 555, 557, 559, 561, 562, 577, 677, 777, 876, 916, 942, 963
\DTLmapdatabreak 471
\DTLmapget . . . 482, 482, 484, 542, 543, 546, 548, 556, 557, 559, 561, 942
\DTLmapgetvalues . 485, 485, 542, 546, 555, 559, 562, 876, 916, 942, 963
\DTLmaprow 481, 481
\DTLmaprowbreak 482
\DTLmax 204, 204, 1272, 1273, 1309, 1310
\dtlmax 204, 205, 287, 295, 301, 310, 1086, 1092, 1222, 1273, 1309, 1310, 1361, 1363
\DTLmaxall 204, 205, 1310
\dtlmaxall 288, 295, 302, 310
DTLmaxauthors (counter) 792
DTLmaxeditors (counter) 794
\DTLmaxforcolumn 560, 1272
\DTLmaxforkeys 558, 1272
\DTLmaxX 961, 965, 967, 976, 1023, 1211–1214, 1216–1221
\DTLmaxY 961, 965, 968, 976, 1024, 1211–1215, 1218, 1219, 1221
\DTLmbibitem 814, 819, 840, 849, 1112, 1114, 1127, 1132
\DTLmbibliography 849, 1132
\DTLmeanforall . 205, 205, 1310, 1311
\dtlmeanforall . . . 288, 295, 302, 310
\DTLmeanforcolumn . 547, 1269, 1270
\DTLmeanforkeys 544, 1268
\DTLmidpt 890, 1090, 1096
\DTLmidsentencefalse 791, 804, 820, 833, 1099, 1100, 1102, 1109, 1115, 1123
\DTLmidsentencetrue . . . 791, 1102
\DTLmin 203, 203, 1086, 1092, 1271, 1272, 1308, 1309
\dtlmin 203, 204, 287, 294, 301, 309, 1273, 1309, 1361, 1363
\DTLminall 204, 204, 1309
\dtlminall 287, 294, 301, 309
\DTLminforcolumn 557, 1271
\DTLminforkeys 555, 1271
\DTLminminortickgap 936, 1017, 1022, 1207, 1223
\DTLminorgridstyle 937, 947, 980, 981, 1207, 1214
\DTLminorticklength 936, 989, 999, 1207, 1215, 1216
\DTLmintickgap 885, 936, 969, 972, 1087, 1093, 1207, 1212
\DTLminX 961, 965, 966, 976, 979, 994, 995, 1000, 1001, 1023, 1211–1221

\DTLminY 961, 962, 965, 967, 976, 978, 983, 985, 989, 990, 1023, 1211–1216, 1218, 1220, 1221
\DTLmonthname 813, 813, 819, 840, 1112, 1114, 1127
\DTLmul 201, 202, 1270, 1306, 1307
\dtlmul 202, 285, 293, 300, 307, 1094, 1200, 1201, 1204, 1205, 1211, 1212, 1215, 1216, 1218–1221, 1223, 1307, 1311, 1312, 1360, 1362
\DTLmultibarchart 860, 899, 1091, 1092
\DTLmultibibs . . 846, 847, 1130, 1131
\DTLmutlibibs 1131, 1132
\DTLneg 202, 203, 1308
\dtlneg 203, 288, 295, 302, 310, 1308, 1361, 1364
\DTLnegextent 857, 857, 858, 882–884, 896, 1084, 1086, 1087, 1090–1093, 1097, 1098
\DTLnewbibitem 781, 782, 1099
\DTLnewbibliteralitem 781
\DTLnewbibrow 781, 1099
\DTLnewcurrencysymbol 54, 55, 187, 190, 1305
\DTLnewdb 399, 401, 432, 597, 611, 625, 661, 781, 848, 1099, 1132, 1141, 1226, 1281, 1286
\DTLnewdbentry 368, 431, 611, 712, 1171, 1173, 1178, 1179, 1235, 1281
\DTLnewdbonloadtrue . . . 595, 1286
\DTLnewrow 403, 403, 432, 611, 712, 1171, 1178, 1229, 1281
\DTLnocite 848, 1131
\dtlnoexpandnewvalue 318, 318, 602, 1235
\dtlnoheaderfalse 1286
\dtlnonlettergroup 41, 44, 262
\dtlnovalue 66, 67, 68, 75, 78, 79, 86, 321, 325, 330–332, 459, 463, 465, 466, 484, 486, 510, 571, 572, 589, 590, 729, 754, 1180, 1181, 1186, 1188, 1236–1240, 1243–1245, 1255–1257, 1259, 1276, 1277
\dtlnumbergroup 42, 44, 262
\DTLnumbernull 66, 67, 68, 78, 198, 205, 206, 208, 336, 441, 484, 532, 1238
\DTLnumcompare 231, 589
\dtlnumericformat 517, 517
\DTLnumitemsinlist 31, 1261, 1262, 1294
\DTLofseries 795, 799
\DTLofseriesfmt 795, 795, 802
\DTLouterratio 909, 909, 913, 914, 1198, 1199
\dtlpadleadingzeros 28
\dtlpadleadingzerosminus . 28, 28
\dtlpadleadingzerosplus . . 28, 28
\DTLpar 320, 675, 689, 693, 1150, 1159, 1161, 1226
\DTLparse 35, 86
\dtlparsewords 238, 1343
\DTLpcite 802, 803, 805, 805, 1107–1109
\DTLperiodfalse 791, 798, 799, 1099, 1100, 1102, 1106
\DTLperiodtrue 1102
\DTLpieatbegintikz 911, 919, 1198, 1200
\DTLpieatendtikz 911, 926, 1198, 1204
\DTLpieatsegment 911, 921
\DTLpiechart 910, 926, 928, 1198–1200
\DTLpieoutlinecolor 912, 915, 921, 1199, 1201
\DTLpieoutlinewidth 912, 915, 921, 1199, 1201
\DTLpiepercent 910, 927, 1198
DTLpieroundvar (counter) 910
\DTLpievariable 909, 910, 915, 917, 926, 928, 1198, 1200
\DTLplot 943, 964, 1016, 1208, 1210, 1211
\DTLplotatbegintikz 963, 975, 1207, 1213
\DTLplotatendtikz 964, 976, 1208, 1221
\DTLplotdisplayticklabel 939, 939
\DTLplotdisplayXticklabel 939, 972, 984, 985
\DTLplotdisplayYticklabel 939, 974, 995, 996
\dtlplothandlermark 964, 975, 1208, 1213, 1221
\DTLplotheight 936, 946, 961, 1207, 1208, 1211
\DTLplotlegendname 1034, 1035
\DTLplotlegendnamesep . 1034, 1035
\DTLplotlegendsetname 1035
\DTLplotlegendsetxlabel . . . 1036
\DTLplotlegendsetylabel . . . 1036
\DTLplotlegendx 1035, 1035

\DTLplotlegendxy 1034, 1035
\DTLplotlegendxysep ... 1035, 1035
\DTLplotlegendy .. 1034, 1035, 1036
\DTLplotlinecolors .....
.... 936, 944, 945, 1005, 1006,
1017, 1022, 1206, 1208, 1211, 1217
\DTLplotlines 935, 945, 1004, 1006,
1017, 1022, 1206, 1208, 1211, 1218
\DTLplotmarkcolors .....
.... 935, 944, 945, 1004, 1005,
1017, 1022, 1206, 1208, 1211, 1217
\DTLplotmarks 935, 945, 1004, 1005,
1017, 1022, 1206, 1208, 1211, 1217
DTLplotroundXvar (counter) 937
DTLplotroundYvar (counter) 937
\DTLplotstream 963, 1206
\DTLplotwidth .....
... 936, 946, 961, 1207, 1208, 1211
\DTLpostchaptername 795, 800
\DTLpostnumbername .. 795, 801, 838
\DTLpostpagename 794, 799, 800
\DTLpostvolnum 794, 799
\DTLpostvolumename .. 795, 799, 801
\DTLpostvon 794, 796, 797
\DTLprecite 795, 802, 803, 805
\DTLPreProcessCurrencyGroup
..... 42, 43
\DTLPreProcessDateGroup .. 42, 44
\DTLPreProcessDateTimeGroup
..... 42, 43
\DTLPreProcessDecimalGroup .
..... 42, 43
\DTLPreProcessIntegerGroup .
..... 42, 43
\DTLPreProcessLetterGroup 41, 43
\DTLPreProcessNonLetterGroup
..... 41, 43
\DTLPreProcessTimeGroup .. 42, 44
\DTLproceedingstitlefmt 795, 835
\DTLprotectedsaverawdb 624, 1283
\DTLradius 909,
909, 913, 914, 921, 1198, 1199, 1201
\DTLrawmap 641, 1290
\DTLread 596, 627, 627, 641, 708
\dtlrealalign ... 513, 514, 522, 1260
\dtlrealformat .....
..... 517, 532, 1260, 1262, 1264
\DTLrealtype 408, 1231
\dtlrecombine 449,
744, 750, 757, 758, 760, 764, 765,
767, 1181, 1182, 1186–1189, 1240
\dtlrecombineomitcurrent ...
..... 449, 1241
\DTLreconstructdata 625
\DTLreconstructdatabase 617, 626
\DTLreconstructdbdata 626
\DTLremovecurrentrow .. 508, 1258
\DTLremoveentryfromrow 504, 1256
\dtlremoveentryincurrentrow
..... 452, 1242
\DTLremoventryfromrow . 504, 1256
\DTLremoverow 539, 1266
\DTLreplaceentryforrow .....
..... 506, 509, 1257, 1258
\dtlreplaceentryincurrentrow
..... 451, 453, 749, 750, 756,
760, 762, 763, 765, 766, 1182,
1186, 1188, 1189, 1241, 1242, 1244
\DTLresetLanguage 261, 282
\DTLresetpredefined 817, 819
\DTLresetpredefinedabbrv 818, 840
\DTLresetRegion 262
\DTLrmentry 473, 474, 477, 480
\DTLrmrow 472, 473, 481
\dtlroot 203, 286,
293, 301, 307, 1308, 1312, 1360, 1362
\DTLrotateinnerfalse .. 908, 1198
\DTLrotateinnertrue 1198
\DTLrotateouterfalse .. 908, 1198
\DTLrotateoutertrue 1198
\DTLround 209, 209, 1312
\dtlround 107, 209, 286,
294, 301, 308, 328, 1087, 1091,
1093, 1097, 1200, 1213, 1214,
1216, 1218, 1219, 1312, 1360, 1362
\DTLrowcount 337,
362, 392, 395, 402, 445, 468, 526,
529, 576, 587, 596, 609, 611, 612,
615, 617, 643, 766, 1189, 1228, 1274
\DTLrowincr 494
\dtlrownum .....
372, 373, 386–389, 404, 445, 447,
449, 450, 468, 471, 503, 504, 507,
525, 530, 531, 533, 537, 577, 596,
605, 628, 634, 1142, 1143, 1229,
1240, 1241, 1255–1257, 1275, 1276
\DTLrowreset 494
\DTLsavedb 624, 1279

\DTLsavelastrowcount .. 495, 1249
\DTLsaverawdb 624, 1138, 1281
\DTLsaverowcount 495
\DTLsavetexdb 624, 1281
\DTLscinum 9, 94
\DTLscp . 817, 817, 818, 1113, 1114, 1127
\DTLsdforall ... 207, 209, 1311, 1312
\dtlsdforall 290, 296, 303, 312
\DTLsdforcolumn 553, 1271
\DTLsdforkeys 552, 1271
\DTLsetbarcolor .....
.... 860, 865, 874, 875, 1083, 1084
\dtlsetcharcode 275, 275, 1328
\DTLsetcurrencydatum 91, 92
\DTLsetdecimaldatum 91, 91
\DTLsetdefaultcurrency .....
..... 186, 262, 1305
\dtlsetdefaultUTFviiicharcode
..... 276, 276, 1328
\dtlsetdefaultUTFviiilccharcode
..... 276, 279, 1328, 1331
\DTLsetdelimiter .....
..... 315, 318, 601, 1225, 1226
\DTLsetentry 472–474, 476
\DTLsetfpdatum 90
\DTLsetheader .. 428, 612, 1234, 1281
\DTLsetintegerdatum ... 90, 90, 590
\dtlsetlccharcode . 275, 1328, 1334
\DTLsetLocaleOptions 265
\DTLsetnegbarcolor 860, 875
\DTLsetnumberchars .....
..... 10, 58, 59, 262, 1299
\DTLsetpiesegmentcolor .....
.... 911, 912, 913, 915, 1198, 1199
\DTLsetseparator .....
.... 314, 315, 318, 601, 1224–1226
\DTLsetstringdatum 93
\DTLsettabseparator 315, 602, 1225
\DTLsettemporaldatum 92, 92
\DTLsetup .....
24, 116, 333, 359, 710, 774, 877,
899, 926, 930, 965, 1005, 1021–1023
\dtlsetUTFviiicharcode .....
..... 275, 276, 1328
\dtlsetUTFviiilccharcode ...
..... 276, 1328, 1334, 1343
\dtlshowdb 643, 1290
\dtlshowdbkeys 643, 1290
\DTLshowlinesfalse .....
..... 938, 947, 1207, 1208
\DTLshowlinestrue 946, 1208
\DTLshowmarkersfalse .. 947, 1208
\DTLshowmarkerstrue .....
..... 938, 946, 947, 1207, 1208
\dtlshowtype 643, 1291
\DTLshufflelist 49
\DTLsicomp .....
... 817, 817, 818, 1113, 1114, 1127
\DTLsort 586, 1274
\dtlsort 586, 586, 591, 1146, 1274
\DTLsortdata .....
.. 395, 568, 570, 575, 576–578, 668
\dtlsortdatavalue 583, 583
\DTLsortedactual 40
\DTLsortedletter 40
\DTLsortedvalue 40
\DTLsortlettercasehandler .. 46
\DTLsortletterhandler 46
\dtlsortlist 46, 1296
\DTLsortwordcasehandler 45
\dtlSortWordCommands 32, 813
\DTLsortwordhandler .. 45, 237, 572
\DTLsortwordlist 34
\dtlspecialvalue .....
..... 430, 592, 592, 709,
737, 749, 750, 756, 759–763, 765, 766
\dtlsplitrow 450, 450, 477,
480, 505, 507, 1241, 1242, 1256, 1257
\DTLsplitstring .....
.. 220, 221, 1318, 1319, 1338, 1339
\DTLsqrt 203, 203, 1271, 1308
\dtlsqrt 286, 293, 300, 307
\DTLstartangle .....
.... 909, 914, 917, 919, 1198–1200
\DTLstartpt .....
.. 889, 890, 1089, 1090, 1095, 1096
\DTLstartsentencefalse .....
..... 791, 792, 1099, 1100, 1102
\DTLstartsentencespace .....
..... 791, 792, 794, 797–805,
819–839, 1102–1109, 1114–1126
\DTLstartsentencetrue . 791, 1102
\DTLStoreInitialGetLetter ..
..... 216–219, 219
\DTLstoreinitials 215,
216, 796, 844, 845, 1105, 1129, 1314
\dtlstringalign .....
..... 513, 514, 515, 522, 1260
\dtlstringformat .....
... 516, 531, 533, 1260, 1261, 1264

\DTLstringnull .. 66, 67, 68, 336, 439–441, 484, 531, 533, 1237, 1238
\DTLstringtype 408, 1231, 1278
\DTLsub 201, 201, 1270, 1306
\dtlsub 201, 285, 293, 300, 307, 1087, 1092, 1094, 1201, 1202, 1204, 1211, 1212, 1214–1216, 1221–1223, 1261, 1262, 1306, 1311, 1312, 1360, 1362
\DTLsubstitute 220, 1305, 1318
\DTLsubstituteall 221, 642, 645, 1225, 1290, 1313, 1314, 1319, 1342, 1343
\DTLsumcolumn 543, 1268
\DTLsumforkeys 541, 1267
\dtlswapentriesincurrentrow 453, 1242
\dtlswaprows 537, 1264
\DTLtcs 817, 818, 1113, 1114, 1127
\DTLtemporalvalue 112, 113–115, 119–121, 132–134, 616, 620
\dtltexorsort 32, 33, 187
DTLthebibliography (env.) 812
\DTLtherow 494
\DTLthesistitlefmt 795, 834
\DTLthisval 1267–1273
\DTLthisX 563, 1273
\DTLthisY 563, 1273
\DTLticklabeloffset 897, 898, 936, 946, 982, 993, 998, 1091, 1098, 1207, 1214, 1216
\DTLticklength ... 897, 936, 982, 993, 1090, 1097, 1207, 1214, 1216
\dtltimealign 513, 515
\dtltimeformat 517, 532
\dtltimegroup 42, 44, 262
\DTLtocs 817, 817, 818, 1113, 1114, 1127
\DTLtods 817, 817, 818, 1113, 1114, 1127
\DTLtog . 817, 817, 818, 1113, 1114, 1127
\DTLtoms ... 817, 818, 1113, 1114, 1127
\DTLtoois .. 817, 818, 1113, 1114, 1127
\DTLtoplas . 817, 818, 1113, 1114, 1127
\DTLtotalbargroups .. 852, 902, 903
\DTLtotalbars 852, 878, 879, 882, 902
\DTLtrunc 209, 209, 1312, 1313
\dtltrunc 209, 287, 294, 301, 308, 1312, 1360, 1363
\DTLtwoand .. 792, 793, 798, 1102–1104
\dtltype 510, 1259, 1261, 1264
\DTLundLocaleHook 282
\dtlunknowntag 747, 747
\DTLunsettype 408, 1231, 1278
\dtlupdateentryincurrentrow 455, 455, 744, 1181, 1243
\DTLuse 332, 1036
\DTLusedatum 78
\DTLvarianceforall . 206, 207, 1311
\dtlvarianceforall 289, 296, 302, 311
\DTLvarianceforcolumn 551, 1270, 1271
\DTLvarianceforkeys ... 549, 1269
\dtlverbosetrue 298
\DTLverticalbarsfalse 852, 870, 1083
\DTLverticalbarstrue 851, 869, 1083
\dtlwordindexcompare 236, 1146, 1337
\DTLwrite 606, 624, 655
\DTLxaxisfalse 947, 1209
\DTLXAxisStyle 937, 947, 977, 1207, 1214
\DTLxaxistrue 937, 947, 953–955, 1207–1210
\DTLxminorticsfalse 939, 948–950, 1209
\DTLxminorticstrue 948–950
\DTLxparse 88
\DTLxsetcurrencydatum 91
\DTLxsetdecimaldatum 91
\DTLxsetintegerdatum 90
\DTLxsetstringdatum 93
\DTLxsettemporaldatum 92
\DTLxsplitstring 221
\DTLxticsfalse 947, 1209
\DTLxticsinfalse 950, 951, 1209, 1210
\DTLxticsintrue 937, 950, 951, 1207, 1209
\DTLxticstrue 939, 947–950, 953–955, 1208–1210
\DTLyaxisfalse 947, 1209
\DTLyAxisLabelStyle 854, 899
\DTLYAxisStyle 937, 947, 979, 1207, 1214
\DTLyaxistrue 937, 947, 953–956, 1207, 1209, 1210
\DTLyminorticsfalse 939, 949, 950, 1209
\DTLyminorticstrue 949, 950
\DTLyticsfalse 947, 1209
\DTLyticsinfalse .. 951, 1209, 1210

\DTLyticsintrue 937, 951, 1207, 1209
\DTLyticstrue 939, 947, 949, 950, 953–956, 1209, 1210
\DTMdisplay 17, 21, 22
\DTMdisplaydate 13, 21
\DTMdisplaytime 13, 14, 21
\DTMdisplayzone 14, 21
\DTMnow 610
\DTMsavedate 115
\DTMsavenoparsedate 117
\DTMsavetime 116, 118
\DTMsavetimestamp . . . 115, 117, 119

E

\eappto 1142, 1143, 1166, 1182, 1183, 1274
\edef 33, 269, 271, 272, 274, 292–297, 308, 309, 408, 409, 421, 486, 489, 495, 505, 507, 510, 537–540, 566, 644, 645, 847, 848, 1087–1096, 1098–1100, 1110, 1111, 1128, 1131, 1132, 1142–1144, 1154, 1170, 1171, 1179–1182, 1184–1189, 1193, 1194, 1196, 1201–1205, 1211, 1217–1219, 1222–1226, 1230, 1231, 1233–1237, 1239–1249, 1251, 1252, 1254–1257, 1259, 1261–1270, 1274–1279, 1283–1285, 1287–1289, 1293–1305, 1311–1319, 1323, 1325, 1326, 1335–1337, 1340–1343, 1347, 1348, 1351–1353, 1363–1367, 1370–1375
\editionname 783, 815, 819, 1099, 1113, 1114
\editorname 782, 794, 1099, 1104
\editorsname . . . 782, 794, 1099, 1104
\edtlgetrowforvalue 446, 446, 785, 1100, 1240
\edtlinsertinto 48, 1298
\egroup 616, 626, 709, 711, 1138, 1145, 1146, 1149–1151, 1154, 1155, 1159–1166, 1170, 1171, 1178, 1184, 1185, 1188, 1189, 1195, 1204, 1221, 1283, 1284
\else 47, 48, 102, 103, 191, 224, 228–230, 236, 240, 242–244, 247–250, 257, 259, 261, 269–273, 291, 292, 298, 301, 302, 308, 309, 407, 421, 487, 490, 492–495, 502, 510, 537, 539, 540, 595, 608, 613, 644, 773, 775, 781, 791, 848, 851, 855, 856, 865, 870–872, 886, 890, 892–894, 896–898, 904, 905, 907, 913, 1083, 1084, 1086–1098, 1101–1111, 1116, 1118, 1119, 1122, 1129, 1130, 1132, 1139, 1141, 1143, 1147–1149, 1152, 1153, 1157, 1158, 1162, 1164, 1168, 1180–1182, 1186, 1188, 1193, 1194, 1198–1200, 1202, 1205, 1211–1219, 1223–1225, 1228, 1230, 1233, 1236–1238, 1240, 1243, 1244, 1246–1260, 1262–1270, 1273, 1275–1279, 1286–1288, 1292–1305, 1313–1318, 1320–1328, 1334–1336, 1340, 1342–1353, 1356–1359, 1362, 1363, 1365–1375
\em 1106, 1107, 1109, 1115, 1117, 1121, 1123, 1124
\emph 663, 722, 723, 795, 839, 1126, 1141, 1177
\empty 301, 302, 1286
\encodingdefault 734, 1176
\end 269, 270, 520, 536, 666, 670–673, 679–682, 684, 686–689, 691, 694, 773, 783, 813, 819, 840, 849, 887, 892, 906, 926, 938, 976, 1008, 1033, 1088, 1089, 1091, 1095, 1098, 1099, 1112, 1114, 1127, 1132, 1147, 1148, 1152–1155, 1157, 1158, 1160, 1162, 1193, 1204, 1217, 1219–1221, 1262, 1264, 1293, 1315–1317
\end@dtl@doifinlist 1293
\end@dtl@getfirst 274, 1324, 1327, 1335, 1347, 1348
\end@dtl@getfirst@UTFviii 273, 1292, 1327, 1343
\endcsname 3, 260, 261, 269, 407–409, 486, 487, 489, 490, 492, 495, 498, 502, 503, 505, 507, 510, 540, 614–616, 623, 643, 645, 734, 848, 1083, 1084, 1088, 1094, 1099–1101, 1109–1111, 1127, 1128, 1130–1132, 1139, 1173, 1174, 1176, 1177, 1183, 1184, 1188, 1197–1201, 1204, 1205, 1226–1231, 1233–1236,

1239–1241, 1244–1259, 1261, 1262, 1265–1267, 1276, 1282–1285, 1287–1290, 1293, 1299, 1313, 1314, 1358, 1364–1375
\endfirsthead 535, 1263
\endfoot 535, 1263
\endgroup .. 32, 33, 270, 315, 1129, 1225, 1286, 1289, 1290, 1293, 1294
\endhead 534, 535, 1263, 1264
\endinput 614, 1281, 1283
\endlastfoot 535, 1263
\endlinechar 1285, 1286, 1290
\enspace 647, 656, 714, 715, 1134
\ensuremath 722, 723, 1177
environments:
DTLenvforeach 495
DTLenvforeach* 495
dtlenvgforint 261
dtlenvgforint* 261
DTLenvmapdata 471
dtlgidxchildlist 666
DTLthebibliography 812
\epsilon 732, 1174
\equal 566, 1179, 1274, 1323, 1369
\eta 732, 1174
\etalname 782, 793, 798, 1099, 1103, 1104, 1106
\EUR 192
\Euro 192
\euro 192, 1305
\exp 5, 9–11, 26, 29, 30, 33–35, 40–42, 46–49, 52, 53, 69–74, 76–79, 87, 88, 90–95, 97–101, 104–110, 113–121, 124–130, 132–134, 138, 143, 158, 159, 161–163, 165, 170, 172–176, 178, 184, 186–189, 191, 196, 198, 210–212, 214–219, 221, 223–229, 234–237, 239–241, 243–246, 248, 249, 264, 275, 285, 286, 288–290, 305–312, 314–316, 318, 322, 326–329, 335, 340, 347, 363, 368, 369, 371–373, 376, 377, 382, 389, 394, 396–399, 424, 430–433, 435, 439, 449, 450, 452–460, 463–466, 468, 477, 479–481, 483, 484, 486, 488, 498, 499, 503, 505, 507, 514, 519, 520, 525–531, 533–536, 541, 543–545, 548, 550, 552–554, 557, 558, 560, 561, 564, 565, 569, 572, 575–580, 583–585, 588, 589, 592, 593, 609, 611–615, 617, 619, 620, 622, 623, 625, 627, 628, 631, 635, 637–639, 643, 664, 677, 683, 696, 700–702, 709, 710, 712, 723, 724, 735, 736, 740, 743–746, 748, 750–753, 756, 759–763, 772, 773, 777, 782, 785–790, 793, 794, 796, 798–800, 805–813, 819, 840, 844, 845, 849, 855, 856, 863, 864, 878, 879, 881, 882, 886–891, 893–899, 902, 903, 905–907, 912, 921, 925, 928, 942, 957–959, 965–967, 971, 976–992, 994–1003, 1005–1008, 1015–1017, 1023, 1024, 1034, 1043, 1044, 1047–1050, 1054, 1069, 1073, 1074
\expandafter 3, 33, 260, 261, 271–274, 308, 309, 317, 407–409, 429, 486, 487, 489, 490, 492, 495, 502, 503, 505, 507, 510, 537–541, 566, 614–616, 623, 641–643, 645, 668, 709, 734, 776, 781, 847, 848, 930, 1083, 1084, 1086, 1088–1092, 1094–1098, 1101–1112, 1116, 1118, 1119, 1122, 1127–1132, 1139, 1142, 1143, 1165, 1166, 1170, 1173, 1174, 1176, 1177, 1183, 1184, 1187, 1188, 1196, 1198–1201, 1204, 1205, 1211, 1214, 1217–1219, 1222–1224, 1226–1228, 1230–1232, 1234, 1235, 1237, 1239–1259, 1261–1277, 1279–1290, 1292, 1293, 1295–1302, 1304–1306, 1309–1325, 1327, 1328, 1334, 1335, 1338, 1339, 1341–1343, 1347–1349, 1351–1353, 1358, 1362–1368, 1370–1375
\expandonce 435, 712, 1171, 1173, 1178, 1188, 1194, 1196, 1237, 1241, 1242, 1294, 1325–1327, 1335–1337, 1340–1342, 1344, 1347–1349
\ExplSyntaxOff . 3, 24, 32, 48–50, 55, 192, 250, 268, 270, 273–275, 282, 283, 290, 313, 315, 318, 407, 415, 486, 489, 492, 495, 501,

509, 513, 539, 565, 608, 623, 641, 643, 659, 660, 731, 733, 778, 780, 792, 794, 805, 816, 847, 850, 852, 907–909, 930, 1037, 1038, 1072, 1082
\ExplSyntaxOn 1, 3, 25, 32, 33, 49, 50, 56, 192, 261, 270, 272, 274, 275, 282, 284, 305, 314, 315, 319, 410, 416, 488, 491, 495, 496, 502, 511, 514, 540, 566, 608, 624, 642, 646, 660, 731, 734, 779, 780, 792, 795, 806, 818, 848, 851, 852, 908, 909, 931, 1038, 1039, 1072

F

\faBtc 192
\faEur 193
\fboxrule 938, 1207
\fcolorbox 938, 1207
\femalechild 1067, 1368
\femalechildren 1067, 1368, 1373, 1374
\femalelabels 1042, 1364, 1365
\femalename 1068, 1368, 1375
\femaleobjpronoun 1065, 1368
\femaleparent 1067, 1368
\femalepossadj 1065, 1368
\femaleposspronoun 1066, 1368
\femalepronoun 1065, 1368
\femalesibling 1068, 1368
\femalesiblings .. 1068, 1368, 1375
\fi 25, 30, 47, 48, 102, 103, 191, 224, 228–230, 236, 240, 242, 244, 247, 248, 250, 257, 260, 261, 269–273, 275, 291, 292, 298, 299, 301, 302, 308, 309, 407, 421, 422, 487, 491–495, 502, 511, 537–540, 589, 595, 609, 613, 628, 644, 668, 675, 689, 692, 693, 696, 773, 775, 776, 781, 791, 792, 813, 814, 847, 848, 851, 854–856, 865, 870–872, 886, 890, 892, 893, 895–898, 904, 905, 907, 913, 1083–1112, 1116, 1118, 1119, 1122, 1128–1134, 1136–1141, 1143, 1147–1162, 1164, 1165, 1168, 1170, 1180–1183, 1187, 1188, 1193, 1194, 1198, 1199, 1201, 1202, 1204, 1205, 1208–1219, 1221, 1223–1225, 1229, 1230, 1233, 1236–1240, 1243–1260, 1262–1267, 1269, 1270, 1273, 1274, 1276–1279, 1286–1289, 1291–1305, 1314–1318, 1320–1328, 1334–1337, 1340–1353, 1356–1360, 1363, 1365–1375
\field 735, 736, 1177
\file 607, 628, 641
\fill 921, 1088, 1095, 1201
\FirstId 726, 1172
\fmtversion 1176
\forallpeople 1061, 1062
\forcsvlist 1166
\foreachperson ... 1061, 1061, 1367
\foreachpersonbreak 1061, 1062, 1063
\ForEachTrackedDialect 267, 268, 850, 1082
\forlistloop 239
\fp 2, 28, 73, 77, 78, 90, 94, 97, 98, 112, 116, 118, 119, 121, 122, 134–136, 138, 140, 147, 148, 205–208, 231, 284–290, 309, 319, 375–383, 385–387, 541–564, 853, 854, 857, 858, 863, 864, 878, 879, 881–886, 888–890, 892–894, 896, 900, 903–907, 916–924, 929–933, 941, 958–963, 965–973, 977–1003, 1007–1015, 1017–1020, 1022, 1024–1032
\FPabs 302, 1361
\FPadd 300, 302–304, 1360
\FPclip 301, 1361
\FPdiv 300, 302–304, 1360
\FPifeq 298, 1359
\FPifgt 298, 1359
\FPiflt 298, 1359
\FPmax 301, 302, 1361
\FPmessagesfalse 1359
\FPmessagestrue 1359
\FPmin 301, 1361
\FPmul 300, 303, 304, 1360
\FPneg 302, 1361
\FProot 301, 304, 1360
\FPround 301, 1360
\FPsub 300, 303, 304, 1360
\FPtrunc 301, 1361

G

\g 495, 592, 607, 608, 628, 629, 653, 725,

726, 728, 736, 738, 751, 753, 756, 757, 765, 767, 777, 840, 841, 843, 1042–1045, 1049, 1051, 1053, 1054
\G@refundefinedtrue ... 848, 1131
\Gamma ... 733, 1175
\gamma ... 732, 1174
\gappto ... 32, 1173, 1184
\gdef ... 32, 260, 315, 487, 489, 490, 616, 653, 1127, 1138, 1173, 1195, 1225, 1245–1247, 1250, 1251, 1253, 1254, 1261, 1262, 1283, 1284, 1286, 1289, 1290, 1358, 1364–1366
\gDTLforeachbibentry .. 787, 1100
\gDTLformatbibentry ... 784, 1099
\getpersonforenames ... 1081
\getpersonfullname 1081, 1375
\getpersongender ... 1080, 1375
\getpersongenderlabel ... 1081
\getpersonname ... 1081, 1375
\getpersonsurname ... 1081
\getpersontitle ... 1082
\global ... 51, 52, 260, 261, 269, 272, 439, 487, 488, 490, 491, 505, 511, 566, 614, 615, 623, 628, 1100–1102, 1128, 1129, 1138, 1178–1180, 1184, 1196, 1197, 1227–1229, 1234, 1237–1239, 1245–1247, 1250–1254, 1256, 1258, 1259, 1261, 1264, 1265, 1267, 1274, 1276, 1282–1285, 1290, 1293, 1296, 1298, 1299, 1306–1313, 1358, 1365, 1366
\Gls ... 770, 1191
\gls ... 770, 770, 1190
\glsadd ... 722, 723, 752, 764, 1146, 1177, 1185, 1188
\glsaddall ... 766, 1189
\Glsdispentry .. 754, 771, 1185, 1191
\glsdispentry 754, 754, 771, 1185, 1191
\Glslink ... 767
\glslink ... 767, 1190
\Glsnl ... 771, 1191
\glsnl ... 770, 771, 1191
\Glspl ... 770, 1191
\glspl ... 770, 770, 1191
\Glsplnl ... 771, 1191
\glsplnl ... 771, 771, 1191
\glsreset ... 749, 750, 1182, 1183
\glsresetall ... 750, 1182
\Glssym ... 771, 1191
\glssym ... 771, 771, 1191
\glsunset ... 750, 751, 1182, 1183
\glsunsetall ... 750, 1183
\group ... 33–35, 49, 69, 70, 114, 115, 119, 187, 215, 237–239, 371, 372, 430, 525, 536, 541, 543–545, 547–555, 557, 558, 560–562, 564, 576, 578, 587, 588, 606, 607, 627, 628, 642, 643, 655, 664, 665, 668, 674, 676–678, 683–685, 690–692, 694, 696–700, 708–710, 712, 722–724, 740, 753, 764, 766–770, 773, 776, 782, 841, 843, 844, 877–879, 881, 899, 901, 902, 926–928, 1016, 1018, 1021, 1024

H

\hangindent .. 668, 682, 690, 691, 693, 1146, 1154, 1159, 1160, 1162
\hbox ... 25, 26, 848, 1131
\hfil ... 1260
\hfill ... 656, 715, 1134
\HierSort ... 726, 740
\href ... 806, 807
\hspace . 676, 677, 698, 1150, 1151, 1165
\hyperlink ... 26, 669, 1146
\hypertarget ... 26, 668, 753, 766, 1146, 1184, 1189

I

\IeC ... 734, 1176
\if ... 224, 1301
\if@datagidx@warn ... 653, 1138
\if@datagidxsymbolleft ... 649, 1136, 1147, 1148, 1152, 1153, 1156, 1157
\if@dtl@condition ... 221, 240, 242–244, 247, 249–259, 808–812, 1109–1111, 1287, 1288, 1315, 1317, 1319, 1344–1348, 1351–1357
\if@dtl@insertdone 47, 48, 48, 271, 272, 1275, 1276, 1296–1298
\if@dtl@numgrpsep ... 1320–1322
\if@dtl@sequential 700, 1166–1168
\if@dtl@utf8 ... 3
\if@endfor ... 1101, 1143, 1294, 1295, 1366, 1367
\if@endpe ... 668, 776

\if@filesw 781, 814, 847, 1099, 1112, 1131, 1132
\if@tempswa 1162
\if@twocolumn 773, 1193
\ifallfemale . 1059, 1367, 1373, 1375
\ifallmale . . . 1058, 1366, 1373, 1375
\ifblank 499, 1251, 1254
\IfBlankTF 265
\ifbool 272–275, 1292, 1327
\IfBooleanT 471
\IfBooleanTF . 239–243, 246–249, 265, 403, 405, 406, 418, 432, 461, 485, 491, 492, 525, 786, 787, 1045
\ifboolexpr 276–281, 1328–1333
\ifcase 229, 421, 813, 1084, 1112, 1133–1136, 1139, 1170, 1193, 1208, 1209, 1219, 1233, 1260, 1261, 1264, 1323, 1340, 1359
\ifcat 1314, 1328, 1334
\ifcsdef . 192, 434, 655, 1138, 1148, 1165, 1179, 1181, 1185, 1195, 1196
\ifcsempty 1179
\ifcsundef 643, 644, 1186–1188
\ifdatagidxbalance 707, 775, 1170, 1194
\ifdatagidxshowgroups 675, 689, 692, 693, 696, 707, 1149, 1150, 1159, 1161, 1163, 1170
\ifdef 3, 49, 652, 733, 806, 1137, 1145, 1146, 1163, 1176, 1184, 1285, 1291, 1298
\ifdefempty 221, 272, 421, 642, 1136, 1137, 1141–1143, 1150, 1154, 1159, 1161, 1162, 1164–1168, 1171, 1178, 1179, 1182, 1183, 1188, 1193, 1233, 1238, 1271–1275, 1278–1280, 1286–1288, 1290–1292, 1294–1297, 1299, 1300, 1302, 1305–1315, 1318–1321, 1324, 1326, 1327, 1334–1339, 1341, 1347, 1348
\ifdefequal 1150, 1154, 1159, 1161, 1164, 1166, 1167, 1197, 1237, 1307
\ifdim 1087, 1088, 1093–1095, 1100, 1128, 1149, 1151–1153, 1155, 1156, 1158, 1160, 1161, 1201, 1213, 1223
\ifdtlautokeys 595, 1287
\ifDTLbarxaxis 859, 895, 1083, 1090, 1097
\ifDTLbaryaxis 859, 896, 1083, 1090, 1097
\ifDTLbarytics 859, 885, 1083, 1087, 1093
\ifDTLbox 938, 1213, 1215–1217
\ifDTLcolorbarchart 851, 865, 1082, 1084
\ifDTLcolorpiechart 908, 912, 1198, 1199
\ifdtlcompareskipcs 23, 236, 1324, 1325, 1335
\ifDTLgrid 939, 1214
\ifDTLlistskipempty 23, 30, 1293, 1294, 1299
\ifDTLmidsentence . . 791, 1102, 1106–1109, 1116, 1118, 1119, 1122
\ifDTLnewdbonload . . 595, 595, 1286
\ifdtlnoheader 595, 608, 613, 628, 1286, 1287
\ifDTLperiod 790, 791, 1102
\ifDTLrotateinner 908, 1201
\ifDTLrotateouter 908, 1202
\ifDTLshowlines . . . 938, 1207, 1218
\ifDTLshowmarkers . 938, 1207, 1217
\ifDTLstartsentence 791, 792, 1102
\ifdtlverbose 3, 25, 589, 1278, 1279, 1291, 1359
\ifDTLverticalbars 851, 851, 854–856, 870–872, 886, 889, 891–893, 895–898, 904, 905, 907, 1083, 1087–1091, 1093–1098
\ifDTLxaxis 937, 1207, 1214
\ifDTLxminortics 939, 1209, 1212, 1214, 1215
\ifDTLxtics 939, 1212, 1214
\ifDTLxticsin 937, 1207, 1215
\ifDTLxticstrue 939
\ifDTLyaxis 937, 1207, 1214
\ifDTLyminortics 939, 1209, 1213, 1214, 1216
\ifDTLytics 939, 1212, 1216
\ifDTLyticsin . 937, 1207, 1216, 1217
\ifDTLyticstrue 939
\ifentryused 749, 771, 772, 778, 1181, 1191, 1197
\ifeof 1286–1288
\iffemale 1059, 1367
\iffemalelabel 1044, 1364

\IfFileExists 708, 779, 1170, 1285, 1286
\ifFPmessages 297, 1359
\ifmale 1058, 1366, 1375
\ifmalelabel 1043, 1364
\ifmmode 191
\ifnewtermfield 742, 1179
\IfNoValueTF 289, 290, 296, 297, 302, 303, 311, 312, 1058–1061
\ifnum 47, 102, 103, 228–230, 259, 260, 271, 272, 275, 291, 292, 308, 407, 421, 490, 492, 494, 495, 510, 537, 539, 540, 1084, 1103–1105, 1110, 1111, 1128, 1129, 1139, 1142, 1164, 1182, 1183, 1189, 1194, 1198, 1199, 1201, 1205, 1218, 1228, 1230, 1233, 1238, 1247–1250, 1253, 1255–1260, 1264, 1266, 1268–1270, 1274, 1275, 1277, 1278, 1286, 1292, 1294, 1296, 1297, 1300, 1303, 1304, 1316, 1318, 1320–1323, 1325, 1326, 1334–1336, 1339–1341, 1343–1346, 1349–1353, 1357, 1358, 1362, 1363, 1369–1375
\ifnumequal 1167, 1168
\ifnumgreater 1169
\ifodd 1251, 1254
\IfPackageLoadedTF 9, 318, 659, 780, 852, 908, 930, 1038
\ifpersonexists 1054, 1365–1367, 1375
\ifstrempty 420, 644, 1173, 1184, 1188, 1204, 1225, 1233, 1234, 1237, 1260, 1328, 1334
\ifstrequal 276–281, 1187, 1328–1333
\IfSubStringInString 1279, 1280, 1316–1318
\iftermexists 653, 738, 742, 746, 754, 759, 761, 764, 778, 1179–1181, 1197
\ifthenelse 499, 541, 545, 550, 552, 555, 559, 562, 566, 720, 787, 788, 877, 899, 926, 963, 1016, 1087, 1092, 1100, 1101, 1179, 1218, 1251, 1254, 1267, 1268, 1270–1272, 1274, 1323, 1369
\IfTrackedIsoCode 267
\ifundef . . 707, 1086, 1170, 1178, 1291
\IfValueT . 321, 525, 536, 542, 546, 555, 559, 575, 587, 607, 627, 708, 710, 773, 865, 877, 899, 926, 1016
\IfValueTF 541, 545, 550, 552, 555, 559, 562, 728, 786, 863, 963, 1063
\ifx 269, 270, 273, 299, 301, 302, 308, 421, 487, 493, 502, 510, 539, 644, 780, 848, 1086, 1087, 1090–1093, 1097, 1098, 1102, 1104, 1105, 1129–1131, 1141, 1180, 1181, 1186, 1188, 1200, 1211–1219, 1223–1225, 1233, 1236–1240, 1243–1246, 1248, 1251, 1255–1257, 1259, 1263, 1266, 1273, 1276–1278, 1286–1288, 1293, 1300–1304, 1313, 1315–1317, 1320–1322, 1327, 1328, 1334, 1342, 1344, 1350, 1352, 1359, 1360, 1363, 1365–1368
\ignorespaces 814, 1112
\immediate 781, 814, 1099, 1112, 1130–1132
\in 400, 401, 443, 444, 486, 489, 510, 609, 611–613, 616, 617, 1062, 1227, 1228, 1239, 1245, 1246, 1254, 1259, 1261–1264, 1275, 1279, 1281, 1283, 1284, 1367
\indexspace 675, 689, 692, 1149, 1159, 1161
\inname 782, 795, 801–803, 805, 1099, 1107–1109
\input 1170, 1285
\inputencodingname 3
\InputIfFileExists 24, 55
\int 2, 5, 6, 13, 26–30, 36–39, 41, 58, 63–65, 72, 75, 79–81, 83, 86–89, 96–98, 112–114, 116–122, 124, 134–139, 147, 148, 158, 159, 161, 162, 164, 165, 170, 172–175, 177, 184, 194–196, 198–208, 222, 226–229, 231, 234, 288–290, 310–312, 314, 319, 320, 322–325, 328, 329, 335–340, 342–344, 347–349, 352–355, 358, 360–373, 376–380, 384, 386–390, 392–394, 399, 400, 402–404, 411–414, 417, 419–427, 429, 430, 432–434, 440, 441, 444–448, 450–454, 456, 457, 462, 467–480, 482, 484, 486, 492, 495–498, 500–504, 506–508,

511–516, 518, 519, 525–531, 537, 539, 541–554, 569, 573–577, 579–585, 587, 589, 590, 596, 598, 600, 601, 605, 608–612, 622, 627–630, 632, 634–639, 646, 647, 662, 664, 666, 677, 696–701, 704–706, 711, 739, 749, 751–753, 756, 766, 772, 775, 777, 786, 788, 792–794, 797, 798, 809–813, 818, 842–845, 849, 859–864, 867, 876, 887, 888, 897, 902, 906, 912, 916, 917, 919, 921, 926, 929–931, 941–944, 948, 952, 956–959, 976, 985, 986, 990, 991, 996, 997, 1001–1003, 1005, 1006, 1009, 1018, 1019, 1021, 1023, 1024, 1028, 1029, 1034, 1039–1041, 1047–1049, 1051–1054, 1057–1061, 1069
\ior 592, 605, 628, 629
\iota 732, 1175
\iow . . . 592, 607, 608, 614–620, 847, 848
\item 668, 674–676, 683, 684, 689, 693, 1146, 1149, 1150, 1154, 1155, 1159, 1161

J

\jobname 656, 780, 1098, 1139

K

\kappa 732, 1175
\keys 7, 9, 10, 19, 23, 24, 265, 266, 318, 321, 322, 324, 371, 372, 391, 392, 467, 468, 474, 475, 480, 482, 483, 521, 525, 536, 567, 570, 573, 575, 600, 606, 607, 627, 656, 660, 708–710, 712, 720, 724, 726, 735, 773, 775, 779, 780, 851, 852, 866, 876, 877, 880, 899, 901, 908, 913, 916, 926, 928, 944, 961, 1016, 1021, 1038, 1039, 1045, 1047
\keyval 321, 334, 362, 438, 485, 568, 587

L

\L 733, 1175
\l 2–12, 19–23, 25, 26, 29–31, 33–42, 45–47, 49, 50, 53–56, 59–61, 70, 75, 77, 78, 86, 87, 90, 93–104, 108–110, 112–131, 135–149, 157–188, 192–225, 228–230, 233–241, 243–249, 261–264, 266–268, 282, 283, 285–290, 309–312, 314–395, 399, 400, 404, 411–417, 419, 421–431, 433, 434, 443, 445, 447–449, 451, 452, 454–457, 460, 461, 463–486, 488, 514–516, 518–522, 524–537, 541–565, 567–572, 576–580, 582–590, 593–614, 616, 617, 624, 627–641, 646, 647, 649–654, 659–662, 664, 665, 667, 668, 670–672, 676–680, 683–687, 690–692, 696, 698–707, 709, 710, 712, 713, 718–725, 727, 728, 733, 735–745, 747–770, 772–776, 779–781, 785, 786, 790, 792–794, 797, 798, 813, 818, 819, 841–846, 852–856, 858–864, 866–869, 871, 872, 874–907, 909–935, 938–946, 948, 951–1025, 1027–1037, 1039–1041, 1044–1054, 1057–1063, 1069–1074, 1080–1082, 1175
\Label 661, 662, 666, 676, 682, 684, 690, 691, 693, 694, 697, 709, 726, 777, 1133, 1142–1144, 1150, 1154, 1155, 1159, 1160, 1162, 1164, 1171, 1172, 1196, 1197
\label 26, 534, 1263
\Lambda 733, 1175
\lambda 732, 1175
\LaTeX 33
\lccode 275, 1318, 1328
\leavevmode 847, 1131
\legacy 632–634, 675, 747, 777, 784, 790, 791, 799–805, 807, 808, 922, 923, 969, 970, 972, 973, 976–978, 980–983, 985, 986, 988–991, 993, 996, 997, 999–1002, 1005, 1006, 1020
\let 33, 193, 194, 221, 250, 254, 256–261, 269, 271, 272, 299, 301, 302, 306–310, 420, 439, 487, 490, 493, 509–511, 539, 566, 624, 625, 628, 641, 656, 666, 668, 704, 705, 713, 733, 734, 736, 776, 813, 819, 840, 847, 1085–1088, 1091, 1092, 1094, 1095, 1097, 1100, 1102, 1109–1111, 1114, 1127–1131, 1133, 1138, 1141, 1143–1150, 1152, 1153, 1156–1158, 1163,

1164, 1166–1168, 1176–1180, 1182–1185, 1193–1197, 1200–1205, 1210–1214, 1216–1218, 1221, 1223, 1224, 1227, 1228, 1230, 1233, 1237–1239, 1246–1259, 1261, 1264, 1266, 1271–1278, 1285, 1286, 1288–1290, 1293, 1295–1298, 1300–1322, 1324, 1325, 1335, 1338–1344, 1347, 1348, 1350–1353, 1355, 1357–1360, 1362–1368, 1375
\letcs 1100, 1101, 1181, 1182, 1185–1188
\linewidth 670–672, 675, 678, 683, 685, 689, 691–693, 696, 1147, 1148, 1150, 1151, 1154, 1156, 1159–1161, 1163
\loadgidx 708, 1170, 1171
\Location 665, 666, 677, 694, 698, 699, 725, 778, 1142, 1144, 1151, 1155, 1156, 1159, 1161, 1162, 1165, 1166, 1172, 1196, 1197
\Long 667, 725, 1145, 1172
\long . . 259, 260, 269, 270, 486–490, 644, 645, 1225, 1226, 1245, 1246, 1293, 1319, 1357, 1358, 1367
\long@addto@envbody 269, 270, 1293
\long@collect@@body 269, 270, 1293
\long@collect@body 269, 1249, 1250, 1293, 1358
\long@push@begins . . 270, 270, 1293
\LongPlural 725, 1172
\loop 1189, 1286–1288
\lowercase 1317

M

\m@ne 1286
\macro 238
\makeatletter 32, 614, 625, 1282, 1283
\makebox . . 675, 682, 684, 689–691, 693, 1150, 1154, 1155, 1159–1162
\makefirstuc 767
\MakeLowercase . 722, 723, 1177, 1317
\MakeTextLowercase 722, 723, 1133, 1177, 1318
\MakeTextUppercase 722, 723, 1133, 1177, 1181, 1190, 1316
\MakeUppercase 722, 723, 1106–1109, 1116, 1118, 1119, 1122, 1177, 1316, 1370–1375
\malechild 1067, 1368
\malechildren 1067, 1368, 1373
\malelabels 1042, 1364, 1365
\malename 1068, 1368, 1375
\maleobjpronoun 1065, 1368
\maleparent 1067, 1368
\malepossadj 1065, 1368
\maleposspronoun 1066, 1368
\malepronoun 1065, 1368
\malesibling 1068, 1368
\malesiblings 1068, 1368, 1375
\mbox 694, 1162
\MessageBreak 710, 774
\MFUaddmap . . . 754, 767, 768, 770–772
\MFUexcl . . 750, 751, 766, 767, 769, 770
\mscthesisname 783, 1099
\msg 79–86
\mu 732, 1175
\multicolumn 516
\multiply 1150, 1151, 1154, 1165, 1277, 1278, 1301, 1303, 1323

N

\Name 666, 674–678, 682–685, 690–694, 697, 725, 1145, 1149–1151, 1154–1156, 1159–1162, 1164, 1172
\NeedsTeXFormat 1, 282, 283, 291, 297, 304, 313, 646, 778, 850, 907, 930, 1037, 1082, 1098, 1132, 1197, 1206, 1224, 1291, 1358, 1361, 1364
\newacro 745, 755, 765, 1181
\newcommand 4, 9, 10, 12–16, 18, 23, 25, 27, 28, 30–32, 40, 41, 43–46, 48–51, 63, 64, 66, 78, 79, 107, 112, 144, 145, 147, 149, 150, 185, 186, 190–193, 202, 210, 219, 220, 225, 226, 233, 234, 237–259, 261–263, 268–272, 274–276, 279, 282, 284, 291, 292, 308, 314, 315, 317, 318, 330, 332, 401–410, 418–420, 428, 429, 431–434, 436, 438, 440–444, 447, 453, 457, 462, 482, 490–493, 495, 501, 502, 506, 508–513, 516–519, 533, 539, 540, 566, 583, 586, 592, 595, 608, 621–626, 641–645, 647, 648, 651–654, 656, 659, 661–663, 665–670, 673, 694, 695, 698, 702, 703, 705–707, 711,

725, 726, 729–734, 741, 742, 746, 747, 752, 755, 758, 759, 761, 764, 771–773, 776, 778, 782, 785, 788, 789, 791, 792, 794, 795, 805–818, 839–841, 847–851, 854, 856–859, 862, 864, 865, 909–912, 935–939, 964, 1033, 1035, 1037–1040, 1054, 1061, 1064, 1069, 1072, 1082–1085, 1091, 1098–1113, 1126, 1127, 1129–1139, 1141–1149, 1151, 1156, 1160, 1163, 1165–1176, 1178–1189, 1191, 1193, 1195–1200, 1204–1208, 1210, 1221–1232, 1234–1250, 1253, 1255–1262, 1264–1276, 1278, 1279, 1281, 1283, 1285, 1286, 1289–1302, 1304–1315, 1317–1324, 1328, 1331, 1334, 1337, 1341–1375
\newcount 25, 48, 63, 260, 270, 404, 494, 495, 544, 599, 615, 623, 778, 1083, 1102, 1183, 1189, 1197, 1205, 1207, 1226, 1227, 1229, 1249, 1268, 1282–1285, 1291, 1298, 1299, 1319, 1328, 1358
\newcounter 494, 661, 788, 792, 794, 859, 910, 937, 1039, 1083, 1101–1103, 1128, 1133, 1198, 1207, 1249, 1364
\NewDocumentCommand 34, 49, 55, 58, 86, 88, 90–93, 104, 106, 110, 111, 115, 186, 187, 200–210, 215, 216, 225–227, 231, 239–243, 246–249, 265, 285–290, 292–296, 300–303, 305–312, 314, 315, 320, 331, 399–403, 405, 406, 418, 432, 434, 435, 437, 444, 446, 450–455, 457, 458, 461, 467, 471, 476, 480–482, 485, 491, 492, 494, 514, 516, 519, 520, 524, 533, 535, 536, 541, 543, 545, 547, 549, 551–553, 555, 557, 558, 560, 562, 565, 575, 586, 606, 627, 641, 642, 646, 651, 663–665, 669, 673, 677, 699, 708, 710, 727, 742, 743, 745, 749–751, 754, 764, 766–770, 773, 777, 780–787, 789, 790, 792, 796–805, 807, 814, 846, 848, 849, 860, 861, 863, 865, 877, 899, 912, 926, 963, 1016, 1035–1037, 1042–1044, 1052–1054, 1058–1063, 1070–1072, 1075–1082
\NewDocumentEnvironment 261, 471, 495, 496, 812
\newenvironment 666, 1112, 1249, 1250, 1358
\newgidx 655, 707, 710, 741, 743, 1138, 1170
\newif 3, 23, 48, 221, 595, 707, 790, 791, 851, 859, 908, 937–939, 1082, 1083, 1102, 1136, 1138, 1166, 1170, 1198, 1207, 1286, 1293, 1298, 1319, 1321, 1324
\newlength 25, 663, 684, 857, 858, 864, 909, 912, 936–938, 1083, 1084, 1100, 1136, 1156, 1198, 1199, 1207, 1291
\newperson 1044, 1364
\newread 1285
\newrobustcmd 31, 47, 50, 105, 220, 221, 225–228, 230, 231, 238, 244, 245, 298, 299, 320, 332, 407, 409, 418, 428, 442, 446, 449, 462, 496, 502, 504, 537, 665, 770–772, 1109, 1294, 1299
\newterm 655, 734, 735, 745, 755, 765, 1139, 1176, 1181
\newterm@database 721, 1172, 1173, 1177–1182, 1185–1189, 1191, 1193–1195
\newterm@defaultshook 725, 1172, 1173, 1177
\newterm@description 721, 1172, 1177, 1178
\newterm@extrafields 725, 1172, 1173, 1179
\newterm@label 721, 1171, 1176–1180
\newterm@long . 721, 1172, 1177, 1178
\newterm@longplural 721, 1172, 1177, 1178
\newterm@name 721, 1176
\newterm@parent 721, 1172, 1177, 1179
\newterm@parentdatabase . . . 1179
\newterm@plural 721, 1172, 1176, 1178
\newterm@see . . 721, 1172, 1177, 1178
\newterm@seealso 721, 1172, 1177–1179
\newterm@short 721, 1172, 1177, 1178
\newterm@shortplural 721, 1172, 1177, 1178

\newterm@sort . 721, 1172, 1177, 1178
\newterm@styles 1192–1194
\newterm@symbol 721, 1172, 1177, 1178
\newterm@text . 721, 1172, 1176, 1178
\newtermaddfield 727, 728, 729, 741, 1173
\newtermfield 741, 1179
\newtermlabelhook 722, 729, 1174, 1177
\newtermsorthook 724, 729
\newtoks 25, 399, 415, 444, 495, 537, 614, 615, 623, 1226, 1227, 1232, 1239, 1240, 1249, 1274, 1275, 1278, 1282–1284, 1291
\newwrite 624, 1130, 1279
\nobreak 675, 689, 693, 1150, 1159, 1161
\nobreakspace 33
\NoCaseChange 1064, 1188
\node 938
\noexpand 251–259, 408, 409, 505, 507, 510, 537, 538, 540, 566, 644, 645, 808–812, 1089–1091, 1095–1098, 1109–1111, 1142, 1144, 1154, 1163, 1173, 1174, 1177, 1180–1187, 1189, 1193, 1194, 1196, 1204, 1217–1219, 1224–1226, 1229–1231, 1233–1237, 1239–1245, 1247, 1248, 1251, 1252, 1255–1259, 1261, 1263, 1265–1267, 1274–1278, 1288–1290, 1294, 1299, 1300, 1302, 1305, 1313, 1314, 1316, 1317, 1321, 1325–1328, 1334–1337, 1340–1342, 1344–1357, 1361, 1362
\nolinkurl 806
\not 1205
\nr 1084, 1139–1141, 1170, 1191–1193, 1208–1210, 1226, 1291, 1359
\nu 732, 1175
\null 1260
\num 9
\number 270, 294, 302–304, 308, 309, 407, 408, 419–421, 492, 494, 495, 505, 507, 537, 539, 540, 620, 636, 637, 912, 1087, 1088, 1090, 1093, 1094, 1112, 1113, 1142, 1143, 1183, 1189, 1211, 1212, 1214–1216, 1228–1230, 1233–1236, 1240–1245, 1248, 1249, 1251, 1252, 1255–1258, 1261, 1262, 1264–1270, 1275, 1276, 1279, 1282–1284, 1287–1289, 1294, 1295, 1304, 1316, 1318, 1323, 1343, 1361–1363
\numbername 783, 800, 1099, 1107
\numexpr 308, 309, 494, 1363

O

\O 733, 1175
\o 733, 1175
\OE 733, 1175
\oe 733, 1175
\ofname . 782, 795, 801, 1099, 1106, 1107
\Omega 733, 1175
\omega 733, 1175
\onecolumn 773, 1193
\openin 1286
\openout 1131, 1279, 1281, 1283
optimize (option) 659
options:
optimize 659
\or 229, 421, 813, 814, 1084, 1085, 1112, 1133–1137, 1139, 1170, 1193, 1208–1210, 1219–1221, 1233, 1260–1262, 1264, 1323, 1340, 1359

P

\p@ 668, 1146, 1156
\PackageError 5, 11, 22, 24, 79, 81, 83, 89, 92, 114, 116, 118, 119, 135, 190, 199, 259, 314, 317, 321, 331, 332, 399–403, 405–407, 409, 418, 419, 426, 428, 429, 432, 433, 435–437, 439, 442, 444–446, 455, 457, 458, 462, 471, 474–477, 480, 482, 484–486, 491–494, 496, 501–512, 515, 523, 525, 529, 536, 540, 547–549, 552, 554, 556, 558, 560–562, 564, 565, 573, 576, 578, 582, 587, 588, 591, 594, 596, 597, 601, 605, 607, 614, 625, 626, 628, 643, 644, 646, 651, 673, 710, 728, 736, 739, 741–744, 746–748, 755, 765, 767, 774, 776, 780, 783, 846, 847, 857, 858, 860, 866, 877, 878, 882, 899, 900, 910, 926, 929, 952, 957–959, 966–968, 1016–1018, 1045, 1046,

1048, 1055–1057, 1061, 1084, 1086, 1092, 1099, 1130–1133, 1149, 1179, 1180, 1185, 1186, 1188, 1190, 1195, 1198–1200, 1205, 1211, 1221, 1226–1232, 1234–1237, 1239, 1240, 1243–1245, 1247, 1248, 1250, 1253–1260, 1266–1270, 1272, 1273, 1275, 1278, 1280, 1281, 1283–1285, 1289, 1357, 1365–1375
\PackageInfo 8, 55, 643
\PackageWarning . . 8, 10, 11, 104, 117, 118, 198, 206, 208, 270, 304, 313, 334, 351–353, 361, 371, 372, 375, 383, 450, 476, 515, 525–527, 529, 530, 536, 573, 587, 591, 595, 601, 625, 628, 665, 699, 702, 734, 752, 760, 764, 779, 848, 876, 912, 916, 943, 961, 964, 1040, 1064, 1069, 1073, 1074, 1165, 1167, 1187, 1208, 1275, 1286–1288, 1294
\PackageWarningNoLine . 654, 1138
\pagename 783, 799, 800, 1099, 1106, 1107
\pageref 26
\pagesname 783, 799, 800, 1099, 1106, 1107
\par 320, 631, 668, 670, 682, 684, 691, 772, 1146, 1154, 1155, 1159, 1160, 1162, 1163, 1193, 1226
\Parent 666, 678, 683, 685, 690, 692, 726, 777, 1144, 1151, 1155, 1156, 1159, 1161, 1172, 1196
\Parents 1038
\parents 1038
\parindent 668, 674, 676–678, 682, 683, 685, 690–693, 696, 698, 1146, 1149–1151, 1154, 1156, 1159–1163, 1165
\parse@wordsnext 1344
\parshape 683, 696, 1154, 1163
\parskip 668, 674, 676–693, 696, 698, 1146, 1149–1163, 1165
\PassOptionsToPackage 319, 660, 780, 852, 908, 930, 1039
\path 891
\pdfstrcmp 234
\penalty 847, 1131
people (counter) 1039
\Peoplechild 1038, 1079, 1373
\peoplechild 1038, 1079, 1373
\peopleforenames 1070
\peoplefullname 1070, 1369
\peoplename 1071, 1369
\Peopleobjpronoun 1038, 1075, 1371
\peopleobjpronoun 1038, 1075, 1371
\Peopleobjpronounii . . . 1038, 1077
\peopleobjpronounii . . . 1038, 1077
\Peopleparent 1038, 1079, 1374
\peopleparent 1038, 1079, 1374
\Peoplepossadj . . . 1038, 1076, 1372
\peoplepossadj . . . 1038, 1076, 1372
\Peoplepossadjii 1038, 1078
\peoplepossadjii 1038, 1078
\Peopleposspronoun 1038, 1076, 1373
\peopleposspronoun 1038, 1076, 1372
\Peopleposspronounii . . 1038, 1078
\peopleposspronounii . . 1038, 1078
\Peoplepronoun . . . 1038, 1075, 1370
\peoplepronoun . . . 1038, 1075, 1370
\Peoplepronounii 1038, 1077
\peoplepronounii 1038, 1077
\Peoplesibling 1038, 1080
\peoplesibling . . . 1038, 1080, 1375
\peoplesurname 1071
\peopletitlesurname 1072
\person 1041, 1044–1046, 1051–1060, 1062–1082
person (counter) 1040
\PersonAddFemaleLabel . 1042, 1042
\PersonAddMaleLabel . . . 1042, 1042
\PersonAddNonBinaryLabel . . 1043
\Personchild 1079, 1373, 1374
\personchild 1079, 1373
\PersonFemaleCount 1040
\personforenames 1070
\personfullname . . 1070, 1368, 1369
\Persongender 1080
\persongender 1080, 1375
\PersonIfAllFemale 1059, 1059
\PersonIfAllMale 1058, 1058
\PersonIfAllNonBinary 1060
\PersonIfAllUnknownGender . 1061
\PersonIfFemale 1058, 1059
\PersonIfFemaleLabel 1043, 1044, 1047
\PersonIfMale 1057, 1058
\PersonIfMaleLabel 1043, 1043, 1047
\PersonIfNonBinary 1059
\PersonIfNonBinaryLabel 1044, 1048
\PersonIfUnknownGender 1060

\personlastsep 1069, 1069, 1368–1370
\PersonMaleCount 1040
\personname 1071, 1369
\PersonNonBinaryCount 1040
\Personobjpronoun 1075, 1371
\personobjpronoun 1075, 1371
\Personobjpronounii 1077
\personobjpronounii 1077
\Personparent 1079, 1374
\personparent 1079, 1374
\Personpossadj 1076, 1372
\personpossadj . . . 1076, 1371, 1372
\Personpossadjii 1078
\personpossadjii 1078
\Personposspronoun 1076, 1372, 1373
\personposspronoun 1076, 1372, 1373
\Personposspronounii 1078
\personposspronounii 1078
\Personpronoun . . . 1075, 1370, 1371
\personpronoun 1075, 1370
\Personpronounii 1077
\personpronounii 1077
\personsep . . . 1069, 1069, 1368–1370
\PersonSetFemaleLabels 1042
\PersonSetLocalisation 1063
\PersonSetMaleLabels 1042
\PersonSetNonBinaryLabels . 1043
\Personsibling 1080, 1375
\personsibling . . . 1080, 1374, 1375
\personsurname 1071
\persontitlesurname 1071
\persontitlesurnamesep 1049, 1072, 1072
\PersonTotalCount 1039
\PersonUnknownCount 1040
\PersonUnknownGenderCount . 1040
\pgf@pathminx 898, 907
\pgf@pathminy 898, 907
\pgfmathabs 310, 1364
\pgfmathadd . 306, 307, 310–312, 1362
\pgfmathdivide . . 307, 310–312, 1362
\pgfmathifthenelse 305, 306, 1361, 1362
\pgfmathmax 310, 1363
\pgfmathmin 309, 1363
\pgfmathmul 1364
\pgfmathmultiply 307, 311, 312, 1362
\pgfmathneg 310
\pgfmathparse 308, 309, 1208, 1362, 1363
\pgfmathpow 307
\pgfmathresult 305–312, 1208, 1361–1364
\pgfmathsqrt 307, 312, 1362
\pgfmathsubtract 307, 311, 312, 1362
\pgfpathlineto 896, 897, 1033, 1090, 1097, 1224
\pgfpathmoveto 896, 897, 1033, 1090, 1097, 1224
\pgfplothandlerlineto . 1008, 1219
\pgfplothandlermark 964, 1008, 1208, 1219
\pgfplotstreamend 1008, 1219
\pgfplotstreampoint 963, 1008, 1206, 1219
\pgfplotstreamstart . . . 1007, 1218
\pgfpoint 855, 856, 886, 893–895, 897, 898, 904, 905, 962, 975, 1033, 1088–1091, 1094–1098, 1213, 1221, 1224
\pgfpointadd . 1089–1091, 1095–1098
\pgfpointlineattime . . . 1090, 1096
\pgfpointxy 889, 890, 896, 897, 963, 1008, 1089–1091, 1095–1098, 1206, 1219
\pgfsetdash 935, 936, 1206
\pgfsetstrokecolor 1005, 1006
\pgfsetxvec 886, 904, 975, 1088, 1094, 1213
\pgfsetyvec 886, 904, 975, 1088, 1094, 1213
\pgftext 892, 893, 897, 899, 905, 1089–1091, 1094–1098
\pgftransformcm . . . 962, 1213, 1221
\pgftransformreset 964, 976
\pgftransformxscale 1208
\pgftransformyscale 1208
\pgfusepath 897, 1008, 1033, 1090, 1097, 1219, 1224
\pgfuseplotmark 935, 1206
\phdthesisname 783, 1099
\Phi 733, 1175
\phi 733, 1175
\Pi 733, 1175
\pi 732, 1175
\Plural 667, 726, 1145, 1172
\pluralchild . 1067, 1368, 1373, 1374
\pluralobjpronoun 1065, 1368, 1371
\pluralparent 1067, 1368, 1374
\pluralpossadj . . . 1066, 1368, 1372

\pluralpospronoun 1066, 1368, 1373
\pluralpronoun ... 1065, 1368, 1370
\pluralsibling ... 1068, 1368, 1375
\postnewtermhook ... 740, 741, 1179
\pounds 185, 193, 1305
\prg 5–7, 29, 64–73, 103, 104, 141–147, 189, 211, 212, 222, 223, 266, 405, 468, 471, 481, 518, 528, 565, 1055
\printterms 710, 773, 1193
\printterms@condition 712, 1193, 1196
\printterms@setupmulticol 772, 775, 1193, 1194
\printterms@setuptwocol 773, 775, 1193, 1194
\printtermsrestoreonecolumn 773, 773, 1193
\printtermsstartpar 772, 775, 1193, 1195
\process@envbody ... 269, 270, 1293
\ProcessKeyOptions 24, 318, 660, 780, 852, 908, 1039
\ProcessOptions 319, 660, 780, 852, 908, 930, 1039, 1083, 1198
\ProcessOptionsX 1098, 1141, 1226, 1291, 1359, 1361
\prop 55, 101, 102, 148, 149, 177, 186, 321, 325–327, 329–332, 379, 380, 384, 391, 467, 468, 473, 479, 597–601, 627, 629, 630, 633, 634, 636, 637, 1035–1037
\protect 1139, 1301, 1313–1315, 1317, 1320
\protected@edef 45, 270, 317, 503, 507, 593, 642, 722, 724, 758, 1100, 1102, 1106–1109, 1116, 1118, 1119, 1122, 1173, 1184, 1187, 1197, 1235, 1237, 1240, 1243, 1245, 1255, 1257, 1260, 1279, 1280, 1287, 1290, 1293, 1298, 1315, 1317, 1318, 1326, 1344–1352
\protected@write 653, 758, 1138, 1187, 1196, 1280–1284
\protected@xdef 1128, 1177, 1178, 1282, 1290, 1365
\providecommand 4, 663, 668, 734, 782, 783, 1040, 1099, 1141, 1146, 1176, 1226, 1291, 1359, 1361, 1368
\ProvidesFile 282, 283, 291, 297, 304
\ProvidesPackage 1, 304, 313, 646, 779, 850, 908, 930, 1037, 1082, 1098, 1132, 1197, 1206, 1224, 1291, 1358, 1361, 1364
\Psi 733, 1175
\psi 733, 1175

Q

\q 29, 30, 36, 37, 40, 41, 57, 58, 70, 77, 94–96, 100, 102, 103, 149, 216, 223, 224, 262, 263, 272–274, 320, 322, 351, 353, 354, 356, 357, 370, 410–415, 448, 449, 451, 454, 458–461, 463–468, 481, 488, 489, 565, 578, 583, 639, 640, 865, 866, 870, 927, 929
\q@nil 408, 410, 487–491, 493, 1231, 1232, 1240–1242, 1244–1249, 1276
\quark 70, 102, 103, 178, 219, 331, 332, 336, 344, 353, 356, 366, 370, 371, 373, 374, 471, 479, 482, 489, 598, 627, 629, 630, 633, 634, 636, 637, 640

R

\raggedleft 673, 1148
\raggedright ... 674, 696, 1149, 1163
\raisebox 668, 753, 1146, 1185
\read 1285, 1286, 1290
\ref 26
\refstepcounter 26, 494, 496, 499, 676, 684, 691, 694, 698, 787, 788, 1100, 1101, 1150, 1155, 1160, 1162, 1165, 1250, 1251, 1253, 1254
\regex 3, 46, 53, 54, 56, 57, 59–62, 93, 94, 97, 98, 100, 105, 107, 114, 115, 122, 123, 143, 144, 146, 149–172, 174–176, 178–184, 210–214, 238, 244, 245, 267, 283, 314–317, 515, 516, 593, 594, 631, 640, 642, 700, 722, 790, 845, 846
\relax .. 47, 235, 236, 259–261, 269, 271, 273, 275–281, 299, 302–304, 308, 309, 315, 407, 421, 429, 487, 488, 490–492, 495, 502, 505, 509, 510, 537, 539, 540, 615, 616, 623, 624, 644, 682, 792, 1016, 1033, 1084–1087, 1090–1094, 1096–1098, 1100–1105, 1110–1113, 1127–1130, 1133–1136, 1139, 1142, 1144,

1149–1156, 1158–1166, 1170, 1182, 1184, 1186, 1189, 1193–1196, 1198–1201, 1204, 1205, 1208–1219, 1224, 1225, 1227–1230, 1232–1235, 1237, 1238, 1240, 1242, 1243, 1245–1261, 1264, 1266–1270, 1273–1279, 1281–1290, 1292, 1294–1297, 1299–1304, 1313–1346, 1349–1353, 1357–1360, 1362, 1363, 1369–1375
\removeallpeople 1053, 1366
\removepeople 1053, 1365
\removeperson 1052, 1365
\renewcommand 9, 11, 20–22, 50, 261–263, 268, 647, 654, 656–659, 713–716, 817–820, 822, 823, 825, 827, 829, 831, 833–835, 837, 839, 840, 1062, 1063, 1114, 1115, 1117–1119, 1121–1127, 1133–1140, 1149–1151, 1154–1156, 1159–1165, 1169–1172, 1176, 1177, 1191–1193, 1200, 1225, 1226, 1291, 1298, 1299, 1305
\RenewDocumentCommand .. 472–474
\RenewDocumentEnvironment 818, 840
\renewenvironment 1113, 1127
\repeat 1189, 1286–1288
\RequireDataBibDialect . 850, 850
\RequireDatatoolDialect 263, 268
\RequirePackage 1, 297, 304, 313, 319, 646, 660, 780, 852, 908, 909, 931, 1039, 1082, 1083, 1098, 1132, 1198, 1206, 1224, 1226, 1291, 1358, 1359, 1361, 1364
\RequirePersonDialect . 1082, 1082
\reset@font 848, 1131
\rho 732, 1175
\romannumeral 494, 495, 497–513, 860–862, 864, 911, 912, 918–920, 929, 930, 1083, 1084, 1088, 1198–1201, 1204, 1205, 1249–1259, 1285, 1287–1289

S

\s 62, 93, 122, 123, 150–157, 166–169, 315, 316
\s@DTLaddcolumn 418, 1232
\s@dtlenvforeach@args 1250
\s@dtlformatlist 50, 50, 1299
\s@DTLifstringlt 240
\section 667, 695, 1145, 1163
\See 648, 657, 658, 665, 666, 715, 716, 725, 778, 1134–1136, 1142, 1144, 1151, 1155, 1156, 1159, 1161, 1172, 1197
\SeeAlso 648, 666, 725, 778, 1134, 1144, 1151, 1155, 1156, 1159, 1161, 1172, 1197
\seealsoname ... 648, 663, 1134, 1141
\seename 648, 657, 658, 663, 715, 716, 1134–1136, 1141
\seq 2, 4–7, 9, 30, 31, 34–36, 39, 41, 46, 47, 49, 50, 55, 101, 102, 108, 116, 121, 123–130, 135, 141–143, 145–147, 158–188, 198–200, 206–208, 319–322, 330, 334–336, 356, 375–385, 387, 390, 395, 520–522, 524, 526–528, 531, 545–547, 549–554, 567–570, 576–580, 582–584, 586–590, 599–601, 632, 634, 639, 640, 653, 654, 660, 662, 664, 700–702, 777, 845, 853, 854, 866–869, 874, 875, 877, 880–882, 885–887, 897, 899, 901–903, 906, 914, 915, 931, 933–935, 940–944, 951, 952, 955–957, 965, 969–974, 980, 981, 983, 989, 993, 999, 1004, 1006, 1016–1018, 1021–1025, 1027–1029, 1031, 1032, 1037, 1041, 1051–1054, 1062, 1063, 1069, 1073, 1074
\setbool 948, 1140, 1226
\setcounter 494, 694, 697, 792, 794, 859, 910, 937, 1083, 1100, 1102, 1103, 1128, 1162, 1164, 1198, 1207, 1366, 1369
\setkeys 1086, 1092, 1170, 1171, 1177, 1193, 1194, 1200, 1211, 1262, 1286
\setlength 858, 936–938, 1083, 1085, 1087, 1088, 1091, 1093, 1094, 1096, 1098, 1146–1163, 1165, 1192, 1207, 1208, 1212, 1213, 1216, 1223
\settodepth 25, 1087, 1088, 1093, 1094

\settoheight 25, 971, 1087, 1088, 1093, 1094, 1146, 1185, 1212
\settowidth 25, 974, 1087, 1088, 1093, 1100, 1128, 1147, 1148, 1151, 1152, 1155, 1156, 1160, 1161, 1213
\sfcode 792, 1102
\Short 667, 725, 1145, 1172
\ShortPlural 725, 1172
\show 644, 1291
\showthe 643, 1290
\Siblings 1038
\siblings 1038
\Sigma 733, 1175
\sigma 732, 1175
\skip 668, 674, 676, 677, 679–682, 684–693, 698
\smallskip 670
\Sort 726, 740, 1172, 1196
\sort 36, 47, 579, 580, 589, 662
\space 33, 49, 237, 407, 492, 503, 505, 508, 510, 540, 648, 657, 695, 715, 716, 733, 792, 806, 964, 1084, 1086, 1092, 1102, 1131, 1135, 1136, 1138, 1139, 1163, 1174, 1176, 1198–1200, 1208, 1230, 1236, 1237, 1241–1245, 1248, 1250, 1253, 1255–1260, 1267, 1281–1285, 1298, 1313, 1314, 1341–1344, 1365, 1368
\spacefactor 792, 1102
\SS 733, 1175
\ss 733, 1175
\step 259, 260, 498, 1251, 1289, 1357, 1358
\stepcounter ... 26, 1128, 1365, 1369
\str .. 2, 53, 188, 189, 224, 234–237, 239–241, 243, 244, 246, 248, 249
\strcmp 234
\string 33, 55, 304, 313, 510, 608, 653, 734, 781, 814, 847, 848, 964, 1084, 1086, 1092, 1099, 1112, 1131, 1132, 1138, 1139, 1174, 1176, 1183, 1187, 1196, 1198–1200, 1208, 1211, 1237, 1250, 1253, 1255–1260, 1281–1285, 1315–1318
\Symbol 648–651, 667, 694, 725, 1136, 1137, 1145, 1162, 1172
\sys 3, 779

T

\t 631
\tabularnewline 519, 1260
\tau 732, 1175
\TE@or 259–261, 1357, 1358
\TE@throw 251–259, 808–812, 1109–1111, 1353–1357
\techreportname 783, 1099
\TeX 33
\Text 667, 725, 1145, 1172
\text ... 11, 33, 44–46, 77, 210–212, 220, 235, 236, 244, 245, 262, 275, 576, 653, 656, 708, 713, 714, 723, 735, 745, 769, 770, 791, 799–805, 845, 1041, 1058–1062, 1073, 1074
\textasciicircum 1290
\textasciitilde 1290
\textbaht 194
\textbf ... 516, 675, 689, 693, 722, 723, 1150, 1159, 1161, 1177, 1260
\textcent 193
\textcentoldstyle 193
\textcolonmonetary 193
\textcurrency ... 185, 192, 193, 1305
\textdollar 185, 193, 1305
\textdollaroldstyle 193
\textdong 193
\texteuro 185, 193, 1305
\textflorin 193
\textguarani 193
\textit 722, 723, 1177
\textlira 193
\textmd 722, 723, 1177
\textminus 191
\textnaira 193
\textnormal 647, 1134
\textpeso 193
\textrm 722, 723, 1177
\textsc 11, 722, 723, 806, 1177
\textsf 722, 723, 1177
\textsl 722, 723, 1177
\textsmaller 12
\textsterling 193
\textstirling 185, 1305
\textsuperscript ... 722, 723, 1177
\texttt 722, 723, 806, 807, 1177
\textwon 185, 193, 1305
\textyen 185, 193, 1305
\TH 733, 1175
\th 733, 1175

\the 51, 52, 260, 261, 269, 271, 272, 408, 409, 433, 486, 487, 489, 490, 497, 501, 503, 505, 507, 508, 537–539, 566, 642, 652, 1087, 1088, 1093, 1094, 1110, 1111, 1128, 1129, 1154, 1163, 1204, 1205, 1217–1219, 1222–1224, 1231, 1233–1236, 1239–1248, 1250–1258, 1265–1267, 1274–1276, 1278–1280, 1282, 1284, 1285, 1287–1290, 1293, 1295–1300, 1302, 1304, 1305, 1311–1319, 1342, 1358, 1364
\theDTLgidxChildCount . 661, 1133
\Thee 1038
\thee 1038
\theHDTLbibrow 788, 1101
\theHDTLgidxChildCount 661, 1133
\theHDTLrow 494, 788, 1101, 1249
\theHDTLrowi 494, 1249
\theHDTLrowii 494, 1249
\theHDTLrowiii 494, 1249
\Their 1038
\their 1038
\Theirs 1038
\theirs 1038
\Them 1038
\them 1038
\thepage 1131
\Theta 733, 1175
\theta 732, 1175
\They 1038
\they 1038
\thiscol 505, 507, 1256, 1257
\tikz 1094, 1224
\tl 2–5, 8, 10, 11, 13–17, 21, 22, 26, 29–31, 33–35, 37, 40–43, 45–49, 53, 56, 57, 59–61, 66–81, 83–110, 113–121, 124–130, 132, 133, 143, 148–150, 166–169, 177, 178, 186, 188–212, 214, 216–230, 235–238, 244, 245, 261–264, 266, 267, 270, 272–275, 282, 285–290, 306–312, 314, 315, 317–336, 338, 340, 343, 344, 347, 351, 352, 354, 357–362, 364–367, 369–378, 380, 383, 384, 386–389, 391, 394, 395, 405–407, 410–415, 417, 421–424, 426, 427, 429–431, 433, 436–441, 443–451, 458–461, 463–486, 488, 489, 496, 498, 499, 502, 504, 506, 508, 514–516, 519–523, 525–528, 530–536, 541–548, 550–565, 568–579, 581–595, 598–603, 605, 607–610, 618, 620, 622, 624, 625, 627–642, 648–651, 653, 654, 656, 657, 660–665, 667, 668, 670–705, 707–710, 714, 715, 718–721, 723–729, 735–744, 746–748, 752–756, 758–764, 772–775, 777, 779–789, 796, 797, 808, 812, 813, 840–847, 849, 850, 852–861, 863, 865–872, 874, 875, 877–906, 909–911, 913, 914, 917–931, 933–935, 938–942, 947, 952–955, 960, 962, 964–969, 971, 972, 974–1017, 1021–1024, 1029, 1033–1038, 1040, 1041, 1043–1056, 1058–1064, 1070–1072, 1080–1082
\to ... 259, 260, 498, 644, 645, 1225, 1226, 1251, 1287–1289, 1357, 1358
\today 610
\token 114, 116, 119, 135, 321, 332–334, 337–339, 342, 345, 348, 349, 355, 356, 359, 365, 369, 371, 372, 374, 375, 383, 388, 389, 395, 403, 432, 435–437, 444, 445, 450, 455, 474–477, 480, 482, 485, 496, 502–504, 506, 508, 509, 511, 512, 525, 536, 568, 570, 576–578, 587, 591, 596, 609, 611–620, 625, 646, 652, 710, 728, 729, 734, 741, 743, 747, 748, 752, 755, 758, 765, 767, 774, 776, 780, 781, 847, 848, 860, 877, 878, 880, 900, 901, 910, 926–928, 943, 1017, 1018, 1021–1023, 1040, 1061
\toks@ 269, 271, 272, 505, 538, 539, 1110, 1111, 1128, 1129, 1217–1219, 1222–1224, 1240, 1241, 1244, 1245, 1247, 1248, 1256, 1265, 1266, 1275, 1276, 1279, 1280, 1285, 1293, 1295–1298, 1300, 1302, 1305, 1311–1319, 1342
\TrackIfKnownLanguage 268
\TrackLangAddToCaptions 264, 282
\TrackLangEncodingName 3, 55, 610, 612, 614, 616

\TrackLangProvidesResource . 281
\TrackLangRequireDialect ... 268, 850, 1082
\TrackLangRequireDialectOmitDialectLabel 264
\TrackLangRequireDialectOmitDialectLabel 264, 266, 268
\TrackLangRequireResource .. 263, 264
\TrackLangShowWarningsfalse . 8
\TrackLanguageTag ... 267
\two@digits ... 730, 1174
\twocolumn ... 773, 1193
\TwoLetterIsoCountryCode ... 267
\twopeoplesep 1069, 1069, 1368–1370
\typeout ... 25, 576, 1274, 1291

U

\u .. 54, 59, 61, 97, 98, 105–108, 593, 594
\uccode ... 1316
\undef ... 1086, 1137, 1176, 1177, 1189
\undefined ... 1227, 1228, 1365, 1366
\UnsafeLocation ... 653, 654
\uppercase ... 1315
\Upsilon ... 733, 1175
\upsilon ... 732, 1175
\ur 54, 60–62, 150–157, 166–169, 212, 213
\url ... 806, 807
\use ... 3, 8, 11, 12, 26, 40, 114, 115, 190, 224, 392, 479, 484, 486, 542, 545, 550, 553, 555, 559, 562, 571, 573–575, 579, 581, 602, 610, 625, 647, 655, 656, 659, 662, 669, 699, 709, 713, 722, 723, 742, 752, 766, 786, 787, 790, 798, 877, 879, 880, 899, 901, 911, 922–925, 927, 928, 966–968, 1016–1018, 1020–1023
\Used ... 667, 725, 1172
\USEentry ... 768, 1190
\Useentry ... 768, 770–772, 1190, 1191
\useentry 768, 768, 770–772, 1190, 1191
\USEentrynl ... 770, 1190
\Useentrynl 769, 771, 1190, 1191
\useentrynl . 769, 770, 771, 1190, 1191
\usepackage ... 304, 313
\usetikzlibrary ... 931, 1206
\UTFviii@two@octets ... 273, 734, 1176, 1291, 1327
\UTFviii@two@octets@combine ... 734, 1176

V

\val ... 1139–1141, 1170, 1191, 1192, 1208–1210, 1226, 1291, 1359
\varlabel... 1084
\varepsilon ... 732, 1174
\varphiOmitOnlyRegion.. 733, 1175
\varpi ... 732, 1175
\varrho ... 732, 1175
\varsigma ... 732, 1175
\vartheta ... 732, 1175
\volumename ... 782, 799, 801, 1099, 1106, 1107

W

\wasyeuro ... 193
\whiledo 259–261, 1086, 1087, 1092, 1205, 1222–1224, 1357, 1358
\write ... 781, 814, 1099, 1112, 1131, 1132, 1283, 1284

X

\x . 62, 122, 123, 314–316, 593, 631, 1294
\xappto ... 1173
\xcapitalisewords 656, 713, 714, 1133
\xdef ... 566, 1101, 1129, 1163, 1166, 1171, 1173, 1188, 1195, 1224, 1234, 1250, 1253, 1274, 1365
\xDTLassignfirstmatch . 435, 1237
\xdtlgetrowindex ... 462, 1245
\xDTLinitials ... 215
\Xi ... 733, 1175
\xi ... 732, 1175
\xmakefirstuc ... 656, 657, 713, 714, 716, 754, 768, 770, 1133, 1135, 1185, 1190

Y

\yen ... 1305
\You ... 1038
\you ... 1038
\Your ... 1038
\your ... 1038
\Yours ... 1038
\yours ... 1038

Z

\Z ... 56, 57, 59–61, 93, 122, 151–157, 166–169, 213, 238, 267, 314, 316, 700, 790, 846
\z@ ... 1286
\zeta ... 732, 1174