glossaries-extra.sty v1.38: documented code

Nicola L.C. Talbot

Dickimaw Books
http://www.dickimaw-books.com/

2018-12-01

Abstract

This is the documented code for the glossaries-extra package. See glossaries-extra-manual.pdf for the user manual.

This package is experimental and not stable. It's provided for testing purposes only. Future versions may not be compatible with this version. Once it has stabilised I'll add it to CTAN, at which point compatibility with the first stable version will be maintained.

Contents

1 Main Package Code (glossaries-extra.sty) 5
1.1 Package Initialisation and Options . . . 5
1.2 Extra Utilities . . . 30
1.3 Modifications to Commands Provided by glossaries . . . 42
1.3.1 Existence Checks . . . 47
1.3.2 Document Definitions . . . 56
1.3.3 Existing Glossary Style Modifications . . . 62
1.3.4 Entry Formatting, Hyperlinks and Indexing . . . 67
1.3.5 Entry Counting . . . 104
1.3.6 Acronym Modifications . . . 120
1.3.7 Indexing and Displaying Glossaries . . . 122
1.4 Link Counting . . . 160
1.5 Integration with glossaries-accsupp . . . 162
1.6 Categories . . . 175
1.7 Abbreviations . . . 202
1.7.1 Abbreviation Styles Setup . . . 222
1.7.2 Predefined Styles (Default Font) . . . 225
1.7.3 Predefined Styles (Small Capitals) . . . 242
1.7.4 Predefined Styles (Fake Small Capitals) . . . 256
1.7.5 Predefined Styles (Emphasized) . . . 270
1.7.6 Predefined Styles (User Parentheses Hook) . . . 292
1.7.7 Predefined Styles (Hyphen) . . . 301
1.7.8 Predefined Styles (No Short on First Use) . . . 315
1.8 Using Entries in Headings . . . 318
1.9 Multi-Lingual Support . . . 337
1.10 glossaries-extra-bib2gls.sty . . . 338

2 Style Adjustments (glossaries-extra-stylemods.sty) 383
2.1 Package Initialisation . . . 383
2.2 List-Like Styles . . . 384
2.3 Longtable Styles . . . 387
2.4 Long Ragged Styles . . . 389
2.5 Supertabular Styles . . . 391
2.6 Super Ragged Styles . . . 393
2.7 Inline Style . . . 395
2.8 Tree Styles . . . 395
2.9 Multicolumn Styles . . . 413

3 bookindex style (glossary-bookindex.sty) 419

3.1 Package Initialisation and Options 419

4 longextra styles (glossary-longextra.sty) 425

4.1 Package Initialisation and Options 425

Glossary 448

Change History 449

Index 470

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra}[2018/12/01 1.38 (NLCT)]
```

Requires xkeyval to define package options.

```
\RequirePackage{xkeyval}
```

Requires etoolbox package.

```
\RequirePackage{etoolbox}
```

Has glossaries already been loaded?

```
\@ifpackageloaded{glossaries}
{%
```

Already loaded so pass any options to `\setupglossaries`. This means that the options that can only be set when glossaries is loaded can't be used.

```
  \newcommand{\glsxtr@dooption}[1]{\setupglossaries{#1}}%
  \let\@glsxtr@declareoption\@gls@declareoption
}
{%
```

Not already loaded, so pass options to glossaries.

```
  \newcommand{\glsxtr@dooption}[1]{%
    \PassOptionsToPackage{#1}{glossaries}%
  }%
```

Set the defaults.

```
  \PassOptionsToPackage{toc}{glossaries}
  \PassOptionsToPackage{nopostdot}{glossaries}
  \PassOptionsToPackage{noredefwarn}{glossaries}
  \@ifpackageloaded{polyglossia}%
  {}%
  {%
    \@ifpackageloaded{babel}%
    {\PassOptionsToPackage{translate=babel}{glossaries}}%
    {}%
  }%
  \newcommand*{\@glsxtr@declareoption}[2]{%
    \DeclareOptionX{#1}{#2}%
    \DeclareOption{#1}{#2}%
  }
}
```

Declare package options.

sxtrundefaction

Determines what to do if an entry hasn't been defined. The two arguments are the error or warning message and the help message if an error should be produced.

```
\newcommand*{\glsxtrundefaction}[2]{%
  \@glsxtrundeftag\PackageError{glossaries-extra}{#1}{#2}%
}
```

arnonexistsordo

If user wants undefaction=warn, then glossaries v4.19 is required.

```
\newcommand*{\glsxtr@warnonexistsordo}[1]{}
```

\glsxtrundeftag

Text to display when an entry doesn't exist.

```
\newcommand*{\glsxtrundeftag}{??}
\newcommand*{\@glsxtrundeftag}{}
```

This text is switched on at the start of the document to prevent unwanted text inserted into the preamble if any tests are made before the start of the document.

arn@undefaction

This is how `\glsxtrundefaction` should behave if undefaction=warn is set.

```
\newcommand*{\@glsxtr@warn@undefaction}[2]{%
  \@glsxtrundeftag\GlossariesExtraWarning{#1}%
}
```

err@undefaction

This is how `\glsxtrundefaction` should behave if undefaction=error is set.

```
\newcommand*{\@glsxtr@err@undefaction}[2]{%
  \@glsxtrundeftag\PackageError{glossaries-extra}{#1}{#2}%
}
```

rn@onexistsordo

This is how `\glsxtr@warnonexistsordo` should behave if undefaction=warn is set.

```
\newcommand*{\@glsxtr@warn@onexistsordo}[1]{%
  \GlossariesExtraWarning{\string#1\space hasn't been defined, so
  some errors won't be converted to warnings.
  (This most likely means your version of
  glossaries.sty is below version 4.19.)}%
}
```

f@forglsentries

```
\newcommand*{\@glsxtr@redef@forglsentries}{}
```

f@forglsentries

```
\newcommand*{\@glsxtr@do@redef@forglsentries}{%
  \renewcommand*{\forglsentries}[3][\glsdefaulttype]{%
    \edef\@@glo@list{\csname glolist@##1\endcsname}%
    \ifdefstring{\@@glo@list}{,}%
    {%
      \GlossariesExtraWarning{No entries defined in glossary '##1'}%
    }%
    {%
      \@for##2:=\@@glo@list\do
      {%
        \ifdefempty{##2}{}{##3}%
      }%
    }%
  }%
}%
```

undefaction

```
\define@choicekey{glossaries-extra.sty}{undefaction}%
 [\glsxtr@undefaction@val\glsxtr@undefaction@nr]%
 {warn,error}%
 {%
   \ifcase\glsxtr@undefaction@nr\relax
     \let\glsxtrundefaction\@glsxtr@warn@undefaction
     \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
     \let\@glsxtr@redef@forglsentries\@glsxtr@do@redef@forglsentries
   \or
     \let\glsxtrundefaction\@glsxtr@err@undefaction
     \let\glsxtr@warnonexistsordo\@gobble
     \let\@glsxtr@redef@forglsentries\relax
   \fi
 }
```

To assist bib2gls, v1.08 introduces the record option, which will write information to the aux file whenever an entry needs to be indexed.

\@glsxtr@record Does nothing by default.

```
\newcommand*{\@glsxtr@record}[3]{}
```

lsxtr@recordsee Does nothing by default.

```
\newcommand*{\glsxtr@recordsee}[2]{}
```

ultnumberformat

```
\newcommand*{\@glsxtr@defaultnumberformat}{glsnumberformat}%
```

ultNumberFormat

```
\newcommand*{\GlsXtrSetDefaultNumberFormat}[1]{%
  \renewcommand*{\@glsxtr@defaultnumberformat}{#1}%
}%
```

The record option is somewhat problematic. On the first LaTeX run the entries aren't defined. This isn't as straight-forward as commands like `\cite` since attributes associated with the entry's category may switch off the indexing or the entry's glossary type might require a particular counter. This kind of information can't be determined until the entry has been defined. So there are two different commands here. One that's used if the entry hasn't been defined, which tries to use sensible defaults, and one which is used when the entry has been defined.

`cord@wrglossary` The record=only option sets `\@@do@wrglossary` to this command, which means it's done within `\glsadd` and `\@gls@link`, and so is only done if the entry exists.

```
\newcommand*{\@glsxtr@do@record@wrglossary}[1]{%
  \begingroup
    \ifKV@glslink@noindex
    \else
      \edef\@gls@label{\glsdetoklabel{#1}}%
      \let\glslabel\@gls@label
      \glswriteentry{#1}%
      {%
        \ifdefempty{\@glsxtr@thevalue}%
        {%
          \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
          \else
            \let\theHglsentrycounter\@glsxtr@theHvalue
          \fi
          \glsxtr@saveentrycounter
          \let\@@do@@wrglossary\@glsxtr@dorecord
        }%
        {%
          \let\theglsentrycounter\@glsxtr@thevalue
          \let\theHglsentrycounter\@glsxtr@theHvalue
          \let\@@do@@wrglossary\@glsxtr@dorecordnodefer
        }%
        \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
          \glsxtr@@do@wrglossary{#1}%
        \else
          \@@glsxtrwrglossmark
```

Increment associated counter.

```
          \glsxtr@inc@wrglossaryctr{#1}%
          \@@do@@wrglossary
        \fi
      }%
    \fi
  \endgroup
}
```

`ndex@wrglossary` The record=alsoindex option needs to both record and index.

```
\newcommand*{\glsxtr@do@alsoindex@wrglossary}[1]{%
  \glsxtr@@do@wrglossary{#1}%
  \@glsxtr@do@record@wrglossary{#1}%
}
```

`@@glsxtr@record` The record=only option sets `\@glsxtr@record` to this. This performs the recording if the entry *doesn't exist* and is done at the start of `\@gls@field@link` and commands like `\@gls@` (before the existence test). This means that it disregards the wrgloss key.

The first argument is the option list (as passed in the first optional argument to commands like `\gls`). This allows the noindex setting to be picked up. The second argument is the entry's label. The third argument is the key family (`glslink` in most cases, `glossadd` for `\glsadd`).

```
\newcommand*{\@@glsxtr@record}[3]{%
 \ifglsentryexists{#2}{}%
 {%
   \@@glsxtrwrglossmark
   \begingroup
```

Save the label in case it's needed.

```
     \edef\@gls@label{\glsdetoklabel{#2}}%
     \let\glslabel\@gls@label
     \let\@glsnumberformat\@glsxtr@defaultnumberformat
     \def\@glsxtr@thevalue{}%
     \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
     \let\@glsxtr@org@theHvalue\@glsxtr@theHvalue
```

Entry hasn't been defined, so we'll have to assume it's `\glscounter` by default.

```
     \let\@gls@counter\glscounter
```

Unless the equations option is on and this is inside a numbered maths environment.

```
     \if@glsxtr@equations
       \@glsxtr@use@equation@counter
     \fi
```

Check for default options (which may switch off indexing).

```
     \@gls@setdefault@glslink@opts
```

Implement any pre-key settings.

```
     \csuse{@glsxtr@#3@prekeys}%
```

Assign keys.

```
     \setkeys{#3}{#1}%
```

Implement any post-key settings. Is the auto-add on?

```
     \glsxtr@do@autoadd{#3}%
```

Check post-key hook.

```
     \csuse{@glsxtr@#3@postkeys}%
```

Increment associated counter.

```
     \glsxtr@inc@wrglossaryctr{#2}%
```

Check if noindex option has been used.

```
     \ifKV@glslink@noindex
     \else
       \glswriteentry{#2}%
       {%
```

Check if thevalue has been set.

```
         \ifdefempty{\@glsxtr@thevalue}%
         {%
```

Key thevalue hasn't been set, but check if theHvalue has been set. (Not particularly likely, but allow for it.)

```
           \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
```

```
      \else
       \let\theHglsentrycounter\@glsxtr@theHvalue
      \fi
```

Save the entry counter.

```
      \glsxtr@saveentrycounter
```

Temporarily redefine `\@@do@@wrglossary` for use with `\glsxtr@@do@wrglossary`.

```
      \let\@@do@@wrglossary\@glsxtr@dorecord
     }%
     {%
```

thevalue has been set, so there's no need to defer writing the location value. (If it's dependent on the page counter, the counter key should be set instead.)

```
      \let\theglsentrycounter\@glsxtr@thevalue
      \let\theHglsentrycounter\@glsxtr@theHvalue
      \let\@@do@@wrglossary\@glsxtr@dorecordnodefer
     }%
     \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
      \glsxtr@@do@wrglossary{#2}%
     \else
```

No need to escape special characters.

```
      \@@do@@wrglossary
     \fi
    }%
   \fi
  \endgroup
 }%
}
```

`glslink@prekeys`

```
\newcommand{\@glsxtr@glslink@prekeys}{\glslinkpresetkeys}
```

`lslink@postkeys`

```
\newcommand{\@glsxtr@glslink@postkeys}{\glslinkpostsetkeys}
```

`lossadd@prekeys`

```
\newcommand{\@glsxtr@glossadd@prekeys}{\glsaddpresetkeys}
```

`ossadd@postkeys`

```
\newcommand{\@glsxtr@glossadd@postkeys}{\glsaddpostsetkeys}
```

`glsxtr@dorecord` If record=alsoindex is used, then `\@glslocref` may have been escaped, but this isn't appropriate here.

```
\newcommand*\@glsxtr@dorecord{%
  \global\let\@glsrecordlocref\theglsentrycounter
  \let\@glsxtr@orgprefix\@glo@counterprefix
  \ifx\theglsentrycounter\theHglsentrycounter
   \def\@glo@counterprefix{}%
  \else
   \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
     {\theglsentrycounter}{\theHglsentrycounter}%
   }%
   \@do@gls@getcounterprefix
  \fi
```

Don't protect the `\@glsrecordlocref` from premature expansion. If the counter isn't page then it needs expanding. If the location includes `\thepage` then `\protected@write` will automatically deal with it.

```
  \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
    \@glsxtr@do@nameref@record
     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
     {\@glsrecordlocref}%
  \else
    \protected@write\@auxout{}{\string\glsxtr@record
       {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
       {\@glsrecordlocref}}%
  \fi
  \@glsxtr@counterrecordhook
  \let\@glo@counterprefix\@glsxtr@orgprefix
}
```

dorecordnodefer

As above, but don't defer expansion of location. This uses `\theglsentrycounter` directly for the location rather than `\@glslocref` since there's no need to guard against premature expansion of the page counter.

```
\newcommand*\@glsxtr@dorecordnodefer{%
  \ifx\theglsentrycounter\theHglsentrycounter
    \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
      \@glsxtr@do@nameref@record
         {\@gls@label}{}{\@gls@counter}{\@glsnumberformat}%
         {\theglsentrycounter}%
    \else
      \protected@write\@auxout{}{\string\glsxtr@record
         {\@gls@label}{}{\@gls@counter}{\@glsnumberformat}%
         {\theglsentrycounter}}%
    \fi
  \else
    \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
      {\theglsentrycounter}{\theHglsentrycounter}%
    }%
    \@do@gls@getcounterprefix
    \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
      \@glsxtr@do@nameref@record
         {\@gls@label}{\@glo@counterprefix}{\@gls@counter}%
         {\@glsnumberformat}{\theglsentrycounter}%
    \else
      \protected@write\@auxout{}{\string\glsxtr@record
        {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
        {\theglsentrycounter}}%
    \fi
  \fi
  \@glsxtr@counterrecordhook
}
```

`xtr@ifnum@mmode`

Check if in a numbered maths environment. The amsmath package is automatically loaded by datatool-base, which is required by glossaries, so `\ifst@rred` and `\if@display` should both be defined.

```
\newcommand{\@glsxtr@ifnum@mmode}[2]{%
 \ifmmode
   \ifst@rred
    #2%
   \else
```

Non-amsmath environments and regular inline math mode isn't flagged as starred by amsmath, but we can't use `\mathchoice` in this case as it's not the current style that's relevant. Instead we can use amsmath's `\if@display`. This may not work for environments that aren't provided by amsmath.

```
     \if@display #1\else #2\fi
   \fi
 \else
   #2%
 \fi
}
```

`@nameref@record`

With record=nameref, the current label information is included in the record, but this may not have been defined, so `\csuse` will prevent an undefined control sequence error and just leave the last two arguments blank if there's no information. In the event that a record is in amsmath's align environment `\@currentHref` will be out. There may be other instances where `\@currentHref` is out, so this also saves `\theHglsentrycounter`, which is useful if it can't be obtained by prefixing `\theglsentrycounter`.

```
\newcommand*{\@glsxtr@do@nameref@record}[5]{%
  \gls@ifnotmeasuring
  {%
    \protected@write\@auxout{}{\string\glsxtr@record@nameref
     {#1}{#2}{#3}{#4}{#5}%
     {\csuse{@currentlabelname}}{\csuse{@currentHref}}%
     {\theHglsentrycounter}}%
  }%
}
```

`r@recordcounter`

```
\newcommand*{\@@glsxtr@recordcounter}{%
  \@glsxtr@noop@recordcounter
}
```

`p@recordcounter`

```
\newcommand*{\@glsxtr@noop@recordcounter}[1]{%
  \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
   requires record=only or record=alsoindex package option}{}%
}
```

p@recordcounter

```
\newcommand*{\@glsxtr@op@recordcounter}[1]{%
  \eappto\@glsxtr@counterrecordhook{\noexpand\@glsxtr@docounterrecord{#1}}%
}
```

lsxtr@recordsee Deal with `\glssee` in record mode. (This doesn’t increment the associated counter.)

```
\newcommand*{\@glsxtr@recordsee}[2]{%
 \@@glsxtrwrglossmark
 \def\@gls@xref{#2}%
 \@onelevel@sanitize\@gls@xref
 \protected@write\@auxout{}{\string\glsxtr@recordsee{#1}{\@gls@xref}}%
}
```

srtglossaryunit

```
\newcommand{\printunsrtglossaryunit}{%
  \print@noop@unsrtglossaryunit
}
```

tr@setup@record Initialise.

```
\newcommand*{\glsxtr@setup@record}{\let\@@do@wrglossary\glsxtr@@do@wrglossary}
```

aveentrycounter Only store the entry counter information if the indexing is on.

```
\newcommand*{\glsxtr@indexonly@saveentrycounter}{%
 \ifKV@glslink@noindex
 \else
   \glsxtr@saveentrycounter
 \fi
}
```

addloclistfield

```
\newcommand*{\glsxtr@addloclistfield}{%
 \key@ifundefined{glossentry}{loclist}%
 {%
   \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
   \appto\@gls@keymap{,{loclist}{loclist}}%
   \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
   \appto\@newglossaryentryposthook{%
     \gls@assign@field{}{\@glo@label}{loclist}{\@glo@loclist}%
   }%
   \glssetnoexpandfield{loclist}%
 }%
 {}%
```

The loclist field is just a comma-separated list. The location field is the formatted list.

```
\key@ifundefined{glossentry}{location}%
{%
  \define@key{glossentry}{location}{\def\@glo@location{##1}}%
  \appto\@gls@keymap{,{location}{location}}%
  \appto\@newglossaryentryprehook{\def\@glo@location{}}%
  \appto\@newglossaryentryposthook{%
    \gls@assign@field{}{\@glo@label}{location}{\@glo@location}%
  }%
  \glssetnoexpandfield{location}%
}%
{}%
```

Add a key to store the group heading.

```
\key@ifundefined{glossentry}{group}%
{%
  \define@key{glossentry}{group}{\def\@glo@group{##1}}%
  \appto\@gls@keymap{,{group}{group}}%
  \appto\@newglossaryentryprehook{\def\@glo@group{}}%
  \appto\@newglossaryentryposthook{%
    \gls@assign@field{}{\@glo@label}{group}{\@glo@group}%
  }%
  \glssetnoexpandfield{group}%
}%
{}%
}
```

`@record@setting` Keep track of the record package option.

```
\newcommand*{\@glsxtr@record@setting}{off}
```

`tting@alsoindex`

```
\newcommand*{\@glsxtr@record@setting@alsoindex}{alsoindex}
```

`rd@setting@only`

```
\newcommand*{\@glsxtr@record@setting@only}{only}
```

`setting@nameref`

```
\newcommand*{\@glsxtr@record@setting@nameref}{nameref}
```

`@if@record@only`

```
\newcommand*{\@glsxtr@if@record@only}[2]{%
\ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
 #1%
\else
 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
   #1%
 \else
   #2%
 \fi
 \fi
}
```

ord@setting@off

```
\newcommand*{\@glsxtr@record@setting@off}{off}
```

cord@only@setup Initialisation code for record=only and record=nameref

```
\newcommand*{\@glsxtr@record@only@setup}{%
 \def\glsxtr@setup@record{%
   \@glsxtr@autoseeindexfalse
   \let\@do@seeglossary\@glsxtr@recordsee
   \let\@glsxtr@record\@@glsxtr@record
   \let\@@do@wrglossary\@glsxtr@do@record@wrglossary
   \let\@gls@saveentrycounter\relax
   \let\glsxtrundefaction\@glsxtr@warn@undefaction
   \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
   \glsxtr@addloclistfield
   \renewcommand*{\@glsxtr@autoindexcrossrefs}{}%
   \let\@@glsxtr@recordcounter\@glsxtr@op@recordcounter
   \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
```

Switch off the index suppression for aliased entries. (bib2gls will deal with them.)

```
   \def\glsxtrsetaliasnoindex{}%
```

`\@gls@setupsort@none` was only introduced to glossaries v4.30, so it may not be available. If it's defined, use it to remove the unnecessary overhead of escaping and sanitizing the sort value.

```
   \ifdef\@gls@setupsort@none{\@gls@setupsort@none}{}%
```

Warn about using `\printglossary`:

```
   \def\glsxtrNoGlossaryWarning{\@glsxtr@record@noglossarywarning}%
```

Load glossaries-extra-bib2gls:

```
   \RequirePackage{glossaries-extra-bib2gls}%
 }%
}
```

record Now define the record package option.

```
\define@choicekey{glossaries-extra.sty}{record}
 [\@glsxtr@record@setting\glsxtr@record@nr]%
 {off,only,alsoindex,nameref}%
 [only]%
 {%
   \ifcase\glsxtr@record@nr\relax
```

Don't record.

```
     \def\glsxtr@setup@record{%
       \renewcommand*{\@do@seeglossary}{\@glsxtr@doseeglossary}%
       \renewcommand*{\@glsxtr@record}[3]{}%
       \let\@@do@wrglossary\glsxtr@@do@wrglossary
       \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
```

```
      \let\glsxtrundefaction\@glsxtr@err@undefaction
      \let\glsxtr@warnonexistsordo\@gobble
      \let\@@glsxtr@recordcounter\@glsxtr@noop@recordcounter
      \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
      \undef\glsxtrsetaliasnoindex
    }%
  \or
```

Only record (don’t index).

```
    \@glsxtr@record@only@setup
  \or
```

Record and index. This option doesn’t load glossaries-extra-bib2gls as the sorting is performed by `xindy` or `makeindex`.

```
    \def\glsxtr@setup@record{%
      \renewcommand*{\@do@seeglossary}{\@glsxtr@dosee@alsoindex@glossary}%
      \let\@glsxtr@record\@@glsxtr@record
      \let\@@do@wrglossary\glsxtr@do@alsoindex@wrglossary
      \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
      \let\glsxtrundefaction\@glsxtr@warn@undefaction
      \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
      \glsxtr@addloclistfield
      \let\@@glsxtr@recordcounter\@glsxtr@op@recordcounter
      \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
      \undef\glsxtrsetaliasnoindex
    }%
  \or
```

Only record (don’t index) but also include nameref information.

```
    \@glsxtr@record@only@setup
    \ifundef\hyperlink
    {\GlossariesExtraWarning{You have requested record=nameref but
     the document doesn’t support hyperlinks}}%
    {}%
  \fi
}
```

Version 1.06 changes the docdef option to a choice rather than boolean setting. The available values are: false, true or restricted. The restricted option permits document definitions as long as they occur before the first glossary is displayed.

`lsxtr@docdefval`

The docdef value is stored as an integer: 0 (false), 1 (true) and 2 (restricted).

```
\newcommand*{\@glsxtr@docdefval}{0}
```

Need to provide conditional commands that are backward compatible:

`if@glsxtrdocdef`

```
\newcommand*{\if@glsxtrdocdef}{\ifnum\@glsxtr@docdefval>0 }
```

`lsxtrdocdeftrue`

```
\newcommand*{\@glsxtrdocdeftrue}{\def\@glsxtr@docdefval{1}}
```

`sxtrdocdeffalse`

```
\newcommand*{\@glsxtrdocdeffalse}{\def\@glsxtr@docdefval{0}}
```

`docdef` By default don't allow entries to be defined in the document to encourage the user to define them in the preamble, but if the user is really determined to define them in the document allow them to request this.

```
\define@choicekey{glossaries-extra.sty}{docdef}
 [\@glsxtr@docdefsetting\@glsxtr@docdefval]%
 {false,true,restricted,atom}[true]%
{%
  \ifnum\@glsxtr@docdefval>1\relax
    \renewcommand*{\@glsdoifexistsorwarn}{\glsdoifexists}%
  \else
    \renewcommand*{\@glsdoifexistsorwarn}{\glsdoifexistsorwarn}%
  \fi
}
```

`ocdefrestricted`

```
\newcommand*{\if@glsxtrdocdefrestricted}{\ifnum\@glsxtr@docdefval>1 }
```

`oifexistsorwarn` Need an error to notify user if an undefined entry is being referenced in the glossary for the docdef=restricted option. This is used by `\glossentryname` (but not by `\glossentrydesc` etc as one error per entry is sufficient).

```
\newcommand*{\@glsdoifexistsorwarn}{\glsdoifexistsorwarn}
```

`indexcrossrefs` Automatically index cross references at the end of the document

```
\define@boolkey{glossaries-extra.sty}[@glsxtr]{indexcrossrefs}[true]{%
 \if@glsxtrindexcrossrefs
 \else
  \renewcommand*{\@glsxtr@autoindexcrossrefs}{}%
 \fi
}
```

Switch off since this can increase the build time.

```
\@glsxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

`oindexcrossrefs`

```
\newcommand*{\@glsxtr@autoindexcrossrefs}{\@glsxtrindexcrossrefstrue}
```

`autoseeindex` Provide a boolean option to allow the user to prevent the automatic indexing of the cross-referencing keys see, seealso and alias.

```
\define@boolkey{glossaries-extra.sty}[@glsxtr@]{autoseeindex}[true]{%
}
\@glsxtr@autoseeindextrue
```

equations

Provide a boolean option to automatically switch to the equation counter when in a numbered maths environment.

```
\define@boolkey{glossaries-extra.sty}[@glsxtr@]{equations}[true]{%
}
\@glsxtr@equationsfalse
```

\glsxtr@float

```
\let\glsxtr@float\@float
```

glsxtr@dblfloat

```
\let\glsxtr@dblfloat\@dblfloat
```

floats

Provide a boolean option to automatically switch to the the corresponding counter when in a float.

```
\define@boolkey{glossaries-extra.sty}[@glsxtr@]{floats}[true]{%
  \if@glsxtr@floats
   \renewcommand*{\@float}[1]{\renewcommand{\glscounter}{##1}\glsxtr@float{##1}}%
   \renewcommand*{\@dblfloat}[1]{\renewcommand{\glscounter}{##1}\glsxtr@dblfloat{##1}}%
  \else
   \let\@float\glsxtr@float
   \let\@dblfloat\glsxtr@dblfloat
  \fi
}
\@glsxtr@floatsfalse
```

iesExtraWarning

Allow users to suppress warnings.

```
\newcommand*{\GlossariesExtraWarning}[1]{\PackageWarning{glossaries-extra}{#1}}
```

raWarningNoLine

Allow users to suppress warnings.

```
\newcommand*{\GlossariesExtraWarningNoLine}[1]{%
 \PackageWarningNoLine{glossaries-extra}{#1}}
```

```
\@glsxtr@declareoption{nowarn}{%
  \let\GlossariesExtraWarning\@gobble
  \let\GlossariesExtraWarningNoLine\@gobble
  \glsxtr@dooption{nowarn}%
}
```

xtr@defpostpunc

Redefines `\glspostdescription`. The postdot and nopostdot options will have to redefine this.

```
\newcommand*{\@glsxtr@defpostpunc}{}
```

postdot

Shortcut for nopostdot=false

```
\@glsxtr@declareoption{postdot}{%
  \glsxtr@dooption{nopostdot=false}%
  \renewcommand*{\@glsxtr@defpostpunc}{%
    \renewcommand*{\glspostdescription}{%
      \ifglsnopostdot\else.\spacefactor\sfcode`\. \fi}%
  }%
}
```

nopostdot Needs to redefine \@glsxtr@defpostpunc

```
\define@choicekey{glossaries-extra.sty}{nopostdot}{true,false}[true]{%
  \glsxtr@dooption{nopostdot=#1}%
  \renewcommand*{\@glsxtr@defpostpunc}{%
    \renewcommand*{\glspostdescription}{%
      \ifglsnopostdot\else.\spacefactor\sfcode`\. \fi}%
  }%
}
```

postpunc Set the post-description punctuation. This also sets the \ifglsnopostdot conditional, which now indicates if the post-description punctuation has been suppressed.

```
\define@key{glossaries-extra.sty}{postpunc}{%
  \glsxtr@dooption{nopostdot=false}%
  \ifstrequal{#1}{dot}%
  {%
    \renewcommand*{\@glsxtr@defpostpunc}{%
      \renewcommand*{\glspostdescription}{.\spacefactor\sfcode`\. }%
    }%
  }%
  {%
    \ifstrequal{#1}{comma}%
    {%
      \renewcommand*{\@glsxtr@defpostpunc}{%
        \renewcommand*{\glspostdescription}{,}%
      }%
    }%
    {%
      \ifstrequal{#1}{none}%
      {%
        \glsxtr@dooption{nopostdot=true}%
        \renewcommand*{\@glsxtr@defpostpunc}{%
          \renewcommand*{\glspostdescription}{}%
        }%
      }%
      {%
        \renewcommand*{\@glsxtr@defpostpunc}{%
          \renewcommand*{\glspostdescription}{#1}%
        }%
      }%
    }%
  }%
}
```

glsxtrabbrvtype Glossary type for abbreviations.

```
\newcommand*{\glsxtrabbrvtype}{\glsdefaulttype}
```

bbreviationsdef Set by abbreviations option.

```
\newcommand*{\@glsxtr@abbreviationsdef}{}
```

bbreviationsdef

```
\newcommand*{\@glsxtr@doabbreviationsdef}{%
  \@ifpackageloaded{babel}%
  {\providecommand{\abbreviationsname}{\acronymname}}%
  {\providecommand{\abbreviationsname}{Abbreviations}}%
  \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
  \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
  \newcommand*{\printabbreviations}[1][]{%
    \printglossary[type=\glsxtrabbrvtype,##1]%
  }%
  \disable@keys{glossaries-extra.sty}{abbreviations}%
```

If the acronym option hasn't been used, change `\acronymtype` to `\glsxtrabbrvtype`.

```
  \ifglsacronym
  \else
    \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
  \fi
}%
```

abbreviations If abbreviations, create a new glossary type for abbreviations.

```
\@glsxtr@declareoption{abbreviations}{%
  \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
}
```

iationShortcuts Enable shortcut commands for the abbreviations. Unlike the analogous command provided by glossaries, this uses `\newcommand` instead of `\let` as a safety feature (except for `\newabbr` which is also provided with `\GlsXtrDefineAcShortcuts`).

```
\newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
  \newcommand*{\ab}{\cgls}%
  \newcommand*{\abp}{\cglspl}%
  \newcommand*{\as}{\glsxtrshort}%
  \newcommand*{\asp}{\glsxtrshortpl}%
  \newcommand*{\al}{\glsxtrlong}%
  \newcommand*{\alp}{\glsxtrlongpl}%
  \newcommand*{\af}{\glsxtrfull}%
  \newcommand*{\afp}{\glsxtrfullpl}%
  \newcommand*{\Ab}{\cGls}%
  \newcommand*{\Abp}{\cGlspl}%
  \newcommand*{\As}{\Glsxtrshort}%
  \newcommand*{\Asp}{\Glsxtrshortpl}%
  \newcommand*{\Al}{\Glsxtrlong}%
  \newcommand*{\Alp}{\Glsxtrlongpl}%
  \newcommand*{\Af}{\Glsxtrfull}%
  \newcommand*{\Afp}{\Glsxtrfullpl}%
  \newcommand*{\AB}{\cGLS}%
  \newcommand*{\ABP}{\cGLSpl}%
  \newcommand*{\AS}{\GLSxtrshort}%
  \newcommand*{\ASP}{\GLSxtrshortpl}%
  \newcommand*{\AL}{\GLSxtrlong}%
  \newcommand*{\ALP}{\GLSxtrlongpl}%
  \newcommand*{\AF}{\GLSxtrfull}%
  \newcommand*{\AFP}{\GLSxtrfullpl}%

  \providecommand*{\newabbr}{\newabbreviation}%
```

Disable this command after it's been used.

```
  \let\GlsXtrDefineAbbreviationShortcuts\relax
}
```

`fineAcShortcuts` Enable shortcut commands for the abbreviations, but uses the analogous commands provided by glossaries.

```
\newcommand*{\GlsXtrDefineAcShortcuts}{%
  \newcommand*{\ac}{\cgls}%
  \newcommand*{\acp}{\cglspl}%
  \newcommand*{\acs}{\glsxtrshort}%
  \newcommand*{\acsp}{\glsxtrshortpl}%
  \newcommand*{\acl}{\glsxtrlong}%
  \newcommand*{\aclp}{\glsxtrlongpl}%
  \newcommand*{\acf}{\glsxtrfull}%
  \newcommand*{\acfp}{\glsxtrfullpl}%
  \newcommand*{\Ac}{\cGls}%
  \newcommand*{\Acp}{\cGlspl}%
  \newcommand*{\Acs}{\Glsxtrshort}%
  \newcommand*{\Acsp}{\Glsxtrshortpl}%
  \newcommand*{\Acl}{\Glsxtrlong}%
  \newcommand*{\Aclp}{\Glsxtrlongpl}%
  \newcommand*{\Acf}{\Glsxtrfull}%
  \newcommand*{\Acfp}{\Glsxtrfullpl}%
  \newcommand*{\AC}{\cGLS}%
  \newcommand*{\ACP}{\cGLSpl}%
  \newcommand*{\ACS}{\GLSxtrshort}%
  \newcommand*{\ACSP}{\GLSxtrshortpl}%
  \newcommand*{\ACL}{\GLSxtrlong}%
  \newcommand*{\ACLP}{\GLSxtrlongpl}%
  \newcommand*{\ACF}{\GLSxtrfull}%
  \newcommand*{\ACFP}{\GLSxtrfullpl}%

  \providecommand*{\newabbr}{\newabbreviation}%
```

Disable this command after it's been used.

```
  \let\GlsXtrDefineAcShortcuts\relax
}
```

`eOtherShortcuts` Similarly provide shortcut versions for the commands provided by the symbols and numbers options.

```
\newcommand*{\GlsXtrDefineOtherShortcuts}{%
```

```
 \newcommand*{\newentry}{\newglossaryentry}%
 \ifdef\printsymbols
 {%
   \newcommand*{\newsym}{\glsxtrnewsymbol}%
 }{}%
 \ifdef\printnumbers
 {%
   \newcommand*{\newnum}{\glsxtrnewnumber}%
 }{}%
 \let\GlsXtrDefineOtherShortcuts\relax
}
```

Always use the long forms, not the shortcuts, where portability is an issue. (For example, when defining entries in a file that may be input by multiple documents.)

`@setupshortcuts` Command used to set the shortcuts option.

```
\newcommand*{\@glsxtr@setupshortcuts}{}
```

`tr@shortcutsval` Store the value of the shortcuts option. (Needed by bib2gls.)

```
\newcommand*{\@glsxtr@shortcutsval}{\ifglsacrshortcuts acro\else none\fi}%
```

`shortcuts` Provide shortcuts option. Unlike the glossaries version, this is a choice rather than a boolean key but it also provides shortcuts=true and shortcuts=false, which are equivalent to shortcuts=all and shortcuts=none. Multiple use of this option in the *same* option list will override each other. New to v1.17: shortcuts=ac which implements `\GlsXtrDefineAcShortcuts` (not included in shortcuts=all as it conflicts with other shortcuts).

```
\define@choicekey{glossaries-extra.sty}{shortcuts}%
 [\@glsxtr@shortcutsval\@glsxtr@shortcutsnr]%
 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
   \ifcase\@glsxtr@shortcutsnr\relax % acronyms
     \renewcommand*{\@glsxtr@setupshortcuts}{%
       \glsacrshortcutstrue
       \DefineAcronymSynonyms
     }%
   \or % acro
     \renewcommand*{\@glsxtr@setupshortcuts}{%
       \glsacrshortcutstrue
       \DefineAcronymSynonyms
     }%
   \or % abbreviations
     \renewcommand*{\@glsxtr@setupshortcuts}{%
       \GlsXtrDefineAbbreviationShortcuts
     }%
   \or % abbr
     \renewcommand*{\@glsxtr@setupshortcuts}{%
       \GlsXtrDefineAbbreviationShortcuts
     }%
   \or % other
```

```
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \GlsXtrDefineOtherShortcuts
    }%
  \or % all
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \glsacrshortcutstrue

      \GlsXtrDefineAcShortcuts
      \GlsXtrDefineAbbreviationShortcuts
      \GlsXtrDefineOtherShortcuts
    }%
  \or % true
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \glsacrshortcutstrue

      \GlsXtrDefineAcShortcuts
      \GlsXtrDefineAbbreviationShortcuts
      \GlsXtrDefineOtherShortcuts
    }%

  \or % ac
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \glsacrshortcutstrue
      \GlsXtrDefineAcShortcuts
    }%
```

Leave none and false as last option.

```
  \else % none, false
    \renewcommand*{\@glsxtr@setupshortcuts}{}%
  \fi
}
```

lsxtr@doaccsupp

```
\newcommand*{\@glsxtr@doaccsupp}{}
```

accsupp If accsupp, load glossaries-accsupp package.

```
\@glsxtr@declareoption{accsupp}{%
 \renewcommand*{\@glsxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}
```

GlossaryWarning Warning text displayed in document if the external glossary file given by the argument is missing.

```
\newcommand{\glsxtrNoGlossaryWarning}[1]{%
  \GlossariesExtraWarning{Glossary ‘#1’ is missing}%
  \@glsxtr@defaultnoglossarywarning{#1}%
}
```

omissingglstext If true, suppress the text and warning produced if the external glossary file is missing.

```
\define@choicekey{glossaries-extra.sty}{nomissingglstext}
 [\@glsxtr@nomissingglstextval\@glsxtr@nomissingglstextnr]%
```

```
{true,false}[true]{%
  \ifcase\@glsxtr@nomissingglstextnr\relax % true
    \renewcommand{\glsxtrNoGlossaryWarning}[1]{\null}%
  \else % false
    \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
      \@glsxtr@defaultnoglossarywarning{#1}%
    }%
  \fi
}
```

Provide option to load glossaries-extra-stylemods (Deferred to the end.)

xtr@redefstyles

```
\newcommand*{\@glsxtr@redefstyles}{}
```

stylemods

```
\define@key{glossaries-extra.sty}{stylemods}[default]{%
  \ifstrequal{#1}{default}%
  {%
    \renewcommand*{\@glsxtr@redefstyles}{%
      \RequirePackage{glossaries-extra-stylemods}}%
  }%
  {%
    \ifstrequal{#1}{all}%
    {%
      \renewcommand*{\@glsxtr@redefstyles}{%
        \PassOptionsToPackage{all}{glossaries-extra-stylemods}%
        \RequirePackage{glossaries-extra-stylemods}%
      }%
    }%
    {%
      \renewcommand*{\@glsxtr@redefstyles}{}%
      \@for\@glsxtr@tmp:=#1\do{%
        \IfFileExists{glossary-\@glsxtr@tmp.sty}%
        {%
          \eappto\@glsxtr@redefstyles{%
            \noexpand\RequirePackage{glossary-\@glsxtr@tmp}}%
        }%
        {%
           \PackageError{glossaries-extra}%
             {Glossaries style package ‘glossary-\@glsxtr@tmp.sty’
              doesn’t exist (did you mean to use the ‘style’ key?)}%
             {The list of values (#1) in the ‘stylemods’ key should
              match the glossary-xxx.sty files provided with
              glossaries.sty}%
        }%
      }%
      \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
   }
  }%
}
```

glsxtr@do@style

```
\newcommand*{\@glsxtr@do@style}{}
```

style Since the stylemods option can automatically load extra style packages, deal with the style option after those packages have been loaded.

```
\define@key{glossaries-extra.sty}{style}{%
```

Defer actual style change:

```
 \renewcommand*{\@glsxtr@do@style}{%
```

Set this as the default style:

```
   \setkeys{glossaries.sty}{style={#1}}%
```

Set this style:

```
   \setglossarystyle{#1}%
 }%
}
```

c@wrglossaryctr Increments the associated counter if enabled. Does nothing by default. The optional argument is the entry label in case it's required, but the wrglossary counter is globally used by all entries.

```
\newcommand*{\glsxtr@inc@wrglossaryctr}[1]{}
```

ocationHyperlink

`\glsxtrinternallocationhyperlink{⟨counter⟩}{⟨prefix⟩}{⟨location⟩}`

The first two arguments are always control sequences.

```
\newcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
  \glsxtrhyperlink{#1#2#3}{#3}%
}
```

cationhyperlink

```
\newcommand*{\@glsxtr@wrglossary@locationhyperlink}[3]{%
  \pageref{wrglossary.#3}%
}
```

indexcounter Define the wrglossary counter that's incremented every time an entry is indexed, except for cross-references. This is designed for use with `bib2gls` v1.4+. It can work with the other indexing methods but it will interfere with the number list collation. This option automatically implements counter=wrglossary.

Since glossaries automatically loads amsmath, there may be a problem if the indexing occurs in the equation environment, because only one `\label` is allowed in each instance of that environment. It's best to change the counter when in maths mode.

```
\@glsxtr@declareoption{indexcounter}{%
```

```
  \glsxtr@dooption{counter=wrglossary}%
  \ifundef\c@wrglossary
  {%
    \newcounter{wrglossary}%
    \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
  }%
  {}%
  \renewcommand*{\glsxtr@inc@wrglossaryctr}[1]{%
```

Only increment if the current counter is wrglossary.

```
    \ifdefstring\@gls@counter{wrglossary}%
    {%
      \refstepcounter{wrglossary}%
      \label{wrglossary.\thewrglossary}%
    }%
    {}%
  }%
  \renewcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
    \ifdefstring\glsentrycounter{wrglossary}%
    {%
      \@glsxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
    }%
    {\glsxtrhyperlink{##1##2##3}{##3}}%
  }%
}
```

sxtrwrglossmark Marks the place where indexing occurs. Does nothing by default.

```
\newcommand*{\@glsxtrwrglossmark}{}
```

sxtrwrglossmark Since \glsadd can be used in the preamble, this action needs to be disabled until the start of the document.

```
\newcommand*{\@@glsxtrwrglossmark}{}
\AtBeginDocument{\renewcommand*{\@@glsxtrwrglossmark}{\@glsxtrwrglossmark}}
```

sxtrwrglossmark Does nothing by default.

```
\newcommand*{\glsxtrwrglossmark}{\ensuremath{\cdot}}
```

debug Provide extra debug options.

```
\define@choicekey{glossaries-extra.sty}{debug}
 [\@glsxtr@debugval\@glsxtr@debugnr]%
 {true,false,showtargets,showwrgloss,all}[true]{%
   \ifcase\@glsxtr@debugnr\relax % true
    \glsxtr@dooption{debug=true}%
    \renewcommand*{\@glsxtrwrglossmark}{}%
   \or % false
    \glsxtr@dooption{debug=false}%
    \renewcommand*{\@glsxtrwrglossmark}{}%
   \or % showtargets
    \glsxtr@dooption{debug=showtargets}%
   \or % showwrgloss
    \glsxtr@dooption{debug=true}%
    \renewcommand*{\@glsxtrwrglossmark}{\glsxtrwrglossmark}%
   \or % all
    \glsxtr@dooption{debug=showtargets}%
    \renewcommand*{\@glsxtrwrglossmark}{\glsxtrwrglossmark}%
   \fi
 }
```

Pass all other options to glossaries.

```
\DeclareOptionX*{%
 \expandafter\glsxtr@dooption\expandafter{\CurrentOption}}
```

Process options.

```
\ProcessOptionsX
```

Load glossaries if not already loaded.

```
\RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
\@glsxtr@doaccsupp
```

Redefine `\glspostdescription` if required.

```
\@glsxtr@defpostpunc
```

`\glsshowtarget` This command was introduced to glossaries v4.32 so it may not be defined. Therefore it's defined here using `\def`.

```
\def\glsshowtarget#1{%
 \glsxtrtitleorpdforheading
 {%
   \ifmmode
     \texttt{\small [#1]}%
   \else
     \ifinner
       \texttt{\small [#1]}%
     \else
       \marginpar{\texttt{\small #1}}%
     \fi
   \fi
 }%
 {[#1]}%
 {\texttt{\small [#1]}}%
}
```

`g@doseeglossary` Save original definition of `\@do@seeglossary`

```
\let\@glsxtr@org@doseeglossary\@do@seeglossary
```

`r@doseeglossary` This doesn't increment the associated counter.

```
\newcommand*{\@glsxtr@doseeglossary}[2]{%
  \glsdoifexists{#1}%
  {%
    \@@glsxtrwrglossmark
    \@glsxtr@org@doseeglossary{#1}{#2}%
  }%
}
```

`oindex@glossary`

```
\newcommand*{\@glsxtr@dosee@alsoindex@glossary}[2]{%
  \@glsxtr@recordsee{#1}{#2}%
  \@glsxtr@doseeglossary{#1}{#2}%
}
```

`@org@gloautosee` Save and restore original definition of `\@glo@autosee`. (That command may not be defined as it was only introduced to glossaries v4.30, in which case the synonym won't be defined either.)

```
\let\@glsxtr@org@gloautosee\@glo@autosee
```

Check if user tried autoseeindex=false when it can't be supported.

```
\if@glsxtr@autoseeindex
\else
  \ifdef\@glsxtr@org@gloautosee
  {}%
  {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
   option requires at least v4.30 of glossaries.sty}%
   {You need to update the glossaries.sty package}%
  }
\fi
```

`\@glo@autosee` If `\@glo@autosee` has been defined (glossaries v4.30 onwards), redefine it to test the autoseeindex option.

```
\ifdef\@glo@autosee
{%
  \renewcommand*{\@glo@autosee}{%
    \if@glsxtr@autoseeindex\@glsxtr@org@gloautosee\fi}%
}%
{}
```

`checkseeallowed` Don't prohibit the use of the see key before the indexing files have been opened if the automatic see indexing has been disabled, since it's no longer an issue.

```
\renewcommand*{\gls@checkseeallowed}{%
 \if@glsxtr@autoseeindex\@gls@see@noindex\fi
}
```

Define abbreviations glossaries if required.

```
\@glsxtr@abbreviationsdef
\let\@glsxtr@abbreviationsdef\relax
```

Setup shortcuts if required.

```
\@glsxtr@setupshortcuts
```

Redefine `\@glsxtr@redef@forglsentries` if required.

```
\@glsxtr@redef@forglsentries
```

`ariesextrasetup` Allow user to set options after the package has been loaded. First modify `\glsxtr@dooption` so that it now uses `\setupglossaries`:

```
\renewcommand{\glsxtr@dooption}[1]{\setupglossaries{#1}}%
```

Now define the user command:

```
\newcommand*{\glossariesextrasetup}[1]{%
  \let\glsxtr@setup@record\relax
  \let\@glsxtr@setupshortcuts\relax
  \let\@glsxtr@redef@forglsentries\relax
  \setkeys{glossaries-extra.sty}{#1}%
  \@glsxtr@abbreviationsdef
  \let\@glsxtr@abbreviationsdef\relax
  \@glsxtr@setupshortcuts
  \glsxtr@setup@record
  \@glsxtr@redef@forglsentries
}
```

`@@do@wrglossary` Save original definition of `\@@do@wrglossary`.

```
\let\glsxtr@org@@do@wrglossary\@@do@wrglossary
```

`@@do@wrglossary` The new version adds code that can show a marker for debugging and increments the associated counter if enabled.

```
\newcommand*{\glsxtr@@do@wrglossary}[1]{%
 \@@glsxtrwrglossmark
 \glsxtr@inc@wrglossaryctr{#1}%
 \glsxtr@org@@do@wrglossary{#1}%
}
```

`aveentrycounter` Save original definition of `\@gls@saveentrycounter`.

```
\let\glsxtr@saveentrycounter\@gls@saveentrycounter
```

`aveentrycounter` Change `\@gls@saveentrycounter` so that it only stores the entry counter information if the indexing is on.

```
\let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
```

`etcounterprefix` This command is provided by the base glossaries package, but is redefined here. The standard indexing methods don't directly store the hypertarget but instead need to split it into the counter, prefix and location parts, which can be reconstituted in the location list. Unfortunately, not all targets are in this form, so the links fail. With record=nameref, the complete target name can be saved, so this modification adjusts the warning.

```
\renewcommand*\@gls@getcounterprefix[2]{%
  \protected@edef\@gls@thisloc{#1}\protected@edef\@gls@thisHloc{#2}%
  \ifx\@gls@thisloc\@gls@thisHloc
    \def\@glo@counterprefix{}%
  \else
```

```
    \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
      \def\@glo@tmp{##2}%
      \ifx\@glo@tmp\@empty
        \def\@glo@counterprefix{}%
      \else
        \def\@glo@counterprefix{##1}%
      \fi
    }%
    \@gls@get@counterprefix#2.#1\end@getprefix
```

Warn if no prefix can be formed, unless record=nameref.

```
    \ifx\@glo@counterprefix\@empty
      \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
      \else
        \GlossariesExtraWarning{Hyper target ‘#2’ can’t be formed by
         prefixing^^Jlocation ‘#1’. You need to modify the
         definition of \string\theH\@gls@counter^^Jotherwise you
         will get the warning: "‘name{\@gls@counter.#1}’ has been^^J
         referenced but does not exist"%
         \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
         . You may want to consider using record=nameref instead%
         \fi}%
      \fi
    \fi
  \fi
}
```

Provide script dialect hook (does nothing unless redefined by glossaries-extra-bib2gls).

sxtrdialecthook

```
\newcommand*{\@glsxtrdialecthook}{}
```

Set up record option if required.

```
\glsxtr@setup@record
```

Disable preamble-only options and switch on the undefined tag at the start of the document.

```
\AtBeginDocument{%
  \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
  \def\@glsxtrundeftag{\glsxtrundeftag}%
}
```

1.2 Extra Utilities

nusedOrUndefined

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

Does ⟨*true*⟩ if the entry given by ⟨*label*⟩ is either undefined or hasn't been used (or has had the first use flag reset).

```
\newcommand*{\GlsXtrIfUnusedOrUndefined}[3]{%
  \ifglsentryexists{#1}%
  {\ifbool{glo@\glsdetoklabel{#1}@flag}{#3}{#2}}%
  {#2}%
}
```

`rifemptyglossary`

`\glsxtrifemptyglossary{⟨type⟩}{⟨true⟩}{⟨false⟩}`

Provide command to determine if any entries have been added to the glossary (where the glossary label is provided in the first argument). The entries are stored in the comma-separated list `\glolist@⟨type⟩`. If this hasn't been defined, the glosary doesn't exist. If it has been defined and is simply a comma, the glossary exists and is empty. (It's initialised to a comma.)

```
\newcommand{\glsxtrifemptyglossary}[3]{%
  \ifcsdef{glolist@#1}%
  {%
    \ifcsstring{glolist@#1}{,}{#2}{#3}%
  }%
  {%
    \glsxtrundefaction{Glossary type ‘#1’ doesn’t exist}{}%
    #2%
  }%
}
```

`xtrifkeydefined` Tests if the key given in the first argument has been defined.

```
\newcommand*{\glsxtrifkeydefined}[3]{%
  \key@ifundefined{glossentry}{#1}{#3}{#2}%
}
```

`ovidestoragekey` Like `\glsaddstoragekey` but does nothing if the key has already been defined.

```
\newcommand*{\glsxtrprovidestoragekey}{%
  \@ifstar\@sglsxtr@provide@storagekey\@glsxtr@provide@storagekey
}
```

`vide@storagekey` Unstarred version.

```
\newcommand*{\@glsxtr@provide@storagekey}[3]{%
  \key@ifundefined{glossentry}{#1}%
  {%
    \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
    \appto\@gls@keymap{,{#1}{#1}}%
    \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
    \appto\@newglossaryentryposthook{%
      \letcs{\@glo@tmp}{@glo@#1}%
      \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
    }%
```

Allow the user to omit the user level command if they only intended fetching the value with `\glsxtrusefield`

```
    \ifblank{#3}
    {}%
    {%
      \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
    }%
  }%
  {%
```

Provide the no-link command if not already defined.

```
    \ifblank{#3}
    {}%
    {%
      \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
    }%
  }%
}
```

vide@storagekey

Starred version.

```
\newcommand*{\s@glsxtr@provide@storagekey}[1]{%
  \key@ifundefined{glossentry}{#1}%
  {%
    \expandafter\newcommand\expandafter*\expandafter
     {\csname gls@assign@#1@field\endcsname}[2]{%
       \@@gls@expand@field{##1}{#1}{##2}%
     }%
  }%
  {}%
  \@glsxtr@provide@addstoragekey{#1}%
}
```

The name of a text-block control sequence can be stored in a field (given by `\GlsXtrFmtField`). This command can then be used with `\glsxtrfmt[`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}` which effectively does `\glslink[`⟨*options*⟩`]{`⟨*label*⟩`}`{⟨*cs*⟩`{`⟨*text*⟩`}`} If the field hasn't been set for that entry just ⟨*text*⟩ is done.

\GlsXtrFmtField

```
\newcommand{\GlsXtrFmtField}{useri}
```

tDefaultOptions

```
\newcommand{\GlsXtrFmtDefaultOptions}{noindex}
```

\glsxtrfmt

The post-link hook isn't done. This now has a starred form that checks for a final optional argument.

```
\newrobustcmd*{\glsxtrfmt}{\@ifstar\s@glsxtrfmt\@glsxtrfmt}
```

`\@glsxtrfmt` Unstarred form.

```
\newcommand*{\@glsxtrfmt}[3][]{\@@glsxtrfmt{#1}{#2}{#3}{}}
```

`\s@glsxtrfmt` Starred form.

```
\newcommand*{\s@glsxtrfmt}[3][]{%
 \new@ifnextchar[{\s@@glsxtrfmt{#1}{#2}{#3}}%
  {\@@glsxtrfmt{#1}{#2}{#3}{}}%
}
```

`\s@@glsxtrfmt` Pick up final optional argument.

```
\def\s@@glsxtrfmt#1#2#3[#4]{\@@glsxtrfmt{#1}{#2}{#3}{#4}}
```

`\@@glsxtrfmt` Actual inner working.

```
\newcommand*{\@@glsxtrfmt}[4]{%
```

Since there's no post-link hook to worry about, grouping can be added to provide some protection against nesting (but in general nested link text should be avoided).

```
 \begingroup
   \def\glslabel{#2}%
   \glsdoifexistsordo{#2}%
   {%
     \ifglshasfield{\GlsXtrFmtField}{#2}%
     {%
       \let\do@gls@link@checkfirsthyper\relax
       \expandafter\@gls@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
         {\glsxtrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
     }%
     {\glsxtrfmtdisplay{@firstofone}{#3}{#4}}%
   }%
   {%
```

Has the default `noindex` been counteracted? If so, this needs `\glsadd` in case `bib2gls` needs to pick up the record.

```
     \begingroup
       \@gls@setdefault@glslink@opts
       \setkeys{glslink}{\GlsXtrFmtDefaultOptions,#1}%
       \ifKV@glslink@noindex\else\glsadd{#2}\fi
     \endgroup
     \glsxtrfmtdisplay{@firstofone}{#3}{#4}%
   }%
 \endgroup
}
```

`lsxtrfmtdisplay` The command used internally by `\glsxtrfmt` to do the actual formatting. The first argument is the control sequence name, the second is the control sequence's argument, the third is the inserted material (if starred form used).

```
\newcommand{\glsxtrfmtdisplay}[3]{\csuse{#1}{#2}#3}
```

`\glsxtrentryfmt` No link or indexing.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsxtrentryfmt}[2]{%
    \texorpdfstring{\@glsxtrentryfmt{#1}{#2}}{#2}%
  }
}
{
  \newcommand*{\glsxtrentryfmt}{\@glsxtrentryfmt}
}
```

`@glsxtrentryfmt`

```
\newrobustcmd*{\@glsxtrentryfmt}[2]{%
 \glsdoifexistsordo{#1}%
 {%
   \ifglshasfield{\GlsXtrFmtField}{#1}%
   {%
     \csuse{\glscurrentfieldvalue}{#2}%
   }%
   {#2}%
 }%
 {#2}%
}
```

`xtrfieldlistadd` If a field stores an etoolbox internal list (e.g. loclist) then this macro provides a convenient way of adding to the list via etoolbox's `\listcsadd`. The first argument is the entry's label, the second is the field label and the third is the element to add to the list.

```
\newcommand*{\glsxtrfieldlistadd}[3]{%
  \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
}
```

`trfieldlistgadd` Similarly but uses `\listcsgadd`.

```
\newcommand*{\glsxtrfieldlistgadd}[3]{%
  \listcsgadd{glo@\glsdetoklabel{#1}@#2}{#3}%
}
```

`trfieldlisteadd` Similarly but uses `\listcseadd`.

```
\newcommand*{\glsxtrfieldlisteadd}[3]{%
  \listcseadd{glo@\glsdetoklabel{#1}@#2}{#3}%
}
```

`trfieldlistxadd` Similarly but uses `\listcsxadd`.

```
\newcommand*{\glsxtrfieldlistxadd}[3]{%
  \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
}
```

Now provide commands to iterate over these lists.

fielddolistloop

```
\newcommand*{\glsxtrfielddolistloop}[2]{%
  \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
}
```

ieldforlistloop

```
\newcommand*{\glsxtrfieldforlistloop}[3]{%
  \forlistcsloop{#3}{glo@\glsdetoklabel{#1}@#2}%
}
```

List element tests:

trfieldifinlist First argument label, second argument field, third argument item, fourth true part and fifth false part.

```
\newcommand*{\glsxtrfieldifinlist}[5]{%
  \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
}
```

rfieldxifinlist Expands item.

```
\newcommand*{\glsxtrfieldxifinlist}[5]{%
  \xifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
}
```

lsxtrforcsvfield

`\glsxtrforcsvfield{⟨label⟩}{⟨field⟩}{⟨cs handler⟩}`

```
\newcommand*{\glsxtrforcsvfield}[3]{%
 \@glsxtrifhasfield{#2}{#1}%
 {%
   \let\glsxtrendfor\@endfortrue
   \@for\@glsxtr@label:=\glscurrentfieldvalue\do
   {\expandafter#3\expandafter{\@glsxtr@label}}}%
 {}%
}
```

lsxtrifhasfield A simpler alternative to `\ifglshasfield` that doesn't complain if the entry or the field doesn't exist. (No mapping is used.) Grouping is added to the unstarred version allow for nested use.

```
\newrobustcmd{\glsxtrifhasfield}{%
  \@ifstar{\s@glsxtrifhasfield}{\@glsxtrifhasfield}%
}
```

lsxtrifhasfield Unstarred version adds grouping.

```
\newcommand{\@glsxtrifhasfield}[4]{%
  {\s@glsxtrifhasfield{#1}{#2}{#3}{#4}}%
}
```

lsxtrifhasfield Starred version omits grouping.

```
\newcommand{\s@glsxtrifhasfield}[4]{%
  \letcs{\glscurrentfieldvalue}{glo@\glsdetoklabel{#2}@#1}%
  \ifundef\glscurrentfieldvalue
  {#4}%
  {%
   \ifdefempty\glscurrentfieldvalue{#4}{#3}%
  }%
}
```

rIfFieldNonZero Designed for numeric fields.

```
\newcommand{\GlsXtrIfFieldNonZero}[4]{%
  \GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
}
```

sXtrIfFieldEqNum

`\GlsXtrIfFieldEqNum{⟨field⟩}{⟨label⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}`

Designed for numeric fields.

```
\newcommand{\GlsXtrIfFieldEqNum}[5]{%
  \GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
}
```

XtrIfFieldCmpNum

`\GlsXtrIfFieldCmpNum{⟨field⟩}{⟨label⟩}{⟨comparison⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}`

Designed for numeric fields.

```
\newcommand{\GlsXtrIfFieldCmpNum}[6]{%
  {%
    \letcs{\glscurrentfieldvalue}{glo@\glsdetoklabel{#2}@#1}%
    \ifundef\glscurrentfieldvalue
    {\def\glscurrentfieldvalue{0}}%
    {%
     \ifdefempty\glscurrentfieldvalue
     {\def\glscurrentfieldvalue{0}}%
     {}%
    }%
    \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
  }%
}
```

sXtrIfFieldUndef

`\GlsXtrIfFieldUndef{⟨field⟩}{⟨label⟩}{⟨true⟩}{⟨false⟩}`

Just uses `\ifcsundef`.

```
\newcommand{\GlsXtrIfFieldUndef}[2]{%
 \ifcsundef{glo@\glsdetoklabel{#2}@#1}%
}
```

`\glsxtrusefield` Provide a user-level alternative to `\@gls@entry@field`. The first argument is the entry label. The second argument is the field label.

```
\newcommand*{\glsxtrusefield}[2]{%
  \@gls@entry@field{#1}{#2}%
}
```

`\Glsxtrusefield` Provide a user-level alternative to `\@Gls@entry@field`.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsxtrusefield}[2]{%
    \texorpdfstring
     {\@Gls@entry@field{#1}{#2}}
     {\@gls@entry@field{#1}{#2}}%
  }
}
{
  \newcommand*{\Glsxtrusefield}[2]{%
    \@Gls@entry@field{#1}{#2}%
  }
}
```

`\GLSxtrusefield` As above but convert to all caps.

```
\ifdef\texorpdfstring
{
  \newcommand*{\GLSxtrusefield}[2]{%
    \texorpdfstring
     {\glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}}%
     {\@gls@entry@field{#1}{#2}}%
  }
}
{
  \newcommand*{\GLSxtrusefield}[2]{%
    \glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}%
  }
}
```

`\glsxtrdeffield` Just use `\csdef` to provide a field value for the given entry.

```
\newcommand*{\glsxtrdeffield}[2]{\csdef{glo@\glsdetoklabel{#1}@#2}}
```

`glsxtredeffield` Just use `\csedef` to provide a field value for the given entry.

```
\newcommand*{\glsxtredeffield}[2]{\protected@csedef{glo@\glsdetoklabel{#1}@#2}}
```

`etfieldifexists`

```
\newcommand*{\glsxtrsetfieldifexists}[3]{\glsdoifexists{#1}{#3}}
```

\GlsXtrSetField Allow the user to set a field. First argument entry label, second argument field label, third argument value.

```
\newrobustcmd*{\GlsXtrSetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

\GlsXtrLetField Uses `\cslet` instead. Third argument should be a macro.

```
\newrobustcmd*{\GlstrLetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\cslet{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

sGlsXtrLetField Uses `\csletcs` instead. Third argument should be a control sequence name.

```
\newrobustcmd*{\csGlsXtrLetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csletcs{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

LetFieldToField Sets the field for one entry to the field for another entry. Third argument should be the other entry and the fourth argument that other field label.

```
\newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csletcs{glo@\glsdetoklabel{#1}@#2}{glo@\glsdetoklabel{#3}@#4}}%
}
```

gGlsXtrSetField Allow the user to set a field. First argument entry label, second argument field label, third argument value.

```
\newrobustcmd*{\gGlsXtrSetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csgdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

xGlsXtrSetField

```
\newrobustcmd*{\xGlsXtrSetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\protected@csxdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

eGlsXtrSetField

```
\newrobustcmd*{\eGlsXtrSetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\protected@csedef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

XtrIfFieldEqStr

```
\newrobustcmd*{\GlsXtrIfFieldEqStr}[5]{%
```

```
  \glsxtrifhasfield{#1}{#2}%
  {%
    \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
  }%
  {#5}%
}
```

rIfFieldEqXpStr

Like the above but first expands the string.

```
\newrobustcmd*{\GlsXtrIfFieldEqXpStr}[5]{%
  \glsxtrifhasfield{#1}{#2}%
  {%
    \protected@edef\@gls@tmp{#3}%
    \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
  }%
  {#5}%
}
```

fXpFieldEqXpStr

Like the above but also expands the field value.

```
\newrobustcmd*{\GlsXtrIfXpFieldEqXpStr}[5]{%
  \glsxtrifhasfield{#1}{#2}%
  {%
    \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
    \let\glscurrentfieldvalue\@gls@tmp
    \protected@edef\@gls@tmp{#3}%
    \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
  }%
  {#5}%
}
```

lsXtrForeignText

`\GlsXtrForeignText{⟨entry label⟩}{⟨text⟩}`

If a field is used to store a language tag (such as `en-GB` or `de-CH-1996`) then this command uses tracklang's interface to encapsulate ⟨*text*⟩. The field identifying the locale is given by `\GlsXtrForeignTextField`.

```
\ifdef\foreignlanguage
{
  \ifdef\GetTrackedDialectFromLanguageTag
  {
    \newcommand{\GlsXtrForeignText}[2]{%
```

In case this is used inside the argument of `\glsxtrifhasfield`, save and restore `\glscurrentfieldvalue`.

```
      \let\@glsxtr@org@currentfieldvalue\glscurrentfieldvalue
      \glsxtrifhasfield{\GlsXtrForeignTextField}{#1}%
      {%
        \expandafter\GetTrackedDialectFromLanguageTag\expandafter
          {\glscurrentfieldvalue}{\@glsxtr@dialect}%
        \let\@glsxtr@locale\glscurrentfieldvalue
        \let\glscurrentfieldvalue\@glsxtr@org@currentfieldvalue
        \ifdefempty\@glsxtr@dialect
        {%
```

An exact match hasn't been found. A partial match can only be obtained with at least tracklang v1.3.6.

```
          \ifundef\TrackedDialectClosestSubMatch
          {%
            \GlossariesExtraWarning{Can't obtain dialect label
              (tracklang v1.3.6+ required)}%
          }%
          {\let\@glsxtr@dialect\TrackedDialectClosestSubMatch}%
        }%
        {}%
        \ifdefempty\@glsxtr@dialect
        {%
```

No tracked dialect found for the root language.

```
        }%
        {%
```

Check if there's a caption hook for the given dialect label.

```
          \ifcsundef{captions\@glsxtr@dialect}{}%
          {%
```

Dialect label not recognised. Check if there's a known mapping.

```
            \IfTrackedDialectHasMapping{\@glsxtr@dialect}%
            {%
              \edef\@glsxtr@dialect{%
                \GetTrackedDialectToMapping{\@glsxtr@dialect}}%
```

Does a caption hook exist for this?

```
              \ifcsundef{captions\@glsxtr@dialect}{}%
              {%
```

No mapping. Try root language label instead.

```
                \ifcsundef{captions\@tracklang@lang}{}%
                {%
                  \let\@glsxtr@dialect\@tracklang@lang
                }%
              }%
            }%
            {%
```

No mapping. Try root language label instead.

```
              \ifcsundef{captions\@tracklang@lang}{}%
              {%
                \let\@glsxtr@dialect\@tracklang@lang
              }%
            }%
          }%
```

```
        }%
        \ifdefempty\@glsxtr@dialect
        {%
          \GlsXtrUnknownDialectWarning{\@glsxtr@locale}{\@tracklang@lang}%
          #2%
        }%
        {\foreignlanguage{\@glsxtr@dialect}{#2}}%
      }%
      {#2}% key not set
    }
  }
  {
    \newcommand{\GlsXtrForeignText}[2]{%
      \GlossariesExtraWarning{Can't encapsulate foreign text:
        tracklang v1.3.6+ required}%
      #2%
    }
  }
}
{
```

`\foreignlanguage` isn't defined so just do ⟨*text*⟩.

```
  \newcommand{\GlsXtrForeignText}[2]{#2}
}
```

`oreignTextField` This is the user2 field by default but may be redefined as required.

```
\newcommand*{\GlsXtrForeignTextField}{userii}
```

`nDialectWarning`

```
\newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
  \GlossariesExtraWarning{Can't determine valid dialect label
   for locale '#1' (root language: #2)}%
}
```

`\glsxtrpageref` Like `\glsrefentry` but references the page number instead (if entry counting is on). The base glossaries package only introduced `\GlsEntryCounterLabelPrefix` in version 4.38, so it may not be defined.

```
\ifdef\GlsEntryCounterLabelPrefix
{%
  \newcommand*{\glsxtrpageref}[1]{%
    \ifglsentrycounter
      \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
    \else
      \ifglssubentrycounter
        \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
      \else
        \gls{#1}%
      \fi
    \fi
  }
}%
{%
  \newcommand*{\glsxtrpageref}[1]{%
    \ifglsentrycounter
      \pageref{glsentry-\glsdetoklabel{#1}}%
    \else
      \ifglssubentrycounter
        \pageref{glsentry-\glsdetoklabel{#1}}%
      \else
        \gls{#1}%
      \fi
    \fi
  }
}%
```

lossarypreamble

```
\newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
  \ifcsdef{glolist@#1}%
  {%
   \ifcsundef{@glossarypreamble@#1}%
   {\csdef{@glossarypreamble@#1}{}}%
   {}%
   \csappto{@glossarypreamble@#1}{#2}%
  }%
  {%
    \GlossariesExtraWarning{Glossary `#1' is not defined}%
  }%
}
```

lossarypreamble

```
\newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
  \ifcsdef{glolist@#1}%
  {%
   \ifcsundef{@glossarypreamble@#1}%
   {\csdef{@glossarypreamble@#1}{}}%
   {}%
   \cspreto{@glossarypreamble@#1}{#2}%
  }%
  {%
    \GlossariesExtraWarning{Glossary `#1' is not defined}%
  }%
}
```

1.3 Modifications to Commands Provided by glossaries

Some of the commands provided by glossaries are modified to take into account new options or to change default behaviour.

\ifglsused

`\ifglsused{⟨label⟩}{⟨true part⟩}{⟨false part⟩}`

In the event that undefined entries should trigger a warning rather than an error, `\ifglsused` needs to be modified to check for existence. If the boolean variable is undefined, then its state is indeterminate and is neither true nor false, so neither ⟨*true part*⟩ nor ⟨*false*⟩ part will be performed if ⟨*label*⟩ is undefined.

```
\renewcommand*{\ifglsused}[3]{%
  \glsdoifexists{#1}{\ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}}%
}
```

Provide a starred version of `\longnewglossaryentry` that doesn't automatically insert `\leavevmode\unskip\nopostdesc` at the end of the description. The unstarred version is modified to use `\glsxtrpostlongdescription` instead.

ewglossaryentry

```
\renewcommand*{\longnewglossaryentry}{%
 \@ifstar\@glsxtr@s@longnewglossaryentry\@glsxtr@longnewglossaryentry
}
```

ewglossaryentry

Starred version.

```
\newcommand{\@glsxtr@s@longnewglossaryentry}[3]{%
  \glsdoifnoexists{#1}%
  {%
     \bgroup
       \let\@org@newglossaryentryprehook\@newglossaryentryprehook
       \long\def\@newglossaryentryprehook{%
         \long\def\@glo@desc{#3}%
         \@org@newglossaryentryprehook
       }%
       \renewcommand*{\gls@assign@desc}[1]{%
          \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
          \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
        }
       \gls@defglossaryentry{#1}{#2}%
     \egroup
  }%
}
```

ewglossaryentry

Unstarred version.

```
\newcommand{\@glsxtr@longnewglossaryentry}[3]{%
  \glsdoifnoexists{#1}%
  {%
     \bgroup
       \let\@org@newglossaryentryprehook\@newglossaryentryprehook
       \long\def\@newglossaryentryprehook{%
         \long\def\@glo@desc{#3\glsxtrpostlongdescription}%
         \@org@newglossaryentryprehook
       }%
       \renewcommand*{\gls@assign@desc}[1]{%
          \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
```

The following is different from the base glossaries.sty:

```
          \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
        }
       \gls@defglossaryentry{#1}{#2}%
     \egroup
  }%
}
```

`longdescription` Hook at the end of the description when using the unstarred `\longnewglossaryentry`.

```
\newcommand*{\glsxtrpostlongdescription}{\leavevmode\unskip\nopostdesc}
```

Provide a starred version of `\newignoredglossary` that doesn't add the glossary to the nohyperlist list.

`ignoredglossary` Redefine to check for star.

```
\renewcommand{\newignoredglossary}{%
  \@ifstar\glsxtr@s@newignoredglossary\glsxtr@org@newignoredglossary
}
```

`ignoredglossary` The original definition is patched to check for existence.

```
\newcommand*{\glsxtr@org@newignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {%
    \glsxtrundefaction{Glossary type ‘#1’ already exists}{}%
  }%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \edef\@ignored@glossaries{#1}%
    }%
    {%
      \eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
    \ifdefempty\@gls@nohyperlist
    {%
       \renewcommand*{\@gls@nohyperlist}{#1}%
    }%
    {%
```

```
       \eappto\@gls@nohyperlist{,#1}%
    }%
  }%
}
```

ignoredglossary Starred form.

```
\newcommand*{\glsxtr@s@newignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {%
    \glsxtrundefaction{Glossary type ‘#1’ already exists}{}%
  }%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \edef\@ignored@glossaries{#1}%
    }%
    {%
      \eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
  }%
}
```

\glssettoctitle Ignored glossaries don’t have an associated title, so modify `\glssettoctitle` to check for it to prevent an undefined command written to the toc file.

```
\glsifusetranslator
{%
  \renewcommand*{\glssettoctitle}[1]{%
    \ifcsdef{gls@tr@set@#1@toctitle}%
    {%
      \csuse{gls@tr@set@#1@toctitle}%
    }%
    {%
      \ifcsdef{@glotype@#1@title}%
      {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
      {\def\glossarytoctitle{\glossarytitle}}%
    }%
  }%
}
{
  \renewcommand*{\glssettoctitle}[1]{%
    \ifcsdef{@glotype@#1@title}%
    {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
    {\def\glossarytoctitle{\glossarytitle}}%
  }
}
```

`ignoredglossary` As above but won't do anything if the glossary already exists.

```
\newcommand{\provideignoredglossary}{%
 \@ifstar\glsxtr@s@provideignoredglossary\glsxtr@provideignoredglossary
}
```

`ignoredglossary` Unstarred version.

```
\newcommand*{\glsxtr@provideignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {}%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \edef\@ignored@glossaries{#1}%
    }%
    {%
      \eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
    \ifdefempty\@gls@nohyperlist
    {%
       \renewcommand*{\@gls@nohyperlist}{#1}%
    }%
    {%
       \eappto\@gls@nohyperlist{,#1}%
    }%
  }%
}
```

`ignoredglossary` Starred form.

```
\newcommand*{\glsxtr@s@provideignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {}%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \edef\@ignored@glossaries{#1}%
    }%
    {%
      \eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
  }%
}
```

rcopytoglossary Adds an entry label to another glossary list. First argument is entry label. Second argument is glossary label.

```
\newcommand*{\glsxtrcopytoglossary}[2]{%
  \glsdoifexists{#1}%
  {%
    \ifcsdef{glolist@#2}
    {%
      \cseappto{glolist@#2}{#1,}%
    }%
    {%
      \glsxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
    }%
  }%
}
```

1.3.1 Existence Checks

\glsdoifexists Modify `\glsdoifexists` to take account of the undefaction setting.

```
\renewcommand{\glsdoifexists}[2]{%
  \ifglsentryexists{#1}{#2}%
  {%
```

Define `\glslabel` in case it’s needed after this command (for example in the post-link hook).

```
    \edef\glslabel{\glsdetoklabel{#1}}%
    \glsxtrundefaction{Glossary entry ‘\glslabel’
    has not been defined}{You need to define a glossary entry before
    you can reference it.}%
  }%
}
```

glsdoifnoexists Modify `\glsdoifnoexists` to take account of the undefaction setting.

```
\renewcommand{\glsdoifnoexists}[2]{%
  \ifglsentryexists{#1}{%
    \glsxtrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
    has already been defined}{}}{#2}%
}
```

sdoifexistsordo Modify `\glsdoifexistsordo` to take account of the undefaction setting. This command was introduced in glossaries version 4.19, so check if it has been defined first.

```
\ifdef\glsdoifexistsordo
{%
  \renewcommand{\glsdoifexistsordo}[3]{%
    \ifglsentryexists{#1}{#2}%
    {%
      \glsxtrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
      has not been defined}{You need to define a glossary entry
      before you can use it.}%
      #3%
    }%
  }%
}
{%
  \glsxtr@warnonexistsordo\glsdoifexistsordo
  \newcommand{\glsdoifexistsordo}[3]{%
    \ifglsentryexists{#1}{#2}%
    {%
      \glsxtrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
      has not been defined}{You need to define a glossary entry
      before you can use it.}%
      #3%
    }%
  }%
}
```

`arynoexistsordo` Similarly for `\doifglossarynoexistsordo`.

```
\ifdef\doifglossarynoexistsordo
{%
  \renewcommand{\doifglossarynoexistsordo}[3]{%
    \ifglossaryexists{#1}%
    {%
      \glsxtrundefaction{Glossary type ‘#1’ already exists}{}%
      #3%
    }%
    {#2}%
  }%
}
{%
  \glsxtr@warnonexistsordo\doifglossarynoexistsordo
  \newcommand{\doifglossarynoexistsordo}[3]{%
    \ifglossaryexists{#1}%
    {%
      \glsxtrundefaction{Glossary type ‘#1’ already exists}{}%
      #3%
    }%
    {#2}%
  }%
}

```

There are now three types of cross-references: the see key (as original), the alias key (from glossaries-extra v1.12) and the seealso key (from glossaries-extra v1.16). The original see key needs to have a corresponding field (which it doesn't with the base glossaries package).

`ryentryposthook` Hook into end of `\newglossaryentry` to add "see" value as a field.

```
\appto\@newglossaryentryposthook{%
  \ifdefvoid\@glo@see
   {\csxdef{glo@\@glo@label @see}{}}%
   {%
     \csxdef{glo@\@glo@label @see}{\@glo@see}%
     \if@glsxtr@autoseeindex
       \@glsxtr@autoindexcrossrefs
     \fi
   }%
}
\appto\@gls@keymap{,{see}{see}}
```

`\glsxtrusesee` Apply `\glsseeformat` to the see key if not empty.

```
\newcommand*{\glsxtrusesee}[1]{%
  \glsdoifexists{#1}%
  {%
    \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
    \ifdefempty\@glo@see
    {}%
    {%
      \expandafter\glsxtr@usesee\@glo@see\@end@glsxtr@usesee
    }%
  }%
}
```

`\glsxtr@usesee`

```
\newcommand*{\glsxtr@usesee}[1][\seename]{%
  \@glsxtr@usesee[#1]%
}
```

`\@glsxtr@usesee`

```
\def\@glsxtr@usesee[#1]#2\@end@glsxtr@usesee{%
  \glsxtruseseeformat{#1}{#2}%
}
```

`xtruseseeformat` The format used by `\glsxtrusesee`. The first argument is the tag (such as `\seename`). The second argument is the comma-separated list of cross-referenced labels.

```
\newcommand*{\glsxtruseseeformat}[2]{%
  \glsseeformat[#1]{#2}{}%
}
```

`lsseeitemformat` glossaries originally defined `\glsseeitemformat` to use `\glsentryname` but in v3.0 this was switched to use `\glsentrytext` due to problems occurring with the name field being sanitized. Since this is no longer a problem, glossaries-extra restores the original definition as it makes more sense to use the name in the cross-reference list. This still uses `\glsaccesstext` for abbreviations.

```
\renewcommand*{\glsseeitemformat}[1]{%
 \ifglshasshort{#1}{\glsaccesstext{#1}}{\glsaccessname{#1}}%
}
```

`\glsxtrhiername`

```
\glsxtrhiername{⟨label⟩}
```

Displays the hierarchical name for the given entry. The cross-reference format `\glsseeitemformat` may be redefined to use this command to show the hierarchy, if required.

```
\newcommand*{\glsxtrhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {\glsxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
    {}%
    \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}%
  }%
}
```

`\Glsxtrhiername`

```
\Glsxtrhiername{⟨label⟩}
```

As above but displays the top-level name with an initial capital.

```
\newcommand*{\Glsxtrhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {%
      \Glsxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep
      \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}%
    }%
    {\ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}}%
  }%
}
```

`\GlsXtrhiername`

```
\GlsXtrhiername{⟨label⟩}
```

As above but converts the first letter of each name to a capital.

```
\newcommand*{\GlsXtrhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {\GlsXtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
    {}%
    \ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}%
  }%
}
```

`\GLSxtrhiername`

`\GLSxtrhiername{⟨label⟩}`

As above but displays the top-level name in all-caps.

```
\newcommand*{\GLSxtrhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {%
      \GLSxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep
      \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}%
    }%
    {\ifglshasshort{#1}{\GLSaccessshort{#1}}{\GLSaccessname{#1}}}%
  }%
}
```

`\GLSXTRhiername`

`\GLSXTRhiername{⟨label⟩}`

As above but displays all names in all-caps.

```
\newcommand*{\GLSXTRhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {\GLSXTRhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
    {}
    \ifglshasshort{#1}{\GLSaccessshort{#1}}{\GLSaccessname{#1}}%
  }%
}
```

`sxtrhiernamesep` Separator used in `\glsxtrhiername` and variants.

```
\newcommand*{\glsxtrhiernamesep}{\,{\small$\triangleright$}\,}
```

`lsxtruseseealso` Apply `\glsseeformat` to the seealso key if not empty. There’s no optional tag to worry about here.

```
\newcommand*{\glsxtruseseealso}[1]{%
  \glsdoifexists{#1}%
```

```
 {%
   \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
   \ifdefempty\@glo@see
   {}%
   {%
     \expandafter\glsxtruseseealsoformat\expandafter{\@glo@see}%
   }%
 }%
}
```

`seseealsoformat` The format used by `\glsxtruseseealso`. The argument is the comma-separated list of cross-referenced labels.

```
\newcommand*{\glsxtruseseealsoformat}[1]{%
 \glsseeformat[\seealsoname]{#1}{}%
}
```

`\glsxtrseelist` Fully expands argument before passing to `\glsseelist`. (The argument to `\glsseelist` must be a comma-separated list of entry labels.)

```
\newrobustcmd{\glsxtrseelist}[1]{%
 \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
}
```

`\seealsoname` In case this command hasn't been defined. (Should be provided by language packages.)

```
\providecommand{\seealsoname}{see also}
```

`xtrindexseealso` If `\@xdycrossrefhook` is defined, provide a `seealso` crossref class. Otherwise this just does `\glssee` with `\seealsoname` as the tag. The hook is only defined if both xindy and glossaries v4.30+ are being used.

```
\ifdef\@xdycrossrefhook
{
```

Add the cross-reference class definition to the hook.

```
 \appto\@xdycrossrefhook{%
   \write\glswrite{(define-crossref-class \string"seealso\string"
     :unverified )}%
   \write\glswrite{(markup-crossref-list
      :class \string"seealso\string"^^J\space\space\space
      :open \string"\string\glsxtruseseealsoformat\glsopenbrace\string"
      :close \string"\glsclosebrace\string")}%
 }
```

Append to class list.

```
 \appto\@xdylocationclassorder{\space\string"seealso\string"}
```

This essentially works like `\@do@seeglossary` but uses the `seealso` class. This doesn't increment the associated counter.

```
 \newrobustcmd*{\glsxtrindexseealso}[2]{%
   \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
     \@glsxtr@recordsee{#1}{#2}%
```

```
    \fi
    \glsdoifexists{#1}%
    {%
      \@@glsxtrwrglossmark
      \def\@gls@xref{#2}%
      \@onelevel@sanitize\@gls@xref
      \@gls@checkmkidxchars\@gls@xref
      \gls@glossary{\csname glo@#1@type\endcsname}{%
        (indexentry
          :tkey (\csname glo@#1@index\endcsname)
          :xref (\string"\@gls@xref\string")
          :attr \string"seealso\string"
        )
      }%
    }%
  }
}
{
```

xindy not in use or glossaries version too old to support this.

```
  \newrobustcmd*{\glsxtrindexseealso}{\glssee[\seealsoname]}
}
```

The alias key should be set to the label of the synonymous entry. The seealso key essentially behaves like `see=[\seealsoname]{⟨xr-list⟩}`. Neither of these new keys has the optional tag part allowed with see.

If `\gls@set@xr@key` has been defined (glossaries v4.30), use that, otherwise just use `\glsaddstoragekey`.

```
\ifdef\gls@set@xr@key
{
```

We have at least glossaries v4.30. This means the new keys can be governed by the same settings as the see key.

```
  \define@key{glossentry}{alias}{%
    \gls@set@xr@key{alias}{\@glo@alias}{#1}%
  }
  \define@key{glossentry}{seealso}{%
    \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
  }
```

Add to the key mappings.

```
  \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}
```

Set the default value.

```
  \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%
```

Assign the field values.

```
  \appto\@newglossaryentryposthook{%
    \ifdefvoid\@glo@seealso
     {\csxdef{glo@\@glo@label @seealso}{}}%
     {%
       \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
       \if@glsxtr@autoseeindex
         \@glsxtr@autoindexcrossrefs
       \fi
     }%
```

The alias field doesn't trigger the automatic cross-reference indexing performed at the end of the document.

```
    \ifdefvoid\@glo@alias
     {\csxdef{glo@\@glo@label @alias}{}}%
     {%
       \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
     }%
  }
```

Provide user-level commands to access the values.

\glsxtralias

```
  \newcommand*{\glsxtralias}[1]{\@gls@entry@field{#1}{alias}}
```

trseealsolabels

```
  \newcommand*{\glsxtrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}
```

Add to the `\@glo@autosee` hook.

```
  \appto\@glo@autoseehook{%
    \ifdefvoid\@glo@alias
    {%
      \ifdefvoid\@glo@seealso
      {}%
      {%
        \edef\@do@glssee{\noexpand\glsxtrindexseealso
          {\@glo@label}{\@glo@seealso}}%
        \@do@glssee
      }%
    }%
    {%
```

Add cross-reference if see key hasn't been used.

```
      \ifdefvoid\@glo@see
      {%
        \edef\@do@glssee{\noexpand\glssee{\@glo@label}{\@glo@alias}}%
        \@do@glssee
      }%
      {}%
    }%
  }%
}
{
```

We have an older version of glossaries, so just use `\glsaddstoragekey`.

\glsxtralias

```
\glsaddstoragekey*{alias}{}{\glsxtralias}
```

trseealsolabels

```
\glsaddstoragekey*{seealso}{}{\glsxtrseealsolabels}
```

If `\gls@set@xr@key` isn't defined, then `\@glo@autosee` won't be either, so use the post entry definition hook.

ryentryposthook

Append to the hook to check for the alias and seealso keys.

```
\appto\@newglossaryentryposthook{%
  \ifcsvoid{glo@\@glo@label @alias}%
  {%
    \ifcsvoid{glo@\@glo@label @seealso}%
    {}%
    {%
      \edef\@do@glssee{\noexpand\glsxtrindexseealso
        {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
      \@do@glssee
    }%
  }%
  {%
```

Add cross-reference if see key hasn't been used.

```
    \ifdefvoid\@glo@see
    {%
      \edef\@do@glssee{\noexpand\glssee
        {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
      \@do@glssee
    }%
    {}%
  }%
}

}
```

Add all unused cross-references at the end of the document.

```
\AtEndDocument{\if@glsxtrindexcrossrefs\glsxtraddallcrossrefs\fi}
```

addallcrossrefs

Iterate through all used entries and if they have a cross-reference, make sure the cross-reference has been added.

```
\newcommand*{\glsxtraddallcrossrefs}{%
  \forallglossaries{\@glo@type}%
  {%
    \forglsentries[\@glo@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {\expandafter\@glsxtr@addunusedxrefs\expandafter{\@glo@label}}{}%
    }%
  }%
}
```

`@addunusedxrefs` If the given entry has a see or seealso field add all unused cross-references. (The alias field isn't checked.)

```
\newcommand*{\@glsxtr@addunusedxrefs}[1]{%
  \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
  \ifdefvoid\@glo@see
  {}%
  {%
    \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
  }%
  \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
  \ifdefvoid\@glo@see
  {}%
  {%
    \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
  }%
}
```

`lsxtr@addunused` Adds all the entries if they haven't been used.

```
\newcommand*{\glsxtr@addunused}[1][]{%
  \@glsxtr@addunused
}
```

`lsxtr@addunused` Adds all the entries if they haven't been used.

```
\def\@glsxtr@addunused#1\@end@glsxtr@addunused{%
 \@for\@glsxtr@label:=#1\do
 {%
   \ifglsused{\@glsxtr@label}{}%
   {%
     \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
     \glsunset{\@glsxtr@label}%
     \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
   }%
 }%
}
```

`xtrunusedformat`

```
\newcommand*{\glsxtrunusedformat}[1]{\unskip}
```

1.3.2 Document Definitions

`ls@begindocdefs` This command was only introduced to glossaries v4.37, so it may not be defined. If it has been defined, redefine it to check `\@glsxtr@docdefval` so that it only inputs the `.glsdefs` file if docdef=true.

```
\ifdef\gls@begindocdefs
{%
```

```
\renewcommand*{\gls@begindocdefs}{%
  \ifnum\@glsxtr@docdefval=1\relax
    \@gls@enablesavenonumberlist
    \edef\@gls@restoreat{%
      \noexpand\catcode`\noexpand\@=\number\catcode`\@\relax}%
    \makeatletter
    \InputIfFileExists{\jobname.glsdefs}{}{}%
    \@gls@restoreat
    \undef\@gls@restoreat
    \gls@defdocnewglossaryentry
  \else
    \ifnum\@glsxtr@docdefval=3\relax
```

The docdef=atom package option has been set. Create the .glsdefs file for the autocomplete support but don't read it.

```
      \@gls@enablesavenonumberlist
      \let\gls@checkseeallowed\relax
      \let\newglossaryentry\new@atom@glossaryentry
      \global\newwrite\@gls@deffile
      \immediate\openout\@gls@deffile=\jobname.glsdefs
```

Write all currently defined entries.

```
      \forallglsentries{\@glsentry}{\@gls@writedef{\@glsentry}}%
    \fi
  \fi
 }
}
{%
  \ifnum\@glsxtr@docdefval=3\relax
    \PackageError{glossaries-extra}{Package option
    `docdef=\@glsxtr@docdefsetting' requires at least version 4.37
    of the base glossaries.sty package}{}
  \fi
}
```

m@glossaryentry

```
\newrobustcmd{\new@atom@glossaryentry}[2]{%
  \gls@defglossaryentry{#1}{#2}%
  \@gls@writedef{#1}%
}
```

noidxglossaries Modify `\makenoidxglossaries` so that it automatically sets docdef=false (unless the restricted setting is on) and disables the docdef key. This command isn't allowed with the record option.

```
\let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
\renewcommand{\makenoidxglossaries}{%
 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
 {%
   \glsxtr@orgmakenoidxglossaries
```

Add marker to `\@do@seeglossary` but don't increment associated counter.

```
    \renewcommand{\@do@seeglossary}[2]{%
      \@@glsxtrwrglossmark
      \edef\@gls@label{\glsdetoklabel{##1}}%
      \protected@write\@auxout{}{%
        \string\@gls@reference
          {\csname glo@\@gls@label @type\endcsname}%
          {\@gls@label}%
          {%
            \string\glsseeformat##2{}%
          }%
      }%
    }%
```

Check for docdefs=restricted:

```
    \if@glsxtrdocdefrestricted
```

If restricted document definitions allowed, adjust `\@gls@reference` so that it doesn't test for existence.

```
      \renewcommand*{\@gls@reference}[3]{%
        \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
        \ifinlistcs{##2}{@glsref@##1}%
        {}%
        {\listcsgadd{@glsref@##1}{##2}}%
        \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
        {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
        {}%
        \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
      }%
    \else
```

Disable document definitions.

```
      \@glsxtrdocdeffalse
    \fi
    \disable@keys{glossaries-extra.sty}{docdef}%
  }%
  {%
    \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
     not permitted\MessageBreak
     with record=\@glsxtr@record@setting\space package option}%
    {You may only use \string\makenoidxglossaries\ space with the
     record=off option}%
  }%
}
```

`ewglossaryentry` Modify `\gls@defdocnewglossaryentry` so that it checks the docdef value.

```
\renewcommand*{\gls@defdocnewglossaryentry}{%
  \ifcase\@glsxtr@docdefval
```

docdef=false:

```
    \renewcommand*{\newglossaryentry}[2]{%
      \PackageError{glossaries-extra}{Glossary entries must
       be \MessageBreak defined in the preamble with \MessageBreak
       package option ‘docdef=false’\MessageBreak(consider using
       ‘docdef=restricted’)}{Move your glossary definitions to
       the preamble. You can also put them in a \MessageBreak separate file
       and load them with \string\loadglsentries.}%
    }%
  \or
```

(docdef=true case.) Since the see value is now saved in a field, it can be used by entries that have been defined in the document.

```
    \let\gls@checkseeallowed\relax
    \let\newglossaryentry\new@glossaryentry
  \else
```

Restricted mode just needs to allow the see value.

```
    \let\gls@checkseeallowed\relax
  \fi
}%
```

Permit a special form of document definition, but only allow it if the glossaries come at the end of the document. These commands behave a little like a combination of `\newterm` and `\gls`. This must be explicitly enabled with the following.

`rEnableOnTheFly`

```
\newcommand*{\GlsXtrEnableOnTheFly}{%
  \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
}
```

`rEnableOnTheFly` The starred version attempts to allow UTF8 characters in the label, but this may break! (Formatting commands mustn’t be used in the label, but the label may be a command whose replacement text is the actual label. This doesn’t take into account a command that’s defined in terms of another command that may eventually expand to the label text.)

```
\newcommand*{\@sGlsXtrEnableOnTheFly}{%
  \renewcommand*{\glsdetoklabel}[1]{%
    \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
    {%
      \expandafter\detokenize\expandafter{##1}%
    }%
    {\detokenize{##1}}%
  }%
  \@GlsXtrEnableOnTheFly
}
\def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
  \expandafter\if\glsbackslash#1%
    #3%
  \else
    #4%
  \fi
}
```

sxtrstarflywarn

```
\newcommand*{\glsxtrstarflywarn}{%
  \GlossariesExtraWarning{Experimental starred version of
  \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
  read the warnings in the glossaries-extra user manual)}%
}
```

rEnableOnTheFly

```
\newcommand*{\@GlsXtrEnableOnTheFly}{%
```

Don't redefine `\glsdetoklabel` if LuaTeX or XeTeX is being used, since it's mainly to allow accented characters in the label.

These definitions are all assigned the category given by:

\glsxtrcat

```
  \newcommand*{\glsxtrcat}{general}
```

\glsxtr

```
  \newcommand*{\glsxtr}[1][]{%
   \def\glsxtr@keylist{##1}%
   \@glsxtr
  }
```

\@glsxtr

```
  \newcommand*{\@glsxtr}[2][]{%
   \ifglsentryexists{##2}%
   {%
     \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
   }%
   {%
     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
       description={\nopostdesc},##1}%
   }%
   \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
  }
```

\Glsxtr

```
  \newcommand*{\Glsxtr}[1][]{%
   \def\glsxtr@keylist{##1}%
   \@Glsxtr
  }
```

\@Glsxtr

```
  \newcommand*{\@Glsxtr}[2][]{%
   \ifglsentryexists{##2}%
```

```
 {%
   \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
 }%
 {%
   \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
     description={\nopostdesc},##1}%
 }%
 \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
}
```

\glsxtrpl

```
\newcommand*{\glsxtrpl}[1][]{%
 \def\glsxtr@keylist{##1}%
 \@glsxtrpl
}
```

\@glsxtrpl

```
\newcommand*{\@glsxtrpl}[2][]{%
 \ifglsentryexists{##2}%
 {%
   \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
 }%
 {%
   \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
     description={\nopostdesc},##1}%
 }%
 \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
}
```

\Glsxtrpl

```
\newcommand*{\Glsxtrpl}[1][]{%
 \def\glsxtr@keylist{##1}%
 \@Glsxtrpl
}
```

\@Glsxtrpl

```
\newcommand*{\@Glsxtrpl}[2][]{%
 \ifglsentryexists{##2}
 {%
   \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
 }%
 {%
   \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
     description={\nopostdesc},##1}%
 }%
 \expandafter\Glspl\expandafter[\glsxtr@keylist]{##2}%
}
```

\GlsXtrWarning

```
  \newcommand*{\GlsXtrWarning}[2]{%
    \def\@glsxtr@optlist{##1}%
    \@onelevel@sanitize\@glsxtr@optlist
    \GlossariesExtraWarning{The options ‘\@glsxtr@optlist’ have
    been ignored for entry ‘##2’ as it has already been defined}%
  }
```

Disable commands after the glossary:

```
  \renewcommand\@printglossary[2]{%
    \def\@glsxtr@printglossopts{##1}%
    \@glsxtr@orgprintglossary{##1}{##2}%
    \def\@glsxtr{\@glsxtr@disabledflycommand\glsxtr}%
    \def\@glsxtrpl{\@glsxtr@disabledflycommand\glsxtrpl}%
    \def\@Glsxtr{\@glsxtr@disabledflycommand\Glsxtr}%
    \def\@Glsxtrpl{\@glsxtr@disabledflycommand\Glsxtrpl}%
  }
```

`abledflycommand`

```
  \newcommand*{\@glsxtr@disabledflycommand}[1]{%
    \PackageError{glossaries-extra}%
    {\string##1\space can’t be used after any of the \MessageBreak
     glossaries have been displayed}%
    {The on-the-fly commands enabled by
     \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
     before the glossaries. If you want to use any entries \MessageBreak
     after any of the glossaries, you must use the standard \MessageBreak
     method of first defining the entry and then using the \MessageBreak
     entry with commands like \string\gls}%
     \@@glsxtr@disabledflycommand
  }%
  \newcommand*{\@@glsxtr@disabledflycommand}[2][]{##2}
```

End of `\GlsXtrEnableOnTheFly`. Disable since it can only be used once.

```
  \let\GlsXtrEnableOnTheFly\relax
}
\@onlypreamble\GlsXtrEnableOnTheFly
```

1.3.3 Existing Glossary Style Modifications

Modify `\setglossarystyle` to keep track of the current style. This allows the `\glossaries-extra-stylemods` package to reset the current style after the required modifications have been made.

`r@current@style` Initialise the current style to the default style.

```
\newcommand*{\@glsxtr@current@style}{\@glossary@default@style}
```

Modify `\setglossarystyle` to set `\@glsxtr@current@style`.

`etglossarystyle`

```
\renewcommand*{\setglossarystyle}[1]{%
```

```
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
  }%
  {%
    \csname @glsstyle@#1\endcsname
```

Only set the current style if it exists.

```
    \protected@edef\@glsxtr@current@style{#1}%
  }%
  \ifx\@glossary@default@style\relax
    \protected@edef\@glossary@default@style{#1}%
  \fi
}
```

In case we have an old version of glossaries:

```
\ifdef\@glossary@default@style
{}
{%
  \let\@glossary@default@style\relax
}
```

listdottedwidth

If `\glslistdottedwidth` has been defined and is currently equal to `.5\hsize` then make the modification suggested in bug report #92

```
\ifdef\glslistdottedwidth
{%
  \ifdim\glslistdottedwidth=.5\hsize
    \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
    \AtBeginDocument{%
      \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
       \setlength{\glslistdottedwidth}{.5\columnwidth}%
      \fi
    }%
  \fi
}
{}%
```

Similarly for `\glsdescwidth`:

`\glsdescwidth`

```
\ifdef\glsdescwidth
{%
  \ifdim\glsdescwidth=.6\hsize
    \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
    \AtBeginDocument{%
      \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
       \setlength{\glsdescwidth}{.6\columnwidth}%
      \fi
    }%
  \fi
}
{}%
```

and for `\glspagelistwidth`:

`lspagelistwidth`

```
\ifdef\glspagelistwidth
{%
  \ifdim\glspagelistwidth=.1\hsize
    \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
    \AtBeginDocument{%
      \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
       \setlength{\glspagelistwidth}{.1\columnwidth}%
      \fi
    }%
  \fi
}
{}%
```

`aryentrynumbers` Has the nonumberlist option been used?

```
\def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
\ifx\org@glossaryentrynumbers\glossaryentrynumbers
  \glsnonumberlistfalse
  \renewcommand*{\glossaryentrynumbers}[1]{%
    \ifglsentryexists{\glscurrententrylabel}%
    {%
      \@glsxtrpreloctag
      \GlsXtrFormatLocationList{#1}%
      \@glsxtrpostloctag
      \gls@save@numberlist{#1}%
    }{}%
  }%
\else
  \glsnonumberlisttrue
  \renewcommand*{\glossaryentrynumbers}[1]{%
    \ifglsentryexists{\glscurrententrylabel}%
    {%
      \gls@save@numberlist{#1}%
    }{}%
  }%
\fi
```

`matLocationList` Provide an easy interface to change the format of the location list without removing the save number list stuff.

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

Sometimes users want to prefix the location list with “page”/“pages”. The simplest way to determine if the location list consists of a single location is to check for instances of `\delimN` or `\delimR`, but this isn't so easy to do as they might be embedded inside the argument of formatting commands. With a bit of trickery we can find out by adjusting `\delimN` and `\delimR` to set a flag and then save information to the auxiliary file for the next run.

ePreLocationTag

```
\newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
  \let\@glsxtrpreloctag\@@glsxtrpreloctag
  \let\@glsxtrpostloctag\@@glsxtrpostloctag
  \renewcommand*{\@glsxtr@pagetag}{#1}%
  \renewcommand*{\@glsxtr@pagestag}{#2}%
  \renewcommand*{\@glsxtr@savepreloctag}[2]{%
    \csgdef{@glsxtr@preloctag@##1}{##2}%
  }%
  \renewcommand*{\@glsxtr@doloctag}{%
    \ifcsundef{@glsxtr@preloctag@\glscurrententrylabel}%
    {%
      \GlossariesWarning{Missing pre-location tag for `\glscurrententrylabel'.
        Rerun required}%
    }%
    {%
      \csuse{@glsxtr@preloctag@\glscurrententrylabel}%
    }%
  }%
}
\@onlypreamble\GlsXtrEnablePreLocationTag
```

glsxtrpreloctag

```
\newcommand*{\@@glsxtrpreloctag}{%
   \let\@glsxtr@org@delimN\delimN
   \let\@glsxtr@org@delimR\delimR
   \let\@glsxtr@org@glsignore\glsignore
```

`\gdef` is required as the delimiters may occur inside a scope.

```
   \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
   \renewcommand*{\delimN}{%
     \gdef\@glsxtr@thisloctag{\@glsxtr@pagestag}%
     \@glsxtr@org@delimN}%
   \renewcommand*{\delimR}{%
     \gdef\@glsxtr@thisloctag{\@glsxtr@pagestag}%
     \@glsxtr@org@delimR}%
   \renewcommand*{\glsignore}[1]{%
     \gdef\@glsxtr@thisloctag{\relax}%
     \@glsxtr@org@glsignore{##1}}%
   \@glsxtr@doloctag
}
```

glsxtrpreloctag

```
\newcommand*{\@glsxtrpreloctag}{}
```

@glsxtr@pagetag

```
\newcommand*{\@glsxtr@pagetag}{}%
```

glsxtr@pagestag

```
\newcommand*{\@glsxtr@pagestag}{}%
```

lsxtrpostloctag

```
\newcommand*{\@@glsxtrpostloctag}{%
   \let\delimN\@glsxtr@org@delimN
   \let\delimR\@glsxtr@org@delimR
   \let\glsignore\@glsxtr@org@glsignore
   \protected@write\@auxout{}%
    {\string\@glsxtr@savepreloctag{\glscurrententrylabel}{\@glsxtr@thisloctag}}%
}
```

lsxtrpostloctag

```
\newcommand*{\@glsxtrpostloctag}{}
```

lsxtr@preloctag

```
\newcommand*{\@glsxtr@savepreloctag}[2]{}
\protected@write\@auxout{}{%
  \string\providecommand\string\@glsxtr@savepreloctag[2]{}}
```

glsxtr@doloctag

```
\newcommand*{\@glsxtr@doloctag}{}
```

ss@nonumberlist

Modify the nonumberlist key to use `\GlsXtrFormatLocationList` (and also save the number list):

```
\renewcommand*{\KV@printgloss@nonumberlist}[1]{%
 \XKV@plfalse
 \XKV@sttrue
 \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
 {%
   \csname glsnonumberlist\XKV@resa\endcsname
   \ifglsnonumberlist
     \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
   \else
     \def\glossaryentrynumbers##1{%
       \@glsxtrpreloctag
       \GlsXtrFormatLocationList{##1}%
       \@glsxtrpostloctag
       \gls@save@numberlist{##1}}%
   \fi
 }%
}
```

1.3.4 Entry Formatting, Hyperlinks and Indexing

`\glsentryfmt`

Change default entry format. Use the generic format for regular terms (that is, entries that have a category with the regular attribute set) or non-regular terms without a short value and use the abbreviation format for non-regular terms that have a short value. If further attributes need to be checked, then `\glsentryfmt` will need redefining as appropriate (or use `\defglsentryfmt`). The abbreviation format is set here for entries that have a short form, even if they are regular entries to ensure the abbreviation fonts are correct.

```
\renewcommand*{\glsentryfmt}{%
  \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}{}%
  \glsifregular{\glslabel}%
  {\glsxtrregularfont{\glsgenentryfmt}}%
  {%
    \ifglshasshort{\glslabel}%
    {\glsxtrabbreviationfont{\glsxtrgenabbrvfmt}}%
    {\glsxtrregularfont{\glsgenentryfmt}}%
  }%
}
```

`sxtrregularfont`

Font used for regular entries.

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

`bbreviationfont`

Font used for abbreviation entries.

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

Commands like `\glsifplural` are only used by the `\gls`-like commands in the glossaries package, but it might be useful for the postlink hook to know if the user has used, say, `\glsfirst` or `\glsplural`. This can provide better consistency with the formatting of the `\gls`-like commands, even though they don't use `\glsentryfmt`.

`@gls@field@link`

Redefine `\@gls@field@link` so that commands like `\glsfirst` can setup `\glsxtrifwasfirstuse` etc to allow the postlink hook to work better. This now has an optional argument that sets up the defaults.

```
\renewcommand{\@gls@field@link}[4][]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#2}{#3}{glslink}%
  \glsdoifexists{#3}%
  {%
```

Save and restore the hyper setting (`\@gls@link` also does this, but that's too late if the optional argument of `\@gls@field@link` modifies it).

```
    \let\glsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glscustomtext{#4}%
    \@glsxtr@field@linkdefs
    #1%
    \@gls@link[#2]{#3}{#4}%
    \let\ifKV@glslink@hyper\glsxtrorg@ifKV@glslink@hyper
  }%
  \glspostlinkhook
}
```

The commands `\gls`, `\Gls` etc don't use `\@gls@field@link`, so they need modifying as well to use `\@glsxtr@record`.

`\@gls@` Save the original definition and redefine.

```
\let\@glsxtr@org@gls@\@gls@
\def\@gls@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@gls@{#1}{#2}%
}%
```

`\@glspl@` Save the original definition and redefine.

```
\let\@glsxtr@org@glspl@\@glspl@
\def\@glspl@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@glspl@{#1}{#2}%
}%
```

`\@Gls@` Save the original definition and redefine.

```
\let\@glsxtr@org@Gls@\@Gls@
\def\@Gls@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@Gls@{#1}{#2}%
}%
```

`\@Glspl@` Save the original definition and redefine.

```
\let\@glsxtr@org@Glspl@\@Glspl@
\def\@Glspl@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@Glspl@{#1}{#2}%
}%
```

`\@GLS@` Save the original definition and redefine.

```
\let\@glsxtr@org@GLS@\@GLS@
\def\@GLS@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@GLS@{#1}{#2}%
}%
```

`\@GLSpl@` Save the original definition and redefine.

```
\let\@glsxtr@org@GLSpl@\@GLSpl@
\def\@GLSpl@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@GLSpl@{#1}{#2}%
}%
```

`\@glsdisp` This is redefined to allow the recording on the first run. Can't save and restore `\@glsdisp` since it has an optional argument.

```
\renewcommand*{\@glsdisp}[3][]{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}{%
    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \def\glscustomtext{#3}%
    \def\glsinsert{}%
    \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
    \@gls@link[#1]{#2}{\@glo@text}%
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
  \glspostlinkhook
}
```

`\@gls@@link@` Redefine to include `\@glsxtr@record`

```
\renewcommand*{\@gls@@link}[3][]{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexistsordo{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax
```

Post-link hook commands need initialising.

```
    \def\glscustomtext{#3}%
    \@glsxtr@field@linkdefs
    \@gls@link[#1]{#2}{#3}%
  }%
  {%
    \glstextformat{#3}%
  }%
  \glspostlinkhook
}
```

`sxtrinitwrgloss` Set the default if the wrgloss is omitted.

```
\newcommand*{\glsxtrinitwrgloss}{%
 \glsifattribute{\glslabel}{wrgloss}{after}%
 {%
   \glsxtrinitwrglossbeforefalse
 }%
 {%
   \glsxtrinitwrglossbeforetrue
 }%
}
```

trwrglossbefore Conditional to determine if the indexing should be done before the link text.

```
\newif\ifglsxtrinitwrglossbefore
\glsxtrinitwrglossbeforetrue
```

Define a wrgloss key to determine whether to write the glossary information before or after the link text.

```
\define@choicekey{glslink}{wrgloss}%
[\@glsxtr@wrglossval\@glsxtr@wrglossnr]%
{before,after}%
{%
  \ifcase\@glsxtr@wrglossnr\relax
    \glsxtrinitwrglossbeforetrue
  \or
    \glsxtrinitwrglossbeforefalse
  \fi
}
```

```
\define@key{glslink}{thevalue}{\def\@glsxtr@thevalue{#1}}
```

```
\define@key{glslink}{theHvalue}{\def\@glsxtr@theHvalue{#1}}
```

tr@hyperoutside Define a hyperoutside key to determine whether `\hyperlink` should be outside `\glstextformat`.

```
\define@boolkey{glslink}[glsxtr@]{hyperoutside}[true]{}
\glsxtr@hyperoutsidetrue
```

ocal@textformat Provide a key to locally change the text format.

```
\define@key{glslink}{textformat}{%
  \ifcsdef{#1}
  {%
    \letcs{\@glsxtr@local@textformat}{#1}%
  }%
  {%
    \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
  }%
}
```

```
\define@key{glslink}{prefix}{\def\glolinkprefix{#1}}
```

nithyperoutside Set the default if the hyperoutside is omitted.

```
\newcommand*{\glsxtrinithyperoutside}{%
 \glsifattribute{\glslabel}{hyperoutside}{false}%
 {%
   \glsxtr@hyperoutsidefalse
 }%
 {%
   \glsxtr@hyperoutsidetrue
 }%
}
```

`r@inc@linkcount` Does nothing by default.

```
\newcommand*{\glsxtr@inc@linkcount}{}
```

`slinkpresetkeys` User hook performed immediately before options are set. Does nothing by default.

```
\newcommand*{\glslinkpresetkeys}{}
```

`sXtrExpandedFmt` Helper command that (protected) fully expands second argument and then applies it to the first, which must be a command that takes a single argument.

```
\newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
  \protected@edef\@glsxtr@tmp{#2}%
  \expandafter#1\expandafter{\@glsxtr@tmp}%
}
```

`tion@counter@or` If in a numbered equation, change the counter to equation. This can be overridden by explicitly setting the counter in the optional argument of commands like `\gls` and `\glslink`.

```
\newcommand*{\@glsxtr@use@equation@counter}{%
  \@glsxtr@ifnum@mmode{\def\@gls@counter{equation}}{}%
}
```

`sxtr@do@autoadd` If `\GlsXtrAutoAddOnFormat` is used, this will automatically use `\glsadd`. It's therefore only used with `\@gls@link` not with `\glsadd` otherwise it could trigger an infinite loop. The argument indicates the key family (glslink or glossadd).

```
\newcommand*{\glsxtr@do@autoadd}[1]{}
```

`rAutoAddOnFormat` `\GlsXtrAutoAddOnFormat[⟨label⟩]{⟨format list⟩}{⟨glsadd options⟩}`

If an entry is indexed with the format set to one identified in the comma-separated list, then automatically index it using `\glsadd` with the given options, which may override the current options. Scoping is needed to prevent leakage.

```
\newcommand*{\GlsXtrAutoAddOnFormat}[3][\glslabel]{%
  \renewcommand*{\glsxtr@do@autoadd}[1]{%
    \begingroup
      \protected@edef\@glsxtr@do@autoadd{%
        \noexpand\ifstrequal{##1}{glslink}%
        {%
          \noexpand\DTLifinlist{\@glsnumberformat}{#2}{\noexpand\glsadd[format={\@glsnumberfor
        }%
        {}%
      }%
      \@glsxtr@do@autoadd
    \endgroup
  }%
}
```

`\@gls@link` Redefine to allow the indexing to be placed after the link text. By default this is done before the link text to prevent problems that can occur from the whatsit, but there may be times when the user would like the indexing done afterwards even though it causes a whatsit.

```
\def\@gls@link[#1]#2#3{%
  \leavevmode
  \edef\glslabel{\glsdetoklabel{#2}}%
  \def\@gls@link@opts{#1}%
  \let\@gls@link@label\glslabel
  \let\@glsnumberformat\@glsxtr@defaultnumberformat
  \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
  \edef\glstype{\csname glo@\glslabel @type\endcsname}%
  \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Save current value of `\glolinkprefix`:

```
  \let\@glsxtr@org@glolinkprefix\glolinkprefix
```

Initialise `\@glsxtr@local@textformat`

```
  \let\@glsxtr@local@textformat\relax
```

Initialise thevalue and theHvalue (v1.19).

```
  \def\@glsxtr@thevalue{}%
  \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

Initialise when indexing should occur (new to v1.14).

```
  \glsxtrinitwrgloss
```

Initialise whether `\hyperlink` should be outside `\glstextformat` (new to v1.21).

```
  \glsxtrinithyperoutside
```

Note that the default link options may override `\glsxtrinitwrgloss`.

```
  \@gls@setdefault@glslink@opts
```

Increment link counter if enabled (new to v1.26).

```
   \glsxtr@inc@linkcount
```

Check if the equations option has been set (new to v1.37).

```
   \if@glsxtr@equations
     \@glsxtr@use@equation@counter
   \fi
```

As the original definition.

```
  \do@glsdisablehyperinlist
  \do@gls@link@checkfirsthyper
```

User hook before options are set (new to v1.26):

```
  \glslinkpresetkeys
```

Set options.

```
  \setkeys{glslink}{#1}%
```

Perform auto add if set (new to v1.37)

```
  \glsxtr@do@autoadd{glslink}%
```

User hook after options are set:

```
\glslinkpostsetkeys
```

Check thevalue and theHvalue before saving (v1.19).

```
\ifdefempty{\@glsxtr@thevalue}%
{%
  \@gls@saveentrycounter
}%
{%
  \let\theglsentrycounter\@glsxtr@thevalue
  \def\theHglsentrycounter{\@glsxtr@theHvalue}%
}%
\@gls@setsort{\glslabel}%
```

Check if the textformat key has been used.

```
\ifx\@glsxtr@local@textformat\relax
```

Check textformat attribute (new to v1.21).

```
   \glshasattribute{\glslabel}{textformat}%
   {%
     \edef\@glsxtr@attrval{\glsgetattribute{\glslabel}{textformat}}%
     \ifcsdef{\@glsxtr@attrval}%
     {%
       \letcs{\@glsxtr@textformat}{\@glsxtr@attrval}%
     }%
     {%
       \GlossariesExtraWarning{Unknown control sequence name
       ‘\@glsxtr@attrval’ supplied in textformat attribute
       for entry ‘\glslabel’. Reverting to default \string\glstextformat}%
       \let\@glsxtr@textformat\glstextformat
     }%
   }%
   {%
     \let\@glsxtr@textformat\glstextformat
   }%
\else
   \let\@glsxtr@textformat\@glsxtr@local@textformat
\fi
```

Do write if it should occur before the link text:

```
\ifglsxtrinitwrglossbefore
  \@do@wrglossary{#2}%
\fi
```

Do the link text:

```
\ifKV@glslink@hyper
  \ifglsxtr@hyperoutside
    \@glslink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
  \else
    \@glsxtr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
  \fi
\else
  \ifglsxtr@hyperoutside
    \glsdonohyperlink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
  \else
    \@glsxtr@textformat{\glsdonohyperlink{\glolinkprefix\glslabel}{#3}}%
  \fi
\fi
```

Do write if it should occur after the link text:

```
  \ifglsxtrinitwrglossbefore
  \else
    \@do@wrglossary{#2}%
  \fi
```

Restore original value of `\glolinkprefix`:

```
  \let\glolinkprefix\@glsxtr@org@glolinkprefix
```

As the original definition:

```
  \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
}
```

```
\define@key{glossadd}{thevalue}{\def\@glsxtr@thevalue{#1}}
```

```
\define@key{glossadd}{theHvalue}{\def\@glsxtr@theHvalue{#1}}
```

`lsaddpresetkeys`

```
\newcommand*{\glsaddpresetkeys}{}
```

`saddpostsetkeys`

```
\newcommand*{\glsaddpostsetkeys}{}
```

`\glsadd` Redefine to include `\@glsxtr@record` and suppress in headings

```
\renewrobustcmd*{\glsadd}[2][]{%
  \glsxtrifinmark
  {}%
  {%
    \@gls@adjustmode
    \begingroup
      \@glsxtr@record{#1}{#2}{glossadd}%
      \glsdoifexists{#2}%
      {%
        \let\@glsnumberformat\@glsxtr@defaultnumberformat
        \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
        \def\@glsxtr@thevalue{}%
        \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

Implement any default settings (before options are set)

```
        \glsaddpresetkeys
        \setkeys{glossadd}{#1}%
```

Implement any default settings (after options are set)

```
        \glsaddpostsetkeys
        \ifdefempty{\@glsxtr@thevalue}%
        {%
          \@gls@saveentrycounter
        }%
        {%
          \let\theglsentrycounter\@glsxtr@thevalue
          \def\theHglsentrycounter{\@glsxtr@theHvalue}%
        }%
```

Define sort key if necessary (in case of sort=use):

```
        \@gls@setsort{#2}%
```

Ensure that indexing occurs (since that's the point of `\glsadd`). If indexing has been switched off by default, don't want the setting to affect `\glsadd`. The ignored format `\glsignore` can be used for selection without location, but the indexing still needs to be performed.

```
        \KV@glslink@noindexfalse
        \@@do@wrglossary{#2}%
      }%
    \endgroup
  }%
}
```

`\glsaddeach` Performs `\glsadd` for each entry listed in the mandatory argument.

```
\newrobustcmd{\glsaddeach}[2][]{%
  \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
}
```

`@field@linkdefs` Default settings for `\@gls@field@link`

```
\newcommand*{\@glsxtr@field@linkdefs}{%
  \let\glsxtrifwasfirstuse\@secondoftwo
  \let\glsifplural\@secondoftwo
  \let\glscapscase\@firstofthree
  \let\glsinsert\@empty
}
```

Redefine the field link commands that need to modify the above. Also add accessibility support and set the abbreviation styles if required.

`assignfieldfont`

```
\newcommand*{\glsxtrassignfieldfont}[1]{%
  \ifglsentryexists{#1}%
  {%
    \ifglshasshort{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glsifregular{#1}%
      {\let\@gls@field@font\glsxtrregularfont}%
      {\let\@gls@field@font\@firstofone}%
    }%
    {%
      \glsifnotregular{#1}%
      {\let\@gls@field@font\@firstofone}%
      {\let\@gls@field@font\glsxtrregularfont}%
    }%
  }%
  {%
    \let\@gls@field@font\@gobble
  }%
}
```

`\@glstext@` The abbreviation format may also need setting.

```
\def\@glstext@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
}
```

`\@GLStext@` All uppercase version of `\glstext`. The abbreviation format may also need setting.

```
\def\@GLStext@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
    {\@gls@field@font{\GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@Glstext@` First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glstext@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
    {\@gls@field@font{\Glsaccesstext{#2}#3}}%
}
```

Version 1.07 ensures that `\glsfirst` etc honours the nohyperfirst attribute. Allow a convenient way for the user to revert to ignoring this attribute for these commands.

`ecknohyperfirst`

```
\newcommand*{\glsxtrchecknohyperfirst}[1]{%
  \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
}
```

`\@glsfirst@` No case changing version. The abbreviation format may also need setting.

```
\def\@glsfirst@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\glsfirst` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \glsxtrchecknohyperfirst{#2}%
  ]{#1}{#2}%
  {\@gls@field@font{\glsaccessfirst{#2}#3}}%
}
```

`\@Glsfirst@` First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glsfirst@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\Glsfirst` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glscapscase\@secondofthree
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
}
```

`\@GLSfirst@` All uppercase version. The abbreviation format may also need setting.

```
\def\@GLSfirst@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\GLSfirst` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glscapscase\@thirdofthree
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@glsplural@` No case changing version. The abbreviation format may also need setting.

```
\def\@glsplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
    {\@gls@field@font{\glsaccessplural{#2}#3}}%
}
```

`\@Glsplural@` First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glsplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glsifplural\@firstoftwo
   \let\glscapscase\@secondofthree
  ]%
    {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
}
```

`\@GLSplural@` All uppercase version. The abbreviation format may also need setting.

```
\def\@GLSplural@#1#2[#3]{%
```

```
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glsifplural\@firstoftwo
   \let\glscapscase\@thirdofthree
  ]%
    {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
}
```

glsfirstplural@ No case changing version. The abbreviation format may also need setting.

```
\def\@glsfirstplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\glsfirstplural` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
}
```

Glsfirstplural@ First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glsfirstplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\glsfirstplural` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
   \let\glscapscase\@secondofthree
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
}
```

GLSfirstplural@ All uppercase version. The abbreviation format may also need setting.

```
\def\@GLSfirstplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\glsfirstplural` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
   \let\glscapscase\@thirdofthree
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}%
   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@glsname@` Redefine to use accessibility support. The abbreviation format may also need setting.

```
\def\@glsname@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
}
```

`\@Glsname@` First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glsname@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]{#1}{#2}%
  {\@gls@field@font{\Glsaccessname{#2}#3}}%
}
```

`\@GLSname@` All uppercase version. The abbreviation format may also need setting.

```
\def\@GLSname@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}%
    {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@glsdesc@`

```
\def\@glsdesc@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
}
```

`\@Glsdesc@` First letter uppercase version.

```
\def\@Glsdesc@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]{#1}{#2}%
  {\@gls@field@font{\Glsaccessdesc{#2}#3}}%
}
```

`\@GLSdesc@` All uppercase version.

```
\def\@GLSdesc@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`@glsdescplural@` No case-changing version.

```
\def\@glsdescplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo
```

```
   \let\glsifplural\@firstoftwo
  ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
}
```

`@Glsdescplural@` First letter uppercase version.

```
\def\@Glsdescplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo
   \let\glsifplural\@firstoftwo
  ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
}
```

`@GLSdescplural@` All uppercase version.

```
\def\@GLSdesc@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@thirdoftwo
   \let\glsifplural\@firstoftwo
  ]%
    {#1}{#2}%
    {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@glssymbol@`

```
\def\@glssymbol@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
}
```

`\@Glssymbol@` First letter uppercase version.

```
\def\@Glssymbol@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
}
```

`\@GLSsymbol@` All uppercase version.

```
\def\@GLSsymbol@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`lssymbolplural@` No case-changing version.

```
\def\@glssymbolplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo
   \let\glsifplural\@firstoftwo
  ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
}
```

`lssymbolplural@` First letter uppercase version.

```
\def\@Glssymbolplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo
   \let\glsifplural\@firstoftwo
  ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
}
```

`LSsymbolplural@` All uppercase version.

```
\def\@GLSsymbol@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@thirdoftwo
   \let\glsifplural\@firstoftwo
  ]%
    {#1}{#2}%
    {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@Glsuseri@` First letter uppercase version.

```
\def\@Glsuseri@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]{#1}{#2}%
  {\@gls@field@font{\Glsentryuseri{#2}#3}}%
}
```

`\@GLSuseri@` All uppercase version.

```
\def\@GLSuseri@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}}%
}
```

`\@Glsuserii@` First letter uppercase version.

```
\def\@Glsuserii@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
}
```

\@GLSuserii@ All uppercase version.

```
\def\@GLSuserii@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}}%
}
```

\@Glsuseriii@ First letter uppercase version.

```
\def\@Glsuseriii@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
}
```

\@GLSuseriii@ All uppercase version.

```
\def\@GLSuseriii@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}}%
}
```

\@Glsuseriv@ First letter uppercase version.

```
\def\@Glsuseriv@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
}
```

\@GLSuseriv@ All uppercase version.

```
\def\@GLSuseriv@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}%
    {\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}}%
}
```

\@Glsuserv@ First letter uppercase version.

```
\def\@Glsuserv@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
}
```

\@GLSuserv@ All uppercase version.

```
\def\@GLSuserv@#1#2[#3]{%
```

```
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}}%
}
```

`\@Glsuservi@` First letter uppercase version.

```
\def\@Glsuservi@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
}
```

`\@GLSuservi@` All uppercase version.

```
\def\@GLSuservi@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}}%
}
```

Commands like `\acrshort` already set `\glsifplural`, but they don't set `\glsxtrifwasfirstuse` so they need adjusting.

`\@acrshort` No case change.

```
\def\@acrshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsaccessshort{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@Acrshort` First letter uppercase.

```
\def\@Acrshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsaccessshort{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@ACRshort` All uppercase.

```
\def\@ACRshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsaccessshort{#2}}#3}%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@acrshortpl` No case change.

```
\def\@acrshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsaccessshortpl{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@Acrshortpl` First letter uppercase.

```
\def\@Acrshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsaccessshortpl{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@ACRshortpl` All uppercase.

```
\def\@ACRshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsaccessshortpl{#2}}#3}%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@acrlong` No case change.

```
\def\@acrlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsaccesslong{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@Acrlong` First letter uppercase.

```
\def\@Acrlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsaccesslong{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@ACRlong` All uppercase.

```
\def\@ACRlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsaccesslong{#2}}#3}%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@acrlongpl` No case change.

```
\def\@acrlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsaccesslongpl{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@Acrlongpl` First letter uppercase.

```
\def\@Acrlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsaccesslongpl{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@ACRlongpl` All uppercase.

```
\def\@ACRlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsaccesslongpl{#2}}#3}%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

Modify `\@glsaddkey` so additional keys provided by the user can be treated in a similar way.

`\@glsaddkey`

```
\renewcommand*{\@glsaddkey}[7]{%
  \key@ifundefined{glossentry}{#1}%
  {%
    \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
    \appto\@gls@keymap{,{#1}{#1}}%
    \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
    \appto\@newglossaryentryposthook{%
      \letcs{\@glo@tmp}{@glo@#1}%
      \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
    }%
    \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
    \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

Now for the commands with links. First the version with no case change (same as before):

```
    \ifcsdef{@gls@user@#1@}%
```

```
  {%
     \PackageError{glossaries}%
     {Can't define `\string#5' as helper command
      `\expandafter\string\csname @gls@user@#1@\endcsname' already
      exists}%
     {}%
  }%
  {%
    \expandafter\newcommand\expandafter*\expandafter
      {\csname @gls@user@#1\endcsname}[2][]{%
        \new@ifnextchar[%
          {\csuse{@gls@user@#1@}{##1}{##2}}%
          {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
    \csdef{@gls@user@#1@}##1##2[##3]{%
      \@gls@field@link{##1}{##2}{#3{##2}##3}%
    }%
    \newrobustcmd*{#5}{%
      \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
  }%
```

Next the version with the first letter converted to upper case (modified):

```
  \ifcsdef{@Gls@user@#1@}%
  {%
     \PackageError{glossaries}%
     {Can't define `\string#6' as helper command
      `\expandafter\string\csname @Gls@user@#1@\endcsname' already
       exists}%
     {}%
  }%
  {%
    \expandafter\newcommand\expandafter*\expandafter
      {\csname @Gls@user@#1\endcsname}[2][]{%
        \new@ifnextchar[%
          {\csuse{@Gls@user@#1@}{##1}{##2}}%
          {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
    \csdef{@Gls@user@#1@}##1##2[##3]{%
      \@gls@field@link[\let\glscapscase\@secondofthree]%
        {##1}{##2}{#4{##2}##3}%
    }%
    \newrobustcmd*{#6}{%
      \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
  }%
```

Finally the all caps version (modified):

```
  \ifcsdef{@GLS@user@#1@}%
  {%
     \PackageError{glossaries}%
     {Can't define `\string#7' as helper command
      `\expandafter\string\csname @GLS@user@#1@\endcsname' already
       exists}%
       {}%
    }%
    {%
      \expandafter\newcommand\expandafter*\expandafter
        {\csname @GLS@user@#1\endcsname}[2][]{%
          \new@ifnextchar[%
            {\csuse{@GLS@user@#1@}{##1}{##2}}%
            {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
      \csdef{@GLS@user@#1@}##1##2[##3]{%
        \@gls@field@link[\let\glscapscase\@thirdofthree]%
           {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
      }%
      \newrobustcmd*{#7}{%
        \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
    }%
  }%
  {%
    \PackageError{glossaries-extra}{Key '#1' already exists}{}%
  }%
}
```

`checkfirsthyper` Old versions of glossaries don't define this, so provide it just in case it hasn't been defined.

```
\providecommand*{\@gls@link@nocheckfirsthyper}{}
```

`checkfirsthyper` Modify check to determine if the hyperlink should be automatically suppressed, but save the original in case the acronyms are restored.

```
\let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
\renewcommand*{\@gls@link@checkfirsthyper}{%
```

`\ifglsused` isn't useful in the post link hook as it's already been unset by then, so define a command that can be used in the post link hook. Since `\@gls@link@checkfirsthyper` is only used by commands like `\gls` but not by other commands, this seems the best place to put it to automatically set the value for the commands that change the first use flag. The other commands should set `\glsxtrifwasfirstuse` to `\@secondoftwo` (which is done in `\@glsxtr@field@linkdefs`).

```
  \ifglsused{\glslabel}%
   {\let\glsxtrifwasfirstuse\@secondoftwo}
   {\let\glsxtrifwasfirstuse\@firstoftwo}%
```

Store the category label for convenience.

```
  \edef\glscategorylabel{\glscategory{\glslabel}}%
  \ifglsused{\glslabel}%
  {%
    \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
      {\KV@glslink@hyperfalse}{}%
  }%
  {%
    \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
      {\KV@glslink@hyperfalse}{}%
  }%
  \glslinkcheckfirsthyperhook
}
```

ablehyperinlist

This command was introduced in glossaries v4.19. If it hasn't been defined, we're using an earlier version, in which case the nohyper attribute can't be implemented.

```
\ifdef\do@glsdisablehyperinlist
{%
  \let\@glsxtr@do@glsdisablehyperinlist\do@glsdisablehyperinlist
  \renewcommand*{\do@glsdisablehyperinlist}{%
    \@glsxtr@do@glsdisablehyperinlist
    \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
  }
}
{}
```

Define a noindex key to prevent writing information to the external file.

```
\define@boolkey{glslink}{noindex}[true]{}
\KV@glslink@noindexfalse
```

If `\@gls@setdefault@glslink@opts` has been defined (glossaries v4.20) use it to set the default keys in `\@glslink`.

lt@glslink@opts

```
\ifdef\@gls@setdefault@glslink@opts
{
  \renewcommand*{\@gls@setdefault@glslink@opts}{%
    \KV@glslink@noindexfalse
    \@glsxtrsetaliasnoindex
  }
}
{
```

Not defined so prepend it to `\do@glsdisablehyperinlist` to achieve the same effect.

```
  \newcommand*{\@gls@setdefault@glslink@opts}{%
    \KV@glslink@noindexfalse
    \@glsxtrsetaliasnoindex
  }
  \preto\do@glsdisablehyperinlist{\@gls@setdefault@glslink@opts}
}
```

setaliasnoindex

Allow user to hook into the alias noindex setting. Default behaviour switches off indexing for aliases. If the record option is on, this will have been defined to do nothing. (bib2gls will deal with records for aliased entries.)

```
\providecommand*{\glsxtrsetaliasnoindex}{%
 \KV@glslink@noindextrue
}
```

setaliasnoindex

```
\newcommand*{\@glsxtrsetaliasnoindex}{%
 \glsxtrifhasfield{alias}{\glslabel}%
 {%
   \let\glsxtrindexaliased\@glsxtrindexaliased
   \glsxtrsetaliasnoindex
   \let\glsxtrindexaliased\@no@glsxtrindexaliased
 }%
 {}%
}
```

`xtrindexaliased`

```
\newcommand{\@glsxtrindexaliased}{%
 \ifKV@glslink@noindex
 \else
   \begingroup
   \let\@glsnumberformat\@glsxtr@defaultnumberformat
   \edef\@gls@counter{\csname glo@\glsdetoklabel{\glslabel}@counter\endcsname}%
   \glsxtr@saveentrycounter
   \@@do@wrglossary{\glsxtralias{\glslabel}}%
   \endgroup
 \fi
}
```

`xtrindexaliased`

```
\newcommand{\@no@glsxtrindexaliased}{%
  \PackageError{glossaries-extra}{\string\glsxtrindexaliased\space
  not permitted outside definition of \string\glsxtrsetaliasnoindex}%
  {}%
}
```

`xtrindexaliased` Provide a command to redirect alias indexing, but only allow it to be used within `\glsxtrsetaliasnoindex`.

```
\let\glsxtrindexaliased\@no@glsxtrindexaliased
```

`tDefaultGlsOpts` Set the default options for `\glslink` etc.

```
\newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
  \renewcommand*{\@gls@setdefault@glslink@opts}{%
    \setkeys{glslink}{#1}%
    \@glsxtrsetaliasnoindex
  }%
}
```

`lsxtrifindexing` Provide user level command to access it in `\glswriteentry`.

```
\newcommand*{\glsxtrifindexing}[2]{%
  \ifKV@glslink@noindex #2\else #1\fi
}
```

`\glswriteentry` Redefine to test for indexonlyfirst category attribute.

```
\renewcommand*{\glswriteentry}[2]{%
```

```
  \glsxtrifindexing
  {%
   \ifglsindexonlyfirst
     \ifglsused{#1}
     {\glsxtrdoautoindexname{#1}{dualindex}}%
     {#2}%
    \else
     \glsifattribute{#1}{indexonlyfirst}{true}%
     {\ifglsused{#1}
      {\glsxtrdoautoindexname{#1}{dualindex}}%
      {#2}}%
     {#2}%
    \fi
  }%
  {}%
}
```

@do@@wrglossary

Hook into glossary indexing command so that it can also use `\index` at the same time if required and add user hook.

```
\appto\@@do@@wrglossary{\@glsxtr@do@@wrindex
  \glsxtrdowrglossaryhook{\@gls@label}%
}
```

(The label can be obtained from `\@gls@label` at this point.)

Similarly for the “noidx” version:

s@noidxglossary

```
\appto\gls@noidxglossary{\@glsxtr@do@@wrindex
  \glsxtrdowrglossaryhook{\@gls@label}%
}
```

xtr@do@@wrindex

```
\newcommand*{\@glsxtr@do@@wrindex}{%
  \glsxtrdoautoindexname{\@gls@label}{dualindex}%
}
```

owrglossaryhook

Allow user to hook into indexing code. (Always used by `\glsadd`. Used by `\gls` when indexing, which may or may not occur depending on the indexing settings.)

```
\newcommand*{\glsxtrdowrglossaryhook}[1]{}
```

gls@alt@hyp@opt

Commands like `\gls` have a star or plus version. Provide a third symbol that the user can adapt for convenience.

```
\newcommand*{\@gls@alt@hyp@opt}[1]{%
 \let\glslinkvar\@firstofthree
 \let\@gls@hyp@opt@cs#1\relax
 \@ifstar{\s@gls@hyp@opt}%
 {\@ifnextchar+%
   {\@firstoftwo{\p@gls@hyp@opt}}%
   {%
     \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
     {\@firstoftwo{\@alt@gls@hyp@opt}}%
     {#1}%
   }%
 }%
}
```

`alt@gls@hyp@opt` User version

```
\newcommand*{\@alt@gls@hyp@opt}[1][]{%
 \let\glslinkvar\@firstofthree
 \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

`lt@hyp@opt@char` Contains the character used as the command modifier.

```
\newcommand*{\@gls@alt@hyp@opt@char}{}
```

`lt@hyp@opt@keys` Contains the option list used as the command modifier.

```
\newcommand*{\@gls@alt@hyp@opt@keys}{}
```

`rSetAltModifier`

```
\newcommand*{\GlsXtrSetAltModifier}[2]{%
  \let\@gls@hyp@opt\@gls@alt@hyp@opt
  \def\@gls@alt@hyp@opt@char{#1}%
  \def\@gls@alt@hyp@opt@keys{#2}%
  \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
  {}%
  {%
```

Let `bib2gls` know the modifier.

```
    \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@altmodifier}[1]{}}%
    \protected@write\@auxout{}{\string\@glsxtr@altmodifier{#1}}%
  }%
}
```

`org@dohyperlink`

```
\let\glsxtr@org@dohyperlink\glsdohyperlink
```

`glsnavhyperlink` Now that `\glsdohyperlink` (used by `\@glslink`) references `\glslabel` it's necessary to patch `\glsnavhyperlink` to avoid using it (since `\glslabel` won't be defined). This means temporarily redefining `\glsdohyperlink` to its original definition.

This command is provided by glossary-hypernav so it may not exist.

```
\ifdef\glsnavhyperlink
{
  \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
    \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
```

Scope:

```
    {%
      \let\glsdohyperlink\glsxtr@org@dohyperlink
      \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
    }%
  }%
}
{}
```

`\glsdohyperlink` Unpleasant complications can occur if the text or first key etc contains `\gls`, particularly if there are hyperlinks. To get around this problem, patch `\glsdohyperlink` so that it temporarily makes `\gls` behave like `\glstext[`⟨*hyper=false,noindex*⟩`]`. (This will be overridden if the user explicitly cancels either of those options in the optional argument of `\gls` or using the plus version.) This also patches the short form commands like `\acrshort` and `\glsxtrshort` to use `\glsentryshort` and, similarly, the long form commands like `\acrlong` and `\glsxtrlong` to use `\glsentrylong`. Added attribute check.

```
\renewcommand*{\glsdohyperlink}[2]{%
 \glshasattribute{\glslabel}{targeturl}%
 {%
   \glshasattribute{\glslabel}{targetname}%
   {%
     \glshasattribute{\glslabel}{targetcategory}%
     {%
       \hyperref{\glsgetattribute{\glslabel}{targeturl}}%
         {\glsgetattribute{\glslabel}{targetcategory}}%
         {\glsgetattribute{\glslabel}{targetname}}%
         {{\glsxtrprotectlinks#2}}%
     }%
     {%
       \hyperref{\glsgetattribute{\glslabel}{targeturl}}%
         {}%
         {\glsgetattribute{\glslabel}{targetname}}%
         {{\glsxtrprotectlinks#2}}%
     }%
   }%
   {%
     \href{\glsgetattribute{\glslabel}{targeturl}}%
      {{\glsxtrprotectlinks#2}}%
   }%
 }%
 {%
```

Check for alias.

```
   \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
   \ifdefvoid\gloaliaslabel
   {%
     \glsxtrhyperlink{#1}{{\glsxtrprotectlinks#2}}%
   }%
   {%
```

Redirect link to the alias target.

```
     \glsxtrhyperlink
     {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
     {{\glsxtrprotectlinks#2}}%
   }%
 }%
}
```

glsxtrhyperlink Allows integration with the base glossaries package's debug=showtargets option.

```
\ifdef\@glsshowtarget
{
  \newcommand{\glsxtrhyperlink}[2]{%
    \@glsshowtarget{#1}%
    \hyperlink{#1}{#2}%
  }%
}
{
  \newcommand{\glsxtrhyperlink}[2]{\hyperlink{#1}{#2}}%
}
```

glsdisablehyper Redefine to set `\glslabel` (to allow it to be picked up by `\glsdohyperlink`). Also made it robust and added grouping to localise the definition of `\glslabel`. The original internal command @glo@label could probably be simply replaced with `\glslabel`, but it's retained in case its removal causes unexpected problems.

```
\renewrobustcmd*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
 \glsdoifexists{#2}%
 {%
   \def\@glo@label{#2}%
   {\edef\glslabel{#2}%
   \@glslink{\glolinkprefix\glslabel}{#1}}%
 }%
}
```

glsdisablehyper Redefine in case we have an old version of glossaries. This now uses `\def` rather than `\let` to allow for redefinitions of `\glsdonohyperlink`.

```
\renewcommand{\glsdisablehyper}{%
  \KV@glslink@hyperfalse
  \def\@glslink{\glsdonohyperlink}%
  \let\@glstarget\@secondoftwo
}
```

\glsenablehyper This now uses `\def` rather than `\let` to allow for redefinitions of `\glsdohypertarget` and `\glsdohyperlink`.

```
\renewcommand{\glsenablehyper}{%
 \KV@glslink@hypertrue
 \def\@glslink{\glsdohyperlink}%
 \def\@glstarget{\glsdohypertarget}%
}
```

lsdonohyperlink This command was only introduced in glossaries v4.20, so it may not be defined (therefore use `\def`). For older glossaries versions, this won't be used if hyperref hasn't been loaded, which means the indexing will still take place. The generated text is scoped (the link text in `\hyperlink` is also scoped, so it's consistent).

```
\def\glsdonohyperlink#1#2{{\glsxtrprotectlinks #2}}
```

`\@glslink` Reset `\@glslink` with patched versions:

```
\ifcsundef{hyperlink}%
{%
  \def\@glslink{\glsdonohyperlink}
}%
{%
  \def\@glslink{\glsdohyperlink}
}
```

`xtrprotectlinks` Make `\gls` (and variants) behave like the corresponding `\glstext` (and variants) with hyperlinking and indexing off.

```
\newcommand*{\glsxtrprotectlinks}{%
  \KV@glslink@hyperfalse
  \KV@glslink@noindextrue
  \let\@gls@\@glsxtr@p@text@
  \let\@Gls@\@Glsxtr@p@text@
  \let\@GLS@\@GLSxtr@p@text@
  \let\@glspl@\@glsxtr@p@plural@
  \let\@Glspl@\@Glsxtr@p@plural@
  \let\@GLSpl@\@GLSxtr@p@plural@
  \let\@glsxtrshort\@glsxtr@p@short@
  \let\@Glsxtrshort\@Glsxtr@p@short@
  \let\@GLSxtrshort\@GLSxtr@p@short@
  \let\@glsxtrlong\@glsxtr@p@long@
  \let\@Glsxtrlong\@Glsxtr@p@long@
  \let\@GLSxtrlong\@GLSxtr@p@long@
  \let\@glsxtrshortpl\@glsxtr@p@shortpl@
  \let\@Glsxtrshortpl\@Glsxtr@p@shortpl@
  \let\@GLSxtrshortpl\@GLSxtr@p@shortpl@
  \let\@glsxtrlongpl\@glsxtr@p@longpl@
  \let\@Glsxtrlongpl\@Glsxtr@p@longpl@
  \let\@GLSxtrlongpl\@GLSxtr@p@longpl@
  \let\@acrshort\@glsxtr@p@acrshort@
  \let\@Acrshort\@Glsxtr@p@acrshort@
  \let\@ACRshort\@GLSxtr@p@acrshort@
  \let\@acrshortpl\@glsxtr@p@acrshortpl@
  \let\@Acrshortpl\@Glsxtr@p@acrshortpl@
  \let\@ACRshortpl\@GLSxtr@p@acrshortpl@
  \let\@acrlong\@glsxtr@p@acrlong@
  \let\@Acrlong\@Glsxtr@p@acrlong@
  \let\@ACRlong\@GLSxtr@p@acrlong@
  \let\@acrlongpl\@glsxtr@p@acrlongpl@
  \let\@Acrlongpl\@Glsxtr@p@acrlongpl@
  \let\@ACRlongpl\@GLSxtr@p@acrlongpl@
}
```

These protected versions need grouping to prevent the label from getting confused.

@glsxtr@p@text@

```
\def\@glsxtr@p@text@#1#2[#3]{{\@glstext@{#1}{#2}[#3]}}
```

@Glsxtr@p@text@

```
\def\@Glsxtr@p@text@#1#2[#3]{{\@Glstext@{#1}{#2}[#3]}}
```

@GLSxtr@p@text@

```
\def\@GLSxtr@p@text@#1#2[#3]{{\@GLStext@{#1}{#2}[#3]}}
```

lsxtr@p@plural@

```
\def\@glsxtr@p@plural@#1#2[#3]{{\@glsplural@{#1}{#2}[#3]}}
```

lsxtr@p@plural@

```
\def\@Glsxtr@p@plural@#1#2[#3]{{\@Glsplural@{#1}{#2}[#3]}}
```

LSxtr@p@plural@

```
\def\@GLSxtr@p@plural@#1#2[#3]{{\@GLSplural@{#1}{#2}[#3]}}
```

glsxtr@p@short@

```
\def\@glsxtr@p@short@#1#2[#3]{%
 {%
  \glssetabbrvfmt{\glscategory{#2}}%
  \glsabbrvfont{\glsentryshort{#2}}#3%
 }%
}
```

Glsxtr@p@short@

```
\def\@Glsxtr@p@short@#1#2[#3]{%
 {%
   \glssetabbrvfmt{\glscategory{#2}}%
   \glsabbrvfont{\Glsentryshort{#2}}#3%
 }%
}
```

GLSxtr@p@short@

```
\def\@GLSxtr@p@short@#1#2[#3]{%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshort{#2}}#3}%
  }%
}
```

sxtr@p@shortpl@

```
\def\@glsxtr@p@shortpl@#1#2[#3]{%
 {%
   \glssetabbrvfmt{\glscategory{#2}}%
```

```
    \glsabbrvfont{\glsentryshortpl{#2}}#3%
  }%
}
```

sxtr@p@shortpl@

```
\def\@Glsxtr@p@shortpl@#1#2[#3]{%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \glsabbrvfont{\Glsentryshortpl{#2}}#3%
  }%
}
```

Sxtr@p@shortpl@

```
\def\@GLSxtr@p@shortpl@#1#2[#3]{%
   {%
     \glssetabbrvfmt{\glscategory{#2}}%
     \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshortpl{#2}}#3}%
   }%
}
```

@glsxtr@p@long@

```
\def\@glsxtr@p@long@#1#2[#3]{{\glsentrylong{#2}#3}}
```

@Glsxtr@p@long@

```
\def\@Glsxtr@p@long@#1#2[#3]{{\Glsentrylong{#2}#3}}
```

@GLSxtr@p@long@

```
\def\@GLSxtr@p@long@#1#2[#3]{%
   {\mfirstucMakeUppercase{\glslongfont{\glsentrylong{#2}}#3}}}
```

lsxtr@p@longpl@

```
\def\@glsxtr@p@longpl@#1#2[#3]{{\glsentrylongpl{#2}#3}}
```

lsxtr@p@longpl@

```
\def\@Glsxtr@p@longpl@#1#2[#3]{{\glslongfont{\Glsentrylongpl{#2}}#3}}
```

LSxtr@p@longpl@

```
\def\@GLSxtr@p@longpl@#1#2[#3]{%
   {\mfirstucMakeUppercase{\glslongfont{\glsentrylongpl{#2}}#3}}}
```

xtr@p@acrshort@

```
\def\@glsxtr@p@acrshort@#1#2[#3]{{\acronymfont{\glsentryshort{#2}}#3}}
```

xtr@p@acrshort@

```
\def\@Glsxtr@p@acrshort@#1#2[#3]{{\acronymfont{\Glsentryshort{#2}}#3}}
```

xtr@p@acrshort@

```
\def\@GLSxtr@p@acrshort@#1#2[#3]{%
   {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}
```

r@p@acrshortpl@

```
\def\@glsxtr@p@acrshortpl@#1#2[#3]{{\acronymfont{\glsentryshortpl{#2}}#3}}
```

r@p@acrshortpl@

```
\def\@Glsxtr@p@acrshortpl@#1#2[#3]{{\acronymfont{\Glsentryshortpl{#2}}#3}}
```

r@p@acrshortpl@

```
\def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
  {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}
```

sxtr@p@acrlong@

```
\def\@glsxtr@p@acrlong@#1#2[#3]{{\glsentrylong{#2}#3}}
```

sxtr@p@acrlong@

```
\def\@Glsxtr@p@acrlong@#1#2[#3]{{\Glsentrylong{#2}#3}}
```

Sxtr@p@acrlong@

```
\def\@GLSxtr@p@acrlong@#1#2[#3]{%
 {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}
```

tr@p@acrlongpl@

```
\def\@glsxtr@p@acrlongpl@#1#2[#3]{{\glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
\def\@Glsxtr@p@acrlongpl@#1#2[#3]{{\Glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
\def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
 {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}
```

Commands to minimise conflict.

`\@glsxtrp@opt`

```
\newcommand*{\@glsxtrp@opt}{hyper=false,noindex}
```

`\glsxtrsetpopts` Used in glossary to switch hyperlinks on for the `\glsxtrp` type of commands.

```
\newcommand*{\glsxtrsetpopts}[1]{%
  \renewcommand*{\@glsxtrp@opt}{#1}%
}
```

`lossxtrsetpopts` Used in glossary to switch hyperlinks on for the `\glsxtrp` type of commands.

```
\newcommand*{\glossxtrsetpopts}{%
  \glsxtrsetpopts{noindex}%
}
```

`\@@glsxtrp`

```
\newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
  {%
    \let\glspostlinkhook\relax
    \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
  }%
}
```

\@glsxtrp

```
\newrobustcmd*{\@glsxtrp}[2]{%
  \ifcsdef{gls#1}%
  {%
    \@@glsxtrp{gls#1}{#2}%
  }%
  {%
    \ifcsdef{glsxtr#1}%
    {%
      \@@glsxtrp{glsxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{‘#1’ not recognised by
        \string\glsxtrp}{}%
    }%
  }%
}
```

\@Glsxtrp

```
\newrobustcmd*{\@Glsxtrp}[2]{%
  \ifcsdef{Gls#1}%
  {%
    \@@glsxtrp{Gls#1}{#2}%
  }%
  {%
    \ifcsdef{Glsxtr#1}%
    {%
      \@@glsxtrp{Glsxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{‘#1’ not recognised by
        \string\Glsxtrp}{}%
    }%
  }%
}
```

\@GLSxtrp

```
\newrobustcmd*{\@GLSxtrp}[2]{%
  \ifcsdef{GLS#1}%
  {%
    \@@glsxtrp{GLS#1}{#2}%
  }%
  {%
    \ifcsdef{GLSxtr#1}%
    {%
      \@@glsxtrp{GLSxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{‘#1’ not recognised by
        \string\GLSxtrp}{}%
    }%
  }%
}
```

`\glsxtr@entry@p`

```
\newrobustcmd*{\glsxtr@headentry@p}[2]{%
 \glsifattribute{#1}{headuc}{true}%
 {%
   \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
 }%
 {%
   \@gls@entry@field{#1}{#2}%
 }%
}
```

`\glsxtrp` Not robust as it needs to expand somewhat.

```
\ifdef\texorpdfstring
{
  \newcommand{\glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{glsxtrhead#1}%
          {%
            {\protect\csuse{glsxtrhead#1}{#2}}%
          }%
          {%
            \glsxtr@headentry@p{#2}{#1}%
          }%
        }%
        {%
          \@glsxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
  }
}
{
  \newcommand{\glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
      {%
        \ifcsdef{glsxtrhead#1}%
        {%
          {\protect\csuse{glsxtrhead#1}}%
        }%
        {%
          \glsxtr@headentry@p{#2}{#1}%
        }%
      }%
      {%
        \@glsxtrp{#1}{#2}%
      }%
    }%
  }
}
```

Provide short synonyms for the most common option.

`\glsps`

```
\newcommand*{\glsps}{\glsxtrp{short}}
```

`\glspt`

```
\newcommand*{\glspt}{\glsxtrp{text}}
```

`\Glsxtrp` As above but use first letter upper case (but not for the bookmarks, which can't process `\uppercase`).

```
\ifdef\texorpdfstring
{
  \newcommand{\Glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{Glsxtrhead#1}%
          {%
            {\protect\csuse{Glsxtrhead#1}{#2}}%
          }%
          {%
            \protect\@Gls@entry@field{#2}{#1}%
          }%
        }%
        {%
          \@Glsxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
  }
}
{
  \newcommand{\Glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
      {%
        \ifcsdef{Glsxtrhead#1}%
        {%
          {\protect\csuse{Glsxtrhead#1}}%
        }%
        {%
          \protect\@Gls@entry@field{#2}{#1}%
        }%
      }%
      {%
        \@Glsxtrp{#1}{#2}%
      }%
    }%
  }
}
```

`\GLSxtrp` As above but all upper case (but not for the bookmarks, which can't process `\uppercase`).

```
\ifdef\texorpdfstring
{
  \newcommand{\GLSxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{GLSxtr#1}%
          {%
            {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
          }%
          {%
            \protect\mfirstucMakeUppercase
            {%
```

```
              \protect\@gls@entry@field{#2}{#1}%
            }%
          }%
        }%
        {%
          \@GLSxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
  }
}
{
  \newcommand{\GLSxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
      {%
        \ifcsdef{GLSxtr#1}%
        {%
          {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
        }%
        {%
          \protect\mfirstucMakeUppercase
          {%
            \protect\@gls@entry@field{#2}{#1}%
          }%
        }%
      }%
      {%
        \@GLSxtrp{#1}{#2}%
      }%
    }%
  }
}
```

1.3.5 Entry Counting

The (use) entry counting mechanism from glossaries is adjusted here to work with category attributes. Provide a convenient command to enable entry counting, set the entrycount attribute for given categories and redefine `\gls` etc to use `\cgls` instead. This form of entry counting is provided to adjust the formatting if the number of times an entry has been used (through commands that unset the first use flag) doesn't exceeding the specified threshold. For link counting, see Section 1.4.

First adjust definitions of the unset and reset commands to provide a hook, but changing the flag can cause problems in certain situations, so to allow the normal unsetting to be temporarily disabled, `\@glsunset` is let to `\@glsxtr@unset`, which performs the actual unsetting through `\@@glsunset` and then does the hook. This means that the unsetting (and the hook) can switched off by redefining `\@glsunset` and then switched back on again by changing the definition back to `\@glsxtr@unset`.

`\@glsxtr@unset` Global unset.

```
\newcommand*{\@glsxtr@unset}[1]{%
  \@@glsunset{#1}%
  \glsxtrpostunset{#1}%
}%
```

`\@glsunset` Global unset.

```
\let\@glsunset\@glsxtr@unset
```

`glsxtrpostunset`

```
\newcommand*{\glsxtrpostunset}[1]{}
```

Provide a command to store a list of labels that will need unsetting.

`tUnsetBuffering`

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

`tUnsetBuffering` Unstarred version doesn't check for duplicates.

```
\newcommand*{\@GlsXtrStartUnsetBuffering}{%
  \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
  \def\@glsxtr@unset@buffer{}%
  \let\@glsunset\@glsxtrbuffer@unset
}
```

`tUnsetBuffering` Starred version checks for duplicates.

```
\newcommand*{\s@GlsXtrStartUnsetBuffering}{%
  \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
  \def\@glsxtr@unset@buffer{}%
  \let\@glsunset\@glsxtrbuffer@nodup@unset
}
```

`xtrbuffer@unset` This must use a global change since `\gls` may have to be placed inside `\mbox` (for example, with soul commands).

```
\newcommand*{\@glsxtrbuffer@unset}[1]{%
  \listxadd\@glsxtr@unset@buffer{#1}%
}
```

`fer@nodup@unset` Alternative version that avoids duplicates. One level of expansion is performed on the argument in case it's a control sequence containing the label. (Not using `\xifinlist` as the added complexity might cause problems that the buffering is trying to overcome.)

```
\newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
  \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}%
  {\listxadd\@glsxtr@unset@buffer{#1}}%
}
```

pUnsetBuffering

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

pUnsetBuffering Unstarred form (global unset).

```
\newcommand*{\@GlsXtrStopUnsetBuffering}{%
  \let\@glsunset\@glsxtr@unset
  \forlistloop\@glsunset\@glsxtr@unset@buffer
  \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
}
```

pUnsetBuffering Starred form (local unset).

```
\newcommand*{\s@GlsXtrStopUnsetBuffering}{%
  \forlistloop\@glslocalunset\@glsxtr@unset@buffer
  \let\@glsunset\@glsxtr@unset
}
```

setBufferedList Iterate over labels stored in the current buffer. The argument is the handler macro.

```
\newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
  \forlistloop#1\@glsxtr@unset@buffer
}
```

\@glslocalunset Local unset.

```
\renewcommand*{\@glslocalunset}[1]{%
  \@@glslocalunset{#1}%
  \glsxtrpostlocalunset{#1}%
}%
```

rpostlocalunset

```
\newcommand*{\glsxtrpostlocalunset}[1]{}
```

\@glsreset Global reset.

```
\renewcommand*{\@glsreset}[1]{%
  \@@glsreset{#1}%
  \glsxtrpostreset{#1}%
}%
```

glsxtrpostreset

```
\newcommand*{\glsxtrpostreset}[1]{}
```

`\@glslocalreset` Local reset.

```
\renewcommand*{\@glslocalreset}[1]{%
  \@@glslocalreset{#1}%
  \glsxtrpostlocalreset{#1}%
}%
```

`rpostlocalreset`

```
\newcommand*{\glsxtrpostlocalreset}[1]{}
```

`slocalreseteach` Locally reset a list of entries.

```
\newcommand*{\glslocalreseteach}[1]{%
  \gls@ifnotmeasuring
  {%
    \@for\@gls@thislabel:=#1\do{%
      \glsdoifexists{\@gls@thislabel}%
      {%
        \@glslocalreset{\@gls@thislabel}%
      }%
    }%
  }%
}
```

`slocalunseteach` Locally unset a list of entries.

```
\newcommand*{\glslocalunseteach}[1]{%
  \gls@ifnotmeasuring
  {%
    \@for\@gls@thislabel:=#1\do{%
      \glsdoifexists{\@gls@thislabel}%
      {%
        \@glslocalunset{\@gls@thislabel}%
      }%
    }%
  }%
}
```

`leEntryCounting` The first argument is the list of categories and the second argument is the value of the entrycount attribute.

```
\newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

Enable entry counting:

```
  \glsenableentrycount
```

Redefine `\gls` etc:

```
  \renewcommand*{\gls}{\cgls}%
  \renewcommand*{\Gls}{\cGls}%
  \renewcommand*{\glspl}{\cglspl}%
  \renewcommand*{\Glspl}{\cGlspl}%
  \renewcommand*{\GLS}{\cGLS}%
  \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
 \@glsxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
 \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
  \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
   can't be used with \string\GlsXtrEnableEntryCounting}%
  {Use one or other but not both commands}}%
}
```

ycountunsetattr

```
\newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
 \@for\@glsxtr@cat:=#1\do
 {%
   \ifdefempty{\@glsxtr@cat}{}%
   {%
     \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
   }%
 }%
}
```

Redefine the entry counting commands to take into account the entrycount attribute.

nableentrycount

```
\renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
 \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defcounters}%
```

Just in case the user has switched on the docdef option.

```
 \renewcommand*{\gls@defdocnewglossaryentry}{%
   \renewcommand*\newglossaryentry[2]{%
     \PackageError{glossaries}{\string\newglossaryentry\space
     may only be used in the preamble when entry counting has
     been activated}{If you use \string\glsenableentrycount\space
     you must place all entry definitions in the preamble not in
     the document environment}%
   }%
 }%
```

New commands to access new fields:

```
 \newcommand*{\glsentrycurrcount}[1]{%
  \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
  {0}{\@gls@entry@field{##1}{currcount}}%
 }%
 \newcommand*{\glsentryprevcount}[1]{%
  \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
  {0}{\@gls@entry@field{##1}{prevcount}}%
 }%
```

Adjust post unset and reset:

```
 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
 \renewcommand*{\glsxtrpostunset}[1]{%
   \@glsxtr@entrycount@org@unset{##1}%
   \@gls@increment@currcount{##1}%
 }%
 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
 \renewcommand*{\glsxtrpostlocalunset}[1]{%
   \@glsxtr@entrycount@org@localunset{##1}%
   \@gls@local@increment@currcount{##1}%
 }%
 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
 \renewcommand*{\glsxtrpostreset}[1]{%
   \@glsxtr@entrycount@org@reset{##1}%
   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
 }%
 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
 \renewcommand*{\glsxtrpostlocalreset}[1]{%
   \@glsxtr@entrycount@org@localreset{##1}%
   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
 }%
```

Modifications to take into account the attributes that govern whether the entry should be unset.

```
 \let\@cgls@\@@cgls@
 \let\@cglspl@\@@cglspl@
```

```
 \let\@cGls@\@@cGls@
 \let\@cGlspl@\@@cGlspl@
 \let\@cGLS@\@@cGLS@
 \let\@cGLSpl@\@@cGLSpl@
```

The rest is as the original definition.

```
 \AtEndDocument{\@gls@write@entrycounts}%
 \renewcommand*{\@gls@entry@count}[2]{%
   \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
 }%
 \let\glsenableentrycount\relax
 \renewcommand*{\glsenableentryunitcount}{%
   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
    can’t be used with \string\glsenableentrycount}%
   {Use one or other but not both commands}%
 }%
}
```

ite@entrycounts

Modify this command so that it only writes the information for entries with the entrycount attribute and issue warning if no entries have this attribute set.

```
\renewcommand*{\@gls@write@entrycounts}{%
 \immediate\write\@auxout
   {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
```

```
  \count@=0\relax
  \forallglsentries{\@glsentry}{%
    \glshasattribute{\@glsentry}{entrycount}%
    {%
      \ifglsused{\@glsentry}%
      {%
        \immediate\write\@auxout
         {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
      }%
      {}%
      \advance\count@ by \@ne
    }%
    {}%
  }%
  \ifnum\count@=0
    \GlossariesExtraWarningNoLine{Entry counting has been enabled
     \MessageBreak with \string\glsenableentrycount\space but the
     \MessageBreak attribute `entrycount' hasn't
     \MessageBreak been assigned to any of the defined
     \MessageBreak entries}%
  \fi
}
```

trifcounttrigger

`\glsxtrifcounttrigger{⟨label⟩}{⟨trigger format⟩}{⟨normal⟩}`

```
\newcommand*{\glsxtrifcounttrigger}[3]{%
 \glshasattribute{#1}{entrycount}%
 {%
   \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
    #3%
   \else
    #2%
   \fi
 }%
 {#3}%
}
```

Actual internal definitions of `\cgls` used when entry counting is enabled.

\@@cgls@

```
\def\@@cgls@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cglsformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@gls@{#1}{#2}[#3]%
  }%
}%
```

\@@cglspl@

```
\def\@@cglspl@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cglsplformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@glspl@{#1}{#2}[#3]%
  }%
}%
```

\@@cGls@

```
\def\@@cGls@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cGlsformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@Gls@{#1}{#2}[#3]%
  }%
}%
```

\@@cGlspl@

```
\def\@@cGlspl@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cGlsplformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@Glspl@{#1}{#2}[#3]%
  }%
}%
```

\@@cGLS@

```
\def\@@cGLS@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cGLSformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
```

```
    \@GLS@{#1}{#2}[#3]%
  }%
}%
```

\@@cGLSpl@

```
\def\@@cGLSpl@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cGLSplformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@GLSpl@{#1}{#2}[#3]%
  }%
}%
```

Remove default warnings from `\cgls` etc so that it can be used interchangeable with `\gls` etc.

\@cgls@

```
\def\@cgls@#1#2[#3]{\@gls@{#1}{#2}[#3]}
```

\@cGls@

```
\def\@cGls@#1#2[#3]{\@Gls@{#1}{#2}[#3]}
```

\@cglspl@

```
\def\@cglspl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}
```

\@cGlspl@

```
\def\@cGlspl@#1#2[#3]{\@Glspl@{#1}{#2}[#3]}
```

Add all upper case versions not provided by glossaries.

\cGLS

```
\newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}
```

\@cGLS Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@cGLS}[2][]{%
  \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[]}%
}
```

\@cGLS@

```
\def\@cGLS@#1#2[#3]{\@GLS@{#1}{#2}[#3]}
```

\cGLSformat Format used by `\cGLS` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
\newcommand*{\cGLSformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\cglsformat{#1}{#2}}%
}
```

\cGLSpl

```
\newrobustcmd*{\cGLSpl}{\@gls@hyp@opt\@cGLSpl}
```

\@cGLSpl Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@cGLSpl}[2][]{%
  \new@ifnextchar[{\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2}[]}%
}
```

\@cGLSpl@

```
\def\@cGLSpl@#1#2[#3]{\@GLSpl@{#1}{#2}[#3]}
```

\cGLSplformat Format used by \cGLSpl if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
\newcommand*{\cGLSplformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
}
```

Modify the trigger formats to check for the regular attribute.

\cglsformat

```
\renewcommand*{\cglsformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}#2%
}
```

\cGlsformat

```
\renewcommand*{\cGlsformat}[2]{%
  \glsifregular{#1}
  {\Glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}#2%
}
```

\cglsplformat

```
\renewcommand*{\cglsplformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}#2%
}
```

\cGlsplformat

```
\renewcommand*{\cGlsplformat}[2]{%
  \glsifregular{#1}
  {\Glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}#2%
}
```

New code similar to above for unit counting.

`defunitcounters`

```
\newcommand*{\@@newglossaryentry@defunitcounters}{%
  \edef\@glo@countunit{\csuse{@glsxtr@categoryattr@@\@glo@category @unitcount}}%
  \ifdefvoid\@glo@countunit
  {}%
  {%
    \@glsxtr@ifunitcounter{\@glo@countunit}%
    {}%
    {\expandafter\@glsxtr@addunitcounter\expandafter{\@glo@countunit}}%
  }%
}
```

`r@unitcountlist` List to keep track of which counters are being used by the entry unit count facility.

```
\newcommand*{\@glsxtr@unitcountlist}{}
```

`@addunitcounter`

```
\newcommand*{\@glsxtr@addunitcounter}[1]{%
 \listadd{\@glsxtr@unitcountlist}{#1}%
 \ifcsundef{glsxtr@theunit@#1}
 {%
    \ifcsdef{theH#1}%
   {\csdef{glsxtr@theunit@#1}{\csuse{theH#1}}}%
   {\csdef{glsxtr@theunit@#1}{\csuse{the#1}}}%
 }%
 {}%
}
```

`r@ifunitcounter`

```
\newcommand*{\@glsxtr@ifunitcounter}[3]{%
   \xifinlist{#1}{\@glsxtr@unitcountlist}{#2}{#3}%
}
```

`urrentunitcount`

```
\newcommand*\@glsxtr@currentunitcount[1]{%
 glo@\glsdetoklabel{#1}@currunit@\glsgetattribute{#1}{unitcount}.%
 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
}
```

`eviousunitcount`

```
\newcommand*\@glsxtr@previousunitcount[1]{%
 glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
}
```

`t@currunitcount`

```
\newcommand*{\@gls@increment@currunitcount}[1]{%
  \glshasattribute{#1}{unitcount}%
  {%
```

```
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
    \ifcsundef{\@glsxtr@csname}%
    {%
      \csgdef{\@glsxtr@csname}{1}%
      \listcsxadd
       {glo@\glsdetoklabel{#1}@unitlist}%
       {\glsgetattribute{#1}{unitcount}.%
        \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
       }%
    }%
    {%
      \csxdef{\@glsxtr@csname}%
      {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
    }%
  }%
  {}%
}
```

t@currunitcount

```
\newcommand*{\@gls@local@increment@currunitcount}[1]{%
  \glshasattribute{#1}{unitcount}%
  {%
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
    \ifcsundef{\@glsxtr@csname}%
    {%
      \csdef{\@glsxtr@csname}{1}%
      \listcseadd
       {glo@\glsdetoklabel{#1}@unitlist}%
       {\glsgetattribute{#1}{unitcount}.%
        \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
       }%
    }%
    {%
      \csedef{\@glsxtr@csname}%
      {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
    }%
  }%
  {}%
}
```

r@currunitcount

```
\newcommand*{\@glsxtr@currunitcount}[2]{%
 \ifcsundef
 {glo@\glsdetoklabel{#1}@currunit@#2}%
 {0}%
 {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
}%
```

r@prevunitcount

```
\newcommand*{\@glsxtr@prevunitcount}[2]{%
 \ifcsundef
 {glo@\glsdetoklabel{#1}@prevunit@#2}%
 {0}%
 {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
}%
```

eentryunitcount

```
\newcommand*{\glsenableentryunitcount}{%
```

Enable new fields:

```
  \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defunitcounters}%
```

Just in case the user has switched on the docdef option.

```
  \renewcommand*{\gls@defdocnewglossaryentry}{%
    \renewcommand*\newglossaryentry[2]{%
      \PackageError{glossaries}{\string\newglossaryentry\space
      may only be used in the preamble when entry counting has
      been activated}{If you use \string\glsenableentryunitcount\space
      you must place all entry definitions in the preamble not in
      the document environment}%
    }%
  }%
```

New commands to access new fields:

```
  \newcommand*{\glsentrycurrcount}[1]{%
    \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}.%
     \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
  }%
  \newcommand*{\glsentryprevcount}[1]{%
    \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}.%
     \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
  }%
```

Access total count:

```
  \newcommand*{\glsentryprevtotalcount}[1]{%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
    {0}%
    {%
      \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
    }%
  }%
```

Access max value:

```
  \newcommand*{\glsentryprevmaxcount}[1]{%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
    {0}%
    {%
      \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
    }%
  }%
```

Adjust post unset and reset:

```
\let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
\renewcommand*{\glsxtrpostunset}[1]{%
  \@glsxtr@entryunitcount@org@unset{##1}%
  \@gls@increment@currunitcount{##1}%
}%
\let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
\renewcommand*{\glsxtrpostlocalunset}[1]{%
  \@glsxtr@entryunitcount@org@localunset{##1}%
  \@gls@local@increment@currunitcount{##1}%
}%
\let\@glsxtr@entryunitcount@org@reset\glsxtrpostreset
\renewcommand*{\glsxtrpostreset}[1]{%
  \glshasattribute{##1}{unitcount}%
  {%
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{##1}}%
    \ifcsundef{\@glsxtr@csname}%
    {}%
    {\csgdef{\@glsxtr@csname}{0}}%
  }%
  {}%
}%
\let\@glsxtr@entryunitcount@org@localreset\glsxtrpostlocalreset
\renewcommand*{\glsxtrpostlocalreset}[1]{%
  \@glsxtr@entryunitcount@org@localreset{##1}%
  \glshasattribute{##1}{unitcount}%
  {%
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{##1}}%
    \ifcsundef{\@glsxtr@csname}%
    {}%
    {\csdef{\@glsxtr@csname}{0}}%
  }%
  {}%
}%
```

Modifications to take into account the attributes that govern whether the entry should be unset.

```
\let\@cgls@\@@cgls@
\let\@cglspl@\@@cglspl@
```

```
\let\@cGls@\@@cGls@
\let\@cGlspl@\@@cGlspl@
\let\@cGLS@\@@cGLS@
\let\@cGLSpl@\@@cGLSpl@
```

Write information to the aux file.

```
\AtEndDocument{\@gls@write@entryunitcounts}%
\renewcommand*{\@gls@entry@unitcount}[3]{%
  \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
  \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
```

```
    {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
    {%
      \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
        \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
    }%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
    {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
    {%
      \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
      \fi
    }%
  }%
  \let\glsenableentryunitcount\relax
  \renewcommand*{\glsenableentrycount}{%
    \PackageError{glossaries-extra}{\string\glsenableentrycount\space
     can't be used with \string\glsenableentryunitcount}%
    {Use one or other but not both commands}%
  }%
}
\@onlypreamble\glsenableentryunitcount
```

entry@unitcount

```
\newcommand*{\@gls@entry@unitcount}[3]{}
```

ryunitcounts@do

```
\newcommand*{\@gls@write@entryunitcounts@do}[1]{%
  \immediate\write\@auxout
   {\string\@gls@entry@unitcount
     {\@glsentry}%
     {\@glsxtr@currunitcount{\@glsentry}{#1}%
     }%
     {#1}}%
}
```

entryunitcounts

```
\newcommand*{\@gls@write@entryunitcounts}{%
  \immediate\write\@auxout
    {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
  \count@=0\relax
  \forallglsentries{\@glsentry}{%
    \glshasattribute{\@glsentry}{unitcount}%
    {%
      \ifglsused{\@glsentry}%
      {%
        \forlistcsloop
          {\@gls@write@entryunitcounts@do}%
          {glo@\glsdetoklabel{\@glsentry}@unitlist}%
      }%
      {}%
      \advance\count@ by \@ne
    }%
    {}%
  }%
  \ifnum\count@=0
    \GlossariesExtraWarningNoLine{Entry counting has been enabled
     \MessageBreak with \string\glsenableentryunitcount\space but the
     \MessageBreak attribute ‘unitcount’ hasn’t
     \MessageBreak been assigned to any of the defined
     \MessageBreak entries}%
  \fi
}
```

tryUnitCounting

The first argument is the list of categories, the second argument is the value of the entrycount attribute and the third is the counter name.

```
\newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
```

Enable entry counting:

```
  \glsenableentryunitcount
```

Redefine `\gls` etc:

```
  \renewcommand*{\gls}{\cgls}%
  \renewcommand*{\Gls}{\cGls}%
  \renewcommand*{\glspl}{\cglspl}%
  \renewcommand*{\Glspl}{\cGlspl}%
  \renewcommand*{\GLS}{\cGLS}%
  \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
  \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%
```

In case this command is used again:

```
  \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
  \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
    can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
   {Use one or other but not both commands}}%
}
```

tcountunsetattr

```
\newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
 \@for\@glsxtr@cat:=#1\do
 {%
   \ifdefempty{\@glsxtr@cat}{}%
   {%
     \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
     \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
   }%
 }%
}
```

1.3.6 Acronym Modifications

It's more consistent to use the abbreviation code for acronyms, but make some adjustments to allow for continued use of the glossaries package's custom acronym format. (For example, user may already have defined some acronym styles with `\newacronymstyle` which they would like to continue to use.) The original glossaries acronym code can be restored with `\RestoreAcronyms`, but adjust `\SetGenericNewAcronym` so that `\newacronym` adds the category.

`nericNewAcronym`

```
\renewcommand*{\SetGenericNewAcronym}{%
  \let\@Gls@entryname\@Gls@acrentryname
  \renewcommand{\newacronym}[4][]{%
    \ifdefempty{\@glsacronymlists}%
    {%
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
    }%
    {}%
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \protected@edef\@do@newglossaryentry{%
      \noexpand\newglossaryentry{\the\glslabeltok}%
      {%
        type=\acronymtype,%
        name={\expandonce{\acronymentry{##2}}},%
        sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
        text={\the\glsshorttok},%
        short={\the\glsshorttok},%
        shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
        long={\the\glslongtok},%
        longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
        category=acronym,
        \GenericAcronymFields,%
        \the\glskeylisttok
      }%
    }%
    \@do@newglossaryentry
  }%
  \renewcommand*{\acrfullfmt}[3]{%
    \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
  \renewcommand*{\Acrfullfmt}[3]{%
    \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
  \renewcommand*{\ACRfullfmt}[3]{%
    \glslink[##1]{##2}{%
```

```
      \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
  \renewcommand*{\acrfullplfmt}[3]{%
    \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
  \renewcommand*{\Acrfullplfmt}[3]{%
    \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
  \renewcommand*{\ACRfullplfmt}[3]{%
    \glslink[##1]{##2}{%
      \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%
  \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
  \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
  \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
  \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
}
```

This will cause a problem for glossaries that contain a mixture of acronyms and abbreviations, so redefine `\newacronym` to use the new abbreviation interface.

First save the original definitions:

```
\let\@glsxtr@org@setacronymstyle\setacronymstyle
\let\@glsxtr@org@newacronymstyle\newacronymstyle
```

`msAbbreviations` Make acronyms use the same interface as abbreviations. Note that `\newacronymstyle` has a different implementation to `\newabbrevationstyle` so disable `\newacronymstyle` and `\setacronymstyle`.

```
\newcommand*{\MakeAcronymsAbbreviations}{%
   \renewcommand*{\newacronym}[4][]{%
     \glsxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
   }%
   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
   \renewcommand*{\setacronymstyle}[1]{%
      \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
      unavailable.
      Use \string\setabbreviationstyle\space instead.
      The original acronym interface can be restored with
      \string\RestoreAcronyms}{}%
   }%
   \renewcommand*{\newacronymstyle}[1]{%
      \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
      available unless you restore the original acronym interface with
      \string\RestoreAcronyms}%
      \@glsxtr@org@newacronymstyle{##1}%
   }%
}
```

Switch acronyms to abbreviations:

```
\MakeAcronymsAbbreviations
```

`RestoreAcronyms` Restore acronyms to glossaries interface.

```
\newcommand*{\RestoreAcronyms}{%
  \SetGenericNewAcronym
  \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
  \renewcommand{\acronymfont}[1]{##1}%
  \let\setacronymstyle\@glsxtr@org@setacronymstyle
  \let\newacronymstyle\@glsxtr@org@newacronymstyle
```

Need to restore the original definition of `\@gls@link@checkfirsthyper` but `\glsxtrifwasfirstuse` still needs setting for the benefit of the post-link hook.

```
  \renewcommand*\@gls@link@checkfirsthyper{%
    \ifglsused{\glslabel}%
    {\let\glsxtrifwasfirstuse\@secondoftwo}
    {\let\glsxtrifwasfirstuse\@firstoftwo}%
    \@glsxtr@org@checkfirsthyper
  }
  \glssetcategoryattribute{acronym}{regular}{false}%
  \setacronymstyle{long-short}%
}
```

`\glsacspace` Allow the user to customise the maximum value.

```
\renewcommand*{\glsacspace}[1]{%
  \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
  \ifdim\dimen@<\glsacspacemax~\else\space\fi
}
```

`\glsacspacemax` Value used in the above.

```
\newcommand*{\glsacspacemax}{3em}
```

1.3.7 Indexing and Displaying Glossaries

From time-to-time users ask if they can have one glossary sorted normally and another sorted by definition or usage. With the base glossaries package this can only be achieved with the "noidx" commands (Option 1). This is an attempt to mix and match.

First we need a list of the glossaries that require `makeindex`/`xindy`.

`r@reg@glosslist`

```
\newcommand*{\@glsxtr@reg@glosslist}{}
```

Save the original definition of `\makeglossaries`:

```
\let\@glsxtr@org@makeglossaries\makeglossaries
```

Redefine `\makeglossaries` to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application. The optional argument version shouldn't be used with record.

`\makeglossaries`

```
\renewcommand*{\makeglossaries}[1][]{%
 \@glsxtr@if@record@only
```

```
{%
 \PackageError{glossaries-extra}{\string\makeglossaries\space
  not permitted\MessageBreak with record=\@glsxtr@record@setting\space
  package option}%
 {You may only use \string\makeglossaries\space with
  record=off or record=alsoindex options}%
}%
{%
  \ifblank{#1}%
  {\@glsxtr@org@makeglossaries}%
  {%
    \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
      \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
      not permitted\MessageBreak with record=alsoindex package option}%
      {You may only use the hybrid \string\makeglossaries[...]\space with
       record=off option}%
    \else
      \edef\@glsxtr@reg@glosslist{#1}%
      \ifundef{\glswrite}{\newwrite\glswrite}{}%
      \protected@write\@auxout{}{\string\providecommand
        \string\@glsorder[1]{}}
      \protected@write\@auxout{}{\string\providecommand
        \string\@istfilename[1]{}}
      \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
      \protected@write\@auxout{}{\string\@glsorder{\glsorder}}
      \protected@write\@auxout{}{\string\glsxtr@makeglossaries{#1}}
      \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
```

Iterate through each supplied glossary type and activate it.

```
      \@for\@glo@type:=#1\do{%
       \ifdefempty{\@glo@type}{}{\@makeglossary{\@glo@type}}%
      }%
```

New glossaries must be created before `\makeglossaries`:

```
      \renewcommand*\newglossary[4][]{%
      \PackageError{glossaries}{New glossaries
      must be created before \string\makeglossaries}{You need
      to move \string\makeglossaries\space after all your
      \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
       \let\@makeglossary\relax
       \let\makeglossary\relax
       \renewcommand\makeglossaries[1][]{}%
```

Disable all commands that have no effect after `\makeglossaries`

```
       \@disable@onlypremakeg
```

Allow see key:

```
       \let\gls@checkseeallowed\relax
```

Adjust `\@do@seeglossary`. This needs to check for the entry's existence but don't increment associated counter.

```
\renewcommand*{\@do@seeglossary}[2]{%
  \glsdoifexists{##1}%
  {%
    \edef\@gls@label{\glsdetoklabel{##1}}%
    \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
    \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
    {\@glsxtr@org@doseeglossary{##1}{##2}}%
    {%
      \@@glsxtrwrglossmark
      \protected@write\@auxout{}{%
        \string\@gls@reference
          {\gls@type}{\@gls@label}{\string\glsseeformat##2{}}%
      }%
    }%
  }%
}%
```

Adjust `\@@do@@wrglossary`

```
\let\@glsxtr@@do@@wrglossary\@@do@@wrglossary
\def\@@do@@wrglossary{%
  \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
  \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
  {\@glsxtr@@do@@wrglossary}%
  {\gls@noidxglossary}%
}%
```

Suppress warning about no `\makeglossaries`

```
\let\warn@nomakeglossaries\relax
\def\warn@noprintglossary{%
  \GlossariesWarningNoLine{No \string\printglossary\space
    or \string\printglossaries\space
    found.^^J(Remove \string\makeglossaries\space if you don't want
    any glossaries.)^^JThis document will not have a glossary}%
}%
```

Only warn for glossaries not listed.

```
\renewcommand{\@gls@noref@warn}[1]{%
  \edef\@gls@type{##1}%
  \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
  {%
    \GlossariesExtraWarning{Can't use
       \string\printnoidxglossary[type={\@gls@type}]
       when '\@gls@type' is listed in the optional argument of
       \string\makeglossaries}%
  }%
  {%
    \GlossariesWarning{Empty glossary for
    \string\printnoidxglossary[type={##1}].
```

```
          Rerun may be required (or you may have forgotten to use
          commands like \string\gls)}%
        }%
      }%
```

Adjust display number list to check for type:

```
      \renewcommand*{\glsdisplaynumberlist}[1]{%
        \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
        {\@glsxtr@idx@displaynumberlist{##1}}%
        {\@glsxtr@noidx@displaynumberlist{##1}}%
      }%
```

Adjust entry list:

```
      \renewcommand*{\glsentrynumberlist}[1]{%
        \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
        {\@glsxtr@idx@entrynumberlist{##1}}%
        {\@glsxtr@noidx@entrynumberlist{##1}}%
      }%
```

Adjust number list loop

```
      \renewcommand*{\glsnumberlistloop}[2]{%
        \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
        {%
           \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
            not available for glossary ‘##1’}{}%
        }%
        {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
      }%
```

Only sanitize sort for normal indexing glossaries.

```
      \renewcommand*{\glsprestandardsort}[3]{%
        \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
        {%
          \glsdosanitizesort
        }%
        {%
          \ifglssanitizesort
           \@gls@noidx@sanitizesort
          \else
           \@gls@noidx@nosanitizesort
          \fi
        }%
      }%
```

Unlike `\makenoidxglossaries` we can’t automatically set sanitizesort=false. All entries must be defined in the preamble.

```
      \renewcommand*\new@glossaryentry[2]{%
        \PackageError{glossaries-extra}{Glossary entries must be defined
         in the preamble\MessageBreak when you use the optional argument
         of \string\makeglossaries}{Either move your definitions to the
         preamble or don’t use the optional argument of
```

```
           \string\makeglossaries}%
        }%
```

Only activate sort key for glossaries that aren't listed in #1 (glossary label is stored in `\@glo@type` but this defaults to `\glsdefaulttype` so some expansion is required).

```
        \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
        \renewcommand*{\@printgloss@setsort}{%
```

Need to extract just the type value.

```
          \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%
            type=\glsdefaulttype,\@end@glsxtr@gettype
          \def\@glo@sorttype{\@glo@default@sorttype}%
        }%
```

Check automake setting:

```
        \ifglsautomake
          \renewcommand*{\@gls@doautomake}{%
            \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
              \ifdefempty{\@gls@type}{}{\@gls@automake{\@gls@type}}%
            }%
          }%
        \fi
```

Check the sort setting (glossaries v4.30 onwards):

```
        \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
     \fi
   }%
 }%
}
```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`rgprintglossary` This no longer simply saves `\@printglossary` with `\let` but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes `\provideignoredglossary` to the glstex file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```
\newcommand{\@glsxtr@orgprintglossary}[2]{%
  \def\@glo@type{\glsdefaulttype}%
```

Add check here.

```
  \def\glossarytitle{%
     \ifcsdef{@glotype@\@glo@type @title}%
     {\csuse{@glotype@\@glo@type @title}}%
     {\glossaryname}}%
  \def\glossarytoctitle{\glossarytitle}%
  \let\org@glossarytitle\glossarytitle
  \def\@glossarystyle{%
    \ifx\@glossary@default@style\relax
      \GlossariesWarning{No default glossary style provided \MessageBreak
        for the glossary '\@glo@type'. \MessageBreak
        Using deprecated fallback. \MessageBreak
        To fix this set the style with \MessageBreak
        \string\setglossarystyle\space or use the \MessageBreak
        style key=value option}%
    \fi
  }%
  \def\gls@dotoctitle{\glssettoctitle{\@glo@type}}%
  \let\@org@glossaryentrynumbers\glossaryentrynumbers
  \bgroup
    \@printgloss@setsort
    \setkeys{printgloss}{#1}%
    \ifx\glossarytitle\org@glossarytitle
    \else
      \cslet{@glotype@\@glo@type @title}{\glossarytitle}%
    \fi
    \let\currentglossary\@glo@type
    \let\org@glossaryentrynumbers\glossaryentrynumbers
    \let\glsnonextpages\@glsnonextpages
    \let\glsnextpages\@glsnextpages
```

```
    \glsxtractivatenopost
    \gls@dotoctitle
    \@glossarystyle
    \let\gls@org@glossaryentryfield\glossentry
    \let\gls@org@glossarysubentryfield\subglossentry
    \renewcommand{\glossentry}[1]{%
      \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
      \gls@org@glossaryentryfield{##1}%
    }%
    \renewcommand{\subglossentry}[2]{%
      \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
      \gls@org@glossarysubentryfield{##1}{##2}%
    }%
    \@gls@preglossaryhook
    #2%
  \egroup
  \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
  \global\let\warn@noprintglossary\relax
}
```

`ractivatenopost` Change `\nopostdesc` and `\glsxtrnopostpunc` to behave as they do in the glossary.

```
\newcommand*{\glsxtractivatenopost}{%
  \let\nopostdesc\@nopostdesc
  \let\glsxtrnopostpunc\@glsxtr@nopostpunc
}
```

`lsxtrnopostpunc`

```
\newrobustcmd*{\glsxtrnopostpunc}{}
```

sxtr@nopostpunc Provide a command that works like `\nopostdesc` but only switches of the punctuation without suppressing the post-description hook.

```
\newcommand{\@glsxtr@nopostpunc}{%
 \let\@@glsxtr@org@postdescription\glspostdescription
 \ifglsnopostdot
   \renewcommand{\glspostdescription}{%
     \glsnopostdottrue
     \let\glspostdescription\@@glsxtr@org@postdescription
     \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
     \glsxtrpostdescription
     \@glsxtr@nopostpunc@postdesc}%
 \else
   \renewcommand{\glspostdescription}{%
     \let\glspostdescription\@@glsxtr@org@postdescription
     \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
     \glsxtrpostdescription
     \@glsxtr@nopostpunc@postdesc}%
 \fi
 \glsnopostdotfalse
}
```

stpunc@postdesc

```
\newcommand*{\@glsxtr@nopostpunc@postdesc}{}
```

estore@postpunc

```
\newcommand*{\@glsxtr@restore@postpunc}{%
 \def\@glsxtr@nopostpunc@postdesc{%
   \@glsxtr@org@postdescription
   \let\@glsxtr@nopostpunc@postdesc\@empty
   \let\glsxtrrestorepostpunc\@empty
 }%
}
```

restorepostpunc Does nothing outside of glossary.

```
\newcommand*{\glsxtrrestorepostpunc}{}
```

\@printglossary Redefine.

```
\renewcommand{\@printglossary}[2]{%
  \def\@glsxtr@printglossopts{#1}%
  \@glsxtr@orgprintglossary{#1}{#2}%
}
```

Add a key that switches off the entry targets:

```
\define@choicekey{printgloss}{target}
[\@glsxtr@printglossval\@glsxtr@printglossnr]%
{true,false}[true]%
{%
  \ifcase\@glsxtr@printglossnr
```

```
    \def\@glstarget{\glsdohypertarget}%
  \else
    \let\@glstarget\@secondoftwo
  \fi
}
```

hypernameprefix

```
\newcommand{\@glsxtrhypernameprefix}{}
```

New to v1.20:

```
\define@key{printgloss}{targetnameprefix}{%
  \renewcommand{\@glsxtrhypernameprefix}{#1}%
}
```

```
\define@key{printgloss}{prefix}{%
  \renewcommand{\glolinkprefix}{#1}%
}
```

lsdohypertarget

Redefine to insert `\@glsxtrhypernameprefix` before the target name.

```
\let\@glsxtr@org@glsdohypertarget\glsdohypertarget
\renewcommand{\glsdohypertarget}[2]{%
  \@glsxtr@org@glsdohypertarget{\@glsxtrhypernameprefix#1}{#2}%
}
```

Update `\@glstarget` to use `\def` instead being assigned with `\let` so that it can pick up the new definition and allow any further redefinitions:

```
\ifx\@glstarget\@glsxtr@org@glsdohypertarget
 \def\@glstarget{\glsdohypertarget}%
\fi
```

@makeglossaries

For the benefit of `makeglossaries`

```
\newcommand*{\glsxtr@makeglossaries}[1]{}
```

@glsxtr@gettype

Get just the type.

```
\def\@glsxtr@gettype#1,type=#2,#3\@end@glsxtr@gettype{%
  \def\@glo@type{#2}%
}
```

@assign@sortkey

Assign the sort key.

```
\newcommand\@glsxtr@mixed@assign@sortkey[1]{%
  \edef\@glo@type{\@glo@type}%
  \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glsxtr@reg@glosslist}%
  {%
    \@glo@no@assign@sortkey{#1}%
  }%
  {%
    \@@glo@assign@sortkey{#1}%
  }%
}%
```

Display number list for the regular version:

splaynumberlist

```
\let\@glsxtr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
\newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \def\@gls@noidxloclist@sep{%
      \def\@gls@noidxloclist@sep{%
        \def\@gls@noidxloclist@sep{%
          \glsnumlistsep
        }%
        \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
      }%
    }%
    \def\@gls@noidxloclist@finalsep{}%
    \def\@gls@noidxloclist@prev{}%
    \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
    \@gls@noidxloclist@finalsep
    \@gls@noidxloclist@prev
  }%
  {%

    \glsxtrundeftag
    \glsdoifexists{#1}%
    {%
      \GlossariesWarning{Missing location list for ‘#1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
}%

```

And for the number list loop:

@numberlistloop

```
\newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
  \let\@gls@org@glsseeformat\glsseeformat
  \let\glsnoidxdisplayloc#2\relax
  \let\glsseeformat#3\relax
  \ifdef\@gls@loclist
  {%
    \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
  }%
  {%

    \glsxtrundeftag
    \glsdoifexists{#1}%
    {%
      \GlossariesWarning{Missing location list for ‘##1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
  \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
  \let\glsseeformat\@gls@org@glsseeformat
}%
```

Same for entry number list.

entrynumberlist

```
\newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \glsnoidxloclist{\@gls@loclist}%
  }%
  {%

    \glsxtrundeftag
    \glsdoifexists{#1}%
    {%
      \GlossariesWarning{Missing location list for ‘#1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
}%
```

entrynumberlist

```
\newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}
```

x@getgrouptitle Patch.

```
\renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{#1}%
  \ifdefvoid\@glsxtr@titlelabel
  {}%
  {%
    \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
  }%
  \ifdefvoid{\@glsxtr@titlelabel}%
  {%
    \DTLifint{#1}%
    {%
      \ifnum#1<256\relax
        \edef#2{\char#1\relax}%
      \else
        \edef#2{#1}%
      \fi
    }%
    {%
      \ifcsundef{#1groupname}%
      {\def#2{#1}}%
      {\letcs#2{#1groupname}}%
    }%
  }%
  {%
    \let#2\@glsxtr@titlelabel
  }%
}
```

`g@getgrouptitle` Save original definition of `\@gls@getgrouptitle`

```
\let\glsxtr@org@getgrouptitle\@gls@getgrouptitle
```

`trgetgrouptitle` Provide a user-level command to fetch the group title. The first argument is the group label. The second argument is a control sequence in which to store the title.

```
\newrobustcmd{\glsxtrgetgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \ifcsdef{\@glsxtr@titlelabel}
  {\letcs{#2}{\@glsxtr@titlelabel}}%
  {\glsxtr@org@getgrouptitle{#1}{#2}}%
}
\let\@gls@getgrouptitle\glsxtrgetgrouptitle
```

`trsetgrouptitle` Sets the title for the given group label.

```
\newcommand{\glsxtrsetgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \protected@csxdef{\@glsxtr@titlelabel}{#2}%
}
```

`alsetgrouptitle` As above put only locally defines the title.

```
\newcommand{\glsxtrlocalsetgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \protected@csedef{\@glsxtr@titlelabel}{#2}%
}
```

`\glsnavigation` Redefine to use new user-level command.

```
\renewcommand*{\glsnavigation}{%
  \def\@gls@between{}%
  \ifcsundef{@gls@hypergrouplist@\@glo@type}%
  {%
```

```
    \def\@gls@list{}%
  }%
  {%
    \expandafter\let\expandafter\@gls@list
      \csname @gls@hypergrouplist@\@glo@type\endcsname
  }%
  \@for\@gls@tmp:=\@gls@list\do{%
    \@gls@between
    \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
    \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
    \let\@gls@between\glshypernavsep
  }%
}
```

`@noidx@glossary`

```
\renewcommand*{\@print@noidx@glossary}{%
  \ifcsdef{@glsref@\@glo@type}%
  {%
    \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
    {%
      \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
    }%
    {%
      \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{}%
    }%
    \glossarysection[\glossarytoctitle]{\glossarytitle}%
    \glossarypreamble
```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```
    \def\@gls@currentlettergroup{}%
    \begin{theglossary}%
    \glossaryheader
    \glsresetentrylist
    \forlistcsloop{\@gls@noidx@do}{@glsref@\@glo@type}%
    \end{theglossary}%
    \glossarypostamble
  }%
  {%
```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```
    \glsxtrifemptyglossary{\@glo@type}%
    {}%
    {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
    \@gls@noref@warn{\@glo@type}%
  }%
}
```

`noidxdisplayloc` Patch to check for range formations.

```
\renewcommand*{\glsnoidxdisplayloc}[4]{%
  \setentrycounter[#1]{#2}%
  \@glsxtr@display@loc#3\empty\end@glsxtr@display@loc{#4}%
}
```

xtr@display@loc Patch to check for range formations.

```
\def\@glsxtr@display@loc#1#2\end@glsxtr@display@loc#3{%
  \ifx#1(\relax
    \glsxtrdisplaystartloc{#2}{#3}%
  \else
    \ifx#1)\relax
      \glsxtrdisplayendloc{#2}{#3}%
    \else
      \glsxtrdisplaysingleloc{#1#2}{#3}%
    \fi
  \fi
}
```

isplaysingleloc Single location.

```
\newcommand*{\glsxtrdisplaysingleloc}[2]{%
  \csuse{#1}{#2}%
}
```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glsxtrlocrangefmt`.

displaystartloc Start of a location range.

```
\newcommand*{\glsxtrdisplaystartloc}[2]{%
  \edef\glsxtrlocrangefmt{#1}%
  \ifx\glsxtrlocrangefmt\empty
    \def\glsxtrlocrangefmt{glsnumberformat}%
  \fi
  \expandafter\glsxtrdisplaysingleloc
    \expandafter{\glsxtrlocrangefmt}{#2}%
}
```

trdisplayendloc End of a location range.

```
\newcommand*{\glsxtrdisplayendloc}[2]{%
  \edef\@glsxtr@tmp{#1}%
  \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}%
  \ifx\glsxtrlocrangefmt\@glsxtr@tmp
  \else
    \GlossariesExtraWarning{Mismatched end location range
      (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
  \fi
  \expandafter\glsxtrdisplayendlochook\expandafter{\@glsxtr@tmp}{#2}%
  \expandafter\glsxtrdisplaysingleloc
    \expandafter{\glsxtrlocrangefmt}{#2}%
  \def\glsxtrlocrangefmt{}%
}
```

splayendlochook Allow the user to hook into the end of range command.

```
\newcommand*{\glsxtrdisplayendlochook}[2]{}
```

sxtrlocrangefmt Current range format. Empty if not in a range.

```
\newcommand*{\glsxtrlocrangefmt}{}
```

setentrycounter Adjust `\setentrycounter` to save the original prefix.

```
\renewcommand*{\setentrycounter}[2][]{%
  \def\glsxtrcounterprefix{#1}%
  \ifx\glsxtrcounterprefix\@empty
    \def\@glo@counterprefix{.}%
  \else
    \def\@glo@counterprefix{.#1.}%
  \fi
  \def\glsentrycounter{#2}%
}
```

ls@removespaces Redefine to allow adjustments to location hyperlink.

```
\def\@gls@removespaces#1 #2\@nil{%
 \toks@=\expandafter{\the\toks@#1}%
 \ifx\\#2\\%
   \edef\x{\the\toks@}%
   \ifx\x\empty
   \else
```

Expand location (just in case `\toks@` is needed for something else).

```
    \expandafter\glsxtrlocationhyperlink\expandafter
     \glsentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
   \fi
 \else
   \@gls@ReturnAfterFi{%
     \@gls@removespaces#2\@nil
   }%
 \fi
}
```

ocationhyperlink `\glsxtrlocationhyperlink{⟨counter⟩}{⟨prefix⟩}{⟨location⟩}`

```
\newcommand*{\glsxtrlocationhyperlink}[3]{%
  \ifdefvoid\glsxtrsupplocationurl
  {%
    \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
  }%
  {%
    \hyperref{\glsxtrsupplocationurl}{}{#1#2#3}{#3}%
  }%
}
```

supphypernumber

```
\newcommand*{\glsxtrsupphypernumber}[1]{%
 {%
   \glshasattribute{\glscurrententrylabel}{externallocation}%
   {%
     \def\glsxtrsupplocationurl{%
       \glsgetattribute{\glscurrententrylabel}{externallocation}}%
   }%
   {%
     \def\glsxtrsupplocationurl{}%
   }%
   \glshypernumber{#1}%
 }%
}
```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```
\renewcommand{\@print@glossary}{%
  \makeatletter
  \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
  \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
  {}%
  {\glsxtrNoGlossaryWarning{\@glo@type}}%
  \ifglsxindy
    \ifcsundef{@xdy@\@glo@type @language}%
    {%
      \edef\@do@auxoutstuff{%
        \noexpand\AtEndDocument{%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\providecommand\string\@xdylanguage[2]{}}%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
        }%
      }%
    }%
    {%
      \edef\@do@auxoutstuff{%
        \noexpand\AtEndDocument{%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\providecommand\string\@xdylanguage[2]{}}%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
              @language\endcsname}}%
        }%
      }%
    }%
    \@do@auxoutstuff
    \edef\@do@auxoutstuff{%
      \noexpand\AtEndDocument{%
         \noexpand\immediate\noexpand\write\@auxout{%
          \string\providecommand\string\@gls@codepage[2]{}}%
         \noexpand\immediate\noexpand\write\@auxout{%
          \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
      }%
    }%
    \@do@auxoutstuff
  \fi
  \renewcommand*{\@warn@nomakeglossaries}{%
    \GlossariesWarningNoLine{\string\makeglossaries\space
    hasn't been used,^^Jthe glossaries will not be updated}%
  }%
}
```

Setup the warning text to display if the external file for the given glossary is missing.

`oGlsWarningHead` Header message.

```
\newcommand{\GlsXtrNoGlsWarningHead}[2]{%
 This document is incomplete. The external file associated with
 the glossary '#1' (which should be called \texttt{#2})
 hasn't been created.%
}
```

`rningEmptyStart` No entries have been added to the glossary.

```
\newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
  This has probably happened because there are no entries defined
  in this glossary.%
}
```

`arningEmptyMain` The default "main" glossary is empty.

```
\newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
 If you don't want this glossary,
 add \texttt{nomain} to your package option list when you load
 \texttt{glossaries-extra.sty}. For example:%
}
```

`ingEmptyNotMain` A glossary that isn't the default "main" glossary is empty.

```
\newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
 Did you forget to use \texttt{type=#1} when you defined your
 entries? If you tried to load entries into this glossary with
 \texttt{\string\loadglsentries} did you remember to use
 \texttt{[#1]} as the optional argument? If you did, check that
 the definitions in the file you loaded all had the type set
 to \texttt{\string\glsdefaulttype}.%
}
```

`arningCheckFile` Advisory message to check the file contents.

```
\newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
  Check the contents of the file \texttt{#1}. If
  it's empty, that means you haven't indexed any of your entries in this
  glossary (using commands like \texttt{\string\gls} or
  \texttt{\string\glsadd}) so this list can't be generated.
  If the file isn't empty, the document build process hasn't been
  completed.%
}
```

WarningAutoMake Message when automake option has been used.

```
\newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
  You may need to rerun \LaTeX. If you already have, it may be that
  \TeX's shell escape doesn't allow you to run
  \ifglsxindy xindy\else makeindex\fi. Check the
  transcript file \texttt{\jobname.log}. If the shell escape is
  disabled, try one of the following:

  \begin{itemize}
    \item Run the external (Lua) application:

        \texttt{makeglossaries-lite.lua \string"\jobname\string"}

    \item Run the external (Perl) application:

        \texttt{makeglossaries \string"\jobname\string"}
  \end{itemize}

  Then rerun \LaTeX\ on this document.
  \GlossariesExtraWarning{Rerun required to build the
  glossary '#1' or check TeX's shell escape allows
  you to run \ifglsxindy xindy\else makeindex\fi}%
}
```

WarningMisMatch Mismatching `\makenoidxglossaries`.

```
\newcommand{\GlsXtrNoGlsWarningMisMatch}{%
  You need to either replace \texttt{\string\makenoidxglossaries}
  with \texttt{\string\makeglossaries} or replace
  \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
  \texttt{\string\printnoidxglossary}
  (or \texttt{\string\printnoidxglossaries}) and then rebuild
  this document.%
}
```

arningBuildInfo Build advice.

```
\newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
  Try one of the following:
  \begin{itemize}
    \item Add \texttt{automake} to your package option list when you load
```

```
          \texttt{glossaries-extra.sty}. For example:

          \texttt{\string\usepackage[automake]%
              \glsopenbrace glossaries-extra\glsclosebrace}

    \item Run the external (Lua) application:

          \texttt{makeglossaries-lite.lua \string"\jobname\string"}

    \item Run the external (Perl) application:

          \texttt{makeglossaries \string"\jobname\string"}
  \end{itemize}

  Then rerun \LaTeX\ on this document.%
}
```

`trRecordWarning` Paragraph for record=only.

```
\newcommand{\GlsXtrRecordWarning}[1]{%
 \texttt{\string\printglossary} doesn’t work
 with the \texttt{record=only} package option
 use\par\texttt{\string\printunsrtglossary[type=#1]}\par
 instead (or change the package option).%
}
```

`oGlsWarningTail` Final paragraph.

```
\newcommand{\GlsXtrNoGlsWarningTail}{%
 This message will be removed once the problem has been fixed.%
}
```

`GlsWarningNoOut` No out file created. Build advice.

```
\newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
  The file \texttt{#1} doesn’t exist. This most likely means you haven’t used
  \texttt{\string\makeglossaries} or you have used
  \texttt{\string\nofiles}. If this is just a draft version of the
  document, you can suppress this message using the
  \texttt{nomissingglstext} package option.%
}
```

`glossarywarning`

```
\newcommand*{\@glsxtr@defaultnoglossarywarning}[1]{%
 \glossarysection[\glossarytoctitle]{\glossarytitle}
 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
 \par
 \glsxtrifemptyglossary{#1}%
 {%
    \GlsXtrNoGlsWarningEmptyStart\space
    \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
    \medskip
```

```
   \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
       \glsopenbrace glossaries-extra\glsclosebrace}
   \medskip
   }%
   {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
}%
{%
  \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
  {%
    \GlsXtrNoGlsWarningCheckFile
      {\jobname.\csname @glotype@\@glo@type @out\endcsname}

    \ifglsautomake

     \GlsXtrNoGlsWarningAutoMake{#1}

    \else

       \ifthenelse{\equal{#1}{main}}%
       {%
         \GlsXtrNoGlsWarningEmptyMain\par
         \medskip
         \noindent\texttt{\string\usepackage[nomain]%
           \glsopenbrace glossaries-extra\glsclosebrace}
         \medskip
       }%
       {}%

       \ifdefequal\makeglossaries\@no@makeglossaries
       {%
         \GlsXtrNoGlsWarningMisMatch
       }%
       {%
         \GlsXtrNoGlsWarningBuildInfo
       }%
    \fi
  }%
  {%
    \GlsXtrNoGlsWarningNoOut
      {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
  }%
}%
\par
\GlsXtrNoGlsWarningTail
}
```

glossarywarning Warn about using \printglossary with record

```
\newcommand*{\@glsxtr@record@noglossarywarning}[1]{%
 \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
```

```
  with record=only package option\MessageBreak(use
  \string\printunsrtglossary[type=#1])\MessageBreak
  instead (or change the package option)}%
 \glossarysection[\glossarytoctitle]{\glossarytitle}
 \GlsXtrRecordWarning{#1}
 \GlsXtrNoGlsWarningTail
}
```

Provide some commands to accompany the record option for use with bib2gls.

`xtrresourcefile` Since it's dangerous for an external application to create a file with a .tex extension, as from v1.11 this enforces a .glstex extension to avoid conflict.

```
\newcommand*{\glsxtrresourcefile}[2][]{%
```

The record option can't be set after this command.

```
  \disable@keys{glossaries-extra.sty}{record}%
  \glsxtr@writefields
  \protected@write\@auxout{\glsxtrresourceinit}{\string\glsxtr@resource{#1}{#2}}%
  \let\@glsxtr@org@see@noindex\@gls@see@noindex
  \let\@gls@see@noindex\relax
  \IfFileExists{#2.glstex}%
  {%
```

Can't scope `\@input` so save and restore the category code of `@` to allow for internal commands in the location list.

```
    \edef\@bibgls@restoreat{\noexpand\catcode\noexpand`\noexpand\@=\number\catcode`\@}%
    \makeatletter
    \@input{#2.glstex}%
    \@bibgls@restoreat
```

If the record=nameref option has been set, check if this is supported by the installed version of `bib2gls`.

```
    \@glsxtr@check@bibgls@nameref
  }%
  {%
    \GlossariesExtraWarning{No file `#2.glstex'}%
  }%
  \let\@gls@see@noindex\@glsxtr@org@see@noindex
}
\@onlypreamble\glsxtrresourcefile
```

`@bibgls@nameref` This will only warn after `bib2gls` has created the .glstex file, but there's way to check before.

```
\newcommand{\@glsxtr@check@bibgls@nameref}{%
  \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
    \ifdef\bibglshrefchar
    {}%
    {%
      \GlossariesExtraWarning{record=nameref requires at least
      version 1.8 of bib2gls}%
    }%
  \fi
  \let\@glsxtr@check@bibgls@nameref\relax
}
```

xtrresourceinit Code used during the protected write operation.

```
\newcommand*{\glsxtrresourceinit}{}
```

trresourcecount

```
\newcount\glsxtrresourcecount
```

trLoadResources Short cut that uses `\glsxtrresourcefile` with `\jobname` as the mandatory argument.

```
\newcommand*{\GlsXtrLoadResources}[1][]{%
  \ifnum\glsxtrresourcecount=0\relax
    \glsxtrresourcefile[#1]{\jobname}%
  \else
    \glsxtrresourcefile[#1]{\jobname-\the\glsxtrresourcecount}%
  \fi
  \advance\glsxtrresourcecount by 1\relax
}
```

glsxtr@resource

```
\newcommand*{\glsxtr@resource}[2]{}
```

\glsxtr@fields

```
\newcommand*{\glsxtr@fields}[1]{}
```

xtr@texencoding

```
\newcommand*{\glsxtr@texencoding}[1]{}
```

\glsxtr@langtag

```
\newcommand*{\glsxtr@langtag}[1]{}
```

@pluralsuffixes

```
\newcommand*{\glsxtr@pluralsuffixes}[4]{}
```

tr@shortcutsval

```
\newcommand*{\glsxtr@shortcutsval}[1]{}
```

sxtr@linkprefix

```
\newcommand*{\glsxtr@linkprefix}[1]{}
```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
\newcommand*{\glsxtr@writefields}{%
```

```
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@fields}[1]{}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@resource}[2]{}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@pluralsuffixes}[4]{}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@shortcutsval}[1]{}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@linkprefix}[1]{}}%
\protected@write\@auxout{}{\string\glsxtr@fields{\@gls@keymap}}%
```

```
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@record}[5]{}}%
```

```
\ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
  \protected@write\@auxout{}%
   {\string\providecommand*{\string\glsxtr@record@nameref}[8]{}}%
\fi
```

If any languages have been loaded, the language tag will be available in `\CurrentTrackedLanguageTag` (provided by tracklang). For multilingual documents, the required locale will have to be indicated in the `sort` key when using `\glsxtrresourcefile`.

```
\ifdef\CurrentTrackedLanguageTag
{%
   \protected@write\@auxout{}{%
     \string\glsxtr@langtag{\CurrentTrackedLanguageTag}}%
}%
{}%
\protected@write\@auxout{}{\string\glsxtr@pluralsuffixes
  {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
  {\glsxtrabbrvpluralsuffix}}%
\ifdef\inputencodingname
{%
   \protected@write\@auxout{}{\string\glsxtr@texencoding{\inputencodingname}}%
}%
{%
```

If fontspec has been loaded, assume UTF-8. (The encoding can be changed with `\XeTeXinputencoding`, but I can't work out how to determine the current encoding.)

```
   \@ifpackageloaded{fontspec}%
   {\protected@write\@auxout{}{\string\glsxtr@texencoding{utf8}}}%
   {}%
}%
\protected@write\@auxout{}{\string\glsxtr@shortcutsval{\@glsxtr@shortcutsval}}%
```

Prefix deferred until the beginning of the document in case it's redefined later in the preamble. This is picked up by bib2gls when the external option is used.

```
\AtBeginDocument
  {\protected@write\@auxout{}{\string\glsxtr@linkprefix{\glolinkprefix}}}%
\let\glsxtr@writefields\relax
```

If the automake option is on, try running bib2gls if the aux file exists. The double-quotes around `\jobname` have been removed (v1.19) since `\jobname` will include double-quotes if the file name has spaces.

```
  \ifglsautomake
    \IfFileExists{\jobname.aux}%
    {\immediate\write18{bib2gls \jobname}}{}%
```

If `\makeglossaries` is also used, allow makeindex/xindy to also be run, otherwise disable the error message about requiring `\makeglossaries` with automake=true.

```
    \ifx\@gls@doautomake\@gls@doautomake@err
      \let\@gls@doautomake\relax
    \fi
  \fi
}
```

`do@automake@err`

```
\newcommand*{\@gls@doautomake@err}{%
  \PackageError{glossaries}{You must use
  \string\makeglossaries\space with automake=true}
  {%
    Either remove the automake=true setting or
    add \string\makeglossaries\space to your document preamble.%
  }%
}
```

Allow locations specific to a particular counter to be recorded.

`\glsxtr@record`

```
\newcommand*{\glsxtr@record}[5]{}
```

`@record@nameref` Used with record=nameref to include current label information.

```
\newcommand*{\glsxtr@record@nameref}[8]{}
```

`r@counterrecord` Aux file command.

```
\newcommand*{\glsxtr@counterrecord}[3]{%
  \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
}
```

`unterrecordhook` Hook used by `\@glsxtr@dorecord`.

```
\newcommand*{\@glsxtr@counterrecordhook}{}
```

`trRecordCounter` Activate recording for a particular counter (identified in the argument).

```
\newcommand*{\GlsXtrRecordCounter}[1]{%
  \@@glsxtr@recordcounter{#1}%
}
\@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
\newcommand*{\@glsxtr@docounterrecord}[1]{%
  \protected@write\@auxout{}{\string\glsxtr@counterrecord
    {\@gls@label}{#1}{\csuse{the#1}}}%
}
```

lsxtrglossentry

Users may prefer to have entries displayed throughout the document rather than gathered together in a list. This command emulates the way `\glossentry` behaves (without the style formatting commands like `\item`). This needs to define `\currentglossary` to the current glossary type (normally set at the start of `\@printglossary`) and needs to define `\glscurrententrylabel` to the entry's label (normally set before `\glossentry` and `\subglossentry`). This needs some protection in case it's used in a section heading.

```
\newcommand*{\glsxtrglossentry}[1]{%
  \glsxtrtitleorpdforheading
  {\@glsxtrglossentry{#1}}%
  {\glsentryname{#1}}%
  {\glsxtrheadname{#1}}%
}
```

lsxtrglossentry

Another test is needed in case `\@glsxtrglossentry` has been written to the table of contents.

```
\newrobustcmd*{\@glsxtrglossentry}[1]{%
  \glsxtrtitleorpdforheading
  {%
    \glsdoifexists{#1}%
    {%
      \begingroup
        \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
        \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
        \ifglshasparent{#1}%
        {\GlsXtrStandaloneSubEntryItem{#1}}%
        {\glsentryitem{#1}}%
        \GlsXtrStandaloneEntryName{#1}%
      \endgroup
    }%
  }%
  {\glsentryname{#1}}%
  {\glsxtrheadname{#1}}%
}
```

daloneEntryName

```
\newcommand*{\GlsXtrStandaloneEntryName}[1]{%
  \glstarget{#1}{\glossentryname{#1}}%
}
```

oneGlossaryType

To make it easier to adjust the definition of `\currentglossary` within `\glsxtrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```
\newcommand{\GlsXtrStandaloneGlossaryType}{\glsentrytype{\glscurrententrylabel}}
```

oneSubEntryItem Used for sub-entries in standalone format. The argument is the entry's label.

```
\newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
  \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}%
}
```

glossentryother As `\glsxtrglossentry` but uses a different field. First argument is code to use in the header. The second argument is the entry's label. The third argument is the internal field label. This needs to be expandable in case it occurs in a sectioning command so it can't have an optional argument.

```
\newcommand*{\glsxtrglossentryother}[3]{%
  \ifstrempty{#1}%
  {%
    \ifcsdef{glsxtrhead#3}%
    {%
      \glsxtrtitleorpdforheading
      {\@glsxtrglossentryother{#2}{#3}{#1}}%
      {\@gls@entry@field{#2}{#3}}%
      {\csuse{glsxtrhead#3}{#2}}%
    }%
    {%
      \glsxtrtitleorpdforheading
      {\@glsxtrglossentryother{#2}{#3}{#1}}%
      {\@gls@entry@field{#2}{#3}}%
      {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
    }%
  }%
  {%
    \glsxtrtitleorpdforheading
    {\@glsxtrglossentryother{#2}{#3}{#1}}%
    {\@gls@entry@field{#2}{#3}}%
    {#1}%
  }%
}
```

glossentryother As `\@glsxtrglossentry` but uses a different field.

```
\newrobustcmd*{\@glsxtrglossentryother}[3]{%
  \glsxtrtitleorpdforheading
  {%
    \glsdoifexists{#1}%
    {%
      \begingroup
        \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
        \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
        \ifglshasparent{#1}%
        {\GlsXtrStandaloneSubEntryItem{#1}}%
        {\glsentryitem{#1}}%
        \GlsXtrStandaloneEntryOther{#1}%
      \endgroup
    }%
  }%
  {\@gls@entry@field{#1}{#2}}%
  {#3}%
}
```

aloneEntryOther

```
\newcommand*{\GlsXtrStandaloneEntryOther}[2]{%
  \glstarget{#1}{\glossentrynameother{#1}{#2}}%
}
```

ntunsrtglossary

Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

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

ntunsrtglossary

Unstarred version.

```
\newcommand*{\@printunsrtglossary}[1][]{%
  \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
}
```

ntunsrtglossary

Starred version.

```
\newcommand*{\s@printunsrtglossary}[2][]{%
  \begingroup
    #2%
    \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
  \endgroup
}
```

unsrtglossaries

Similar to `\printnoidxglossaries` but it displays all entries defined for the given glossary without sorting.

```
\newcommand*{\printunsrtglossaries}{%
  \forallglossaries{\@@glo@type}{\printunsrtglossary[type=\@@glo@type]}%
}
```

@unsrt@glossary

```
\newcommand*{\@print@unsrt@glossary}{%
  \glossarysection[\glossarytoctitle]{\glossarytitle}%
  \glossarypreamble
```

check for empty list

```
  \glsxtrifemptyglossary{\@glo@type}%
  {%
    \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
  }%
  {%
```

```
    \key@ifundefined{glossentry}{group}%
    {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
    {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
    \def\@gls@currentlettergroup{}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
    \def\@glsxtr@doglossary{%
      \begin{theglossary}%
      \glossaryheader
      \glsresetentrylist
    }%
    \expandafter\@for\expandafter\glscurrententrylabel\expandafter
      :\expandafter=\csname glolist@\@glo@type\endcsname\do{%
      \ifdefempty{\glscurrententrylabel}
      {}%
      {%
```

Provide a hook (for example to measure width).

```
        \let\glsxtr@process\@firstofone
        \let\printunsrtglossaryskipentry
            \@glsxtr@printunsrtglossaryskipentry
        \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%
```

Don't check group for child entries.

```
        \glsxtr@process
        {%
          \ifglshasparent{\glscurrententrylabel}{}%
          {%
            \@glsxtr@checkgroup\glscurrententrylabel
            \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
              {\@glsxtr@groupheading}%
          }%
          \eappto\@glsxtr@doglossary{%
            \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
        }%
      }%
    }%
    \appto\@glsxtr@doglossary{\end{theglossary}}%
    \printunsrtglossarypredoglossary
    \@glsxtr@doglossary
  }%
  \glossarypostamble
}
```

ntryprocesshook

```
\newcommand*{\printunsrtglossaryentryprocesshook}[1]{}
```

ossaryskipentry

```
\newcommand*{\printunsrtglossaryskipentry}{%
  \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
can only be used within \string\printunsrtglossaryentryprocesshook}{}%
}
```

ntryprocesshook

```
\newcommand*{\@glsxtr@printunsrtglossaryskipentry}{%
  \let\glsxtr@process\@gobble
}
```

rypredoglossary

```
\newcommand*{\printunsrtglossarypredoglossary}{}
```

lossary@handler

```
\newcommand{\@printunsrt@glossary@handler}[1]{%
  \xdef\glscurrententrylabel{#1}%
  \printunsrtglossaryhandler\glscurrententrylabel
}
```

glossaryhandler

```
\newcommand{\printunsrtglossaryhandler}[1]{%
  \glsxtrunsrtdo{#1}%
}
```

xtriflabelinlist

\glsxtriflabelinlist{⟨*label*⟩}{⟨*list*⟩}{⟨*true*⟩}{⟨*false*⟩}

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@gls@ifinlist` which ensures the label and list are fully expanded.

```
\newrobustcmd*{\glsxtriflabelinlist}[4]{%
 \protected@edef\@glsxtr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
 \@glsxtr@doiflabelinlist{#3}{#4}%
}
```

srtglossaryunit

```
\newcommand{\print@op@unsrtglossaryunit}[2][]{%
  \s@printunsrtglossary[type=\glsdefaulttype,#1]{%
    \printunsrtglossaryunitsetup{#2}%
  }%
}
```

ossaryunitsetup

```
\newcommand*{\printunsrtglossaryunitsetup}[1]{%
  \renewcommand{\printunsrtglossaryhandler}[1]{%
    \glsxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
    {\glsxtrunsrtdo{##1}}%
    {}%
  }%
```

Only the target names should have the prefixes adjusted as `\gls` etc need the original `\glolinkprefix`. The `\@gobble` part discards `\glolinkprefix`.

```
  \ifcsundef{theH#1}%
  {%
    \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
  }%
  {%
    \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
  }%
  \renewcommand*{\glossarysection}[2][]{}%
  \appto\glossarypostamble{\glspar\medskip\glspar}%
}
```

`srtglossaryunit`

```
\newcommand{\print@noop@unsrtglossaryunit}[2][]{%
  \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
   requires the record=only or record=alsoindex package option}{}%
}
```

`t@getgrouptitle`

```
\newrobustcmd*{\@glsxtr@unsrt@getgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \ifcsdef{\@glsxtr@titlelabel}
  {\letcs{#2}{\@glsxtr@titlelabel}}%
  {\def#2{#1}}%
}
```

`\glsxtrunsrtdo` Provide a user-level call to `\@glsxtr@noidx@do` to make it easier to define a new handler.

```
\newcommand{\glsxtrunsrtdo}{\@glsxtr@noidx@do}
```

`lsxtrgroupfield` bib2gls provides a supplementary field labelled `secondarygroup` for secondary glossaries, so provide a way of switching to that field. (The group key still needs checking. There's no associated key with the internal field).

```
\newcommand*{\glsxtrgroupfield}{group}
```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while `\@glsxtr@doglossary` is being constructed rather than in the handler.

`sxtr@checkgroup` The argument is the entry's label. (This block of code was formerly in `\@glsxtr@noidx@do`.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in `\@glsxtr@groupheading`, which will be empty if no heading is required.

```
\newcommand*{\@glsxtr@checkgroup}[1]{%
  \def\@glsxtr@groupheading{}%
  \key@ifundefined{glossentry}{group}%
  {%
```

```
    \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
    \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
  }%
  {%
    \protected@edef\@glo@thislettergrp{%
        \csuse{glo@\glsdetoklabel{#1}@\glsxtrgroupfield}}%
  }%
  \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
  {}%
  {%
    \ifdefempty{\@gls@currentlettergroup}{}%
    {\def\@glsxtr@groupheading{\glsgroupskip}}%
    \eappto\@glsxtr@groupheading{%
      \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
    }%
  }%
  \let\@gls@currentlettergroup\@glo@thislettergrp
}
```

`trLocationField` Stores the internal name of the location field.

```
\newcommand*{\GlsXtrLocationField}{location}
```

`glsxtr@noidx@do` Minor modification of `\@gls@noidx@do` to check for location field if present, but also need to check for the group field.

```
\newcommand{\@glsxtr@noidx@do}[1]{%
  \ifglsentryexists{#1}%
  {%
    \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
    \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@\GlsXtrLocationField}%
    \ifglshasparent{#1}%
    {%
      \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
      \ifdefvoid{\@gls@location}%
      {%
        \ifdefvoid{\@gls@loclist}%
        {%
          \subglossentry{\gls@level}{#1}{}%
        }%
        {%
          \subglossentry{\gls@level}{#1}%
          {%
            \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
          }%
        }%
      }%
      {%
        \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
      }%
    }%
    {%
      \ifdefvoid{\@gls@location}%
      {%
        \ifdefvoid{\@gls@loclist}
        {%
          \glossentry{#1}{}%
        }%
        {%
          \glossentry{#1}%
          {%
            \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
          }%
        }%
      }%
      {%
        \glossentry{#1}%
        {%
          \glossaryentrynumbers{\@gls@location}%
        }%
      }%
    }%
  }%
  {}%
}
```

Provide a way to conveniently define commands that behaves like `\gls` with a label prefix. It's possible that the user might want minor variations with the same prefix but different default options, so use a counter to provide unique inner commands.

`\glsxtrnewgls`

```
\newcount\@glsxtrnewgls@inner
```

(The default options supplied in ⟨*options*⟩ below could possibly be used to form the inner control sequence name to help make it unique, but it might feasibly contain thevalue where the value might contain commands.)

`r@providenewgls`

```
\newcommand*{\@glsxtr@providenewgls}{%
  \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@newglslike}[2]{}}%
  \let\@glsxtr@providenewgls\relax
}
```

`identifyglslike` Identify the command given in the second argument for the benefit of `bib2gls`.

```
\newcommand{\glsxtridentifyglslike}[2]{%
 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
 {}%
 {%
```

```
   \@glsxtr@providenewgls
   \protected@write\@auxout{}{\string\@glsxtr@newglslike{#1}{\string#2}}%
 }%
}
```

\@glsxtrnewgls

`\glsxtrnewgls[⟨options⟩]{⟨prefix⟩}{⟨cs⟩}{⟨inner cs name⟩}`

```
\newcommand*{\@glsxtrnewgls}[4]{%
  \ifdef{#3}%
  {%
    \PackageError{glossaries-extra}{Command \string#3\space already
defined}{}%
  }%
  {%
```

Write information to the aux file for bib2gls.

```
     \glsxtridentifyglslike{#2}{#3}%
     \ifcsdef{@#4like@#2}%
     {%
       \advance\@glsxtrnewgls@inner by \@ne
       \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
     }%
     {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
     \expandafter\newrobustcmd\expandafter*\expandafter
      #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
     \ifstrempty{#1}%
     {%
       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
         \new@ifnextchar[%
          {\csname @#4@\endcsname{##1}{#2##2}}%
          {\csname @#4@\endcsname{##1}{#2##2}[]}%
       }%
     }%
     {%
       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
         \new@ifnextchar[%
          {\csname @#4@\endcsname{#1,##1}{#2##2}}%
          {\csname @#4@\endcsname{#1,##1}{#2##2}[]}%
       }%
     }%
  }%
}
```

\glsxtrnewgls

`\glsxtrnewgls[⟨options⟩]{⟨prefix⟩}{⟨cs⟩}`

The first argument prepends to the options and the second argument is the prefix.

```
\newrobustcmd*{\glsxtrnewgls}[3][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{gls}%
}
```

`lsxtrnewglslike` Provide a way to conveniently define commands that behave like `\gls`, `\glspl`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
\newrobustcmd*{\glsxtrnewglslike}[6][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{gls}%
  \@glsxtrnewgls{#1}{#2}{#4}{glspl}%
  \@glsxtrnewgls{#1}{#2}{#5}{Gls}%
  \@glsxtrnewgls{#1}{#2}{#6}{Glspl}%
}
```

`lsxtrnewGLSlike` Provide a way to conveniently define commands that behave like `\GLS`, `\GLSpl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
\newrobustcmd*{\glsxtrnewGLSlike}[4][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{GLS}%
  \@glsxtrnewgls{#1}{#2}{#4}{GLSpl}%
}
```

`\glsxtrnewrgls` As `\glsxtrnewgls` but for `\rgls`.

```
\newrobustcmd*{\glsxtrnewrgls}[3][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
}
```

`sxtrnewrglslike` As `\glsxtrnewglslike` but for `\rgls` etc.

```
\newrobustcmd*{\glsxtrnewrglslike}[6][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
  \@glsxtrnewgls{#1}{#2}{#4}{rglspl}%
  \@glsxtrnewgls{#1}{#2}{#5}{rGls}%
  \@glsxtrnewgls{#1}{#2}{#6}{rGlspl}%
}
```

`sxtrnewrGLSlike` As `\glsxtrnewGLSlike` but for `\rGLS` etc.

```
\newrobustcmd*{\glsxtrnewrGLSlike}[4][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{rGLS}%
  \@glsxtrnewgls{#1}{#2}{#4}{rGLSpl}%
}
```

Provide easy access to record count fields.

`otalRecordCount` Access total record count. This is designed to be expandable. The argument is the label.

```
\newcommand*{\GlsXtrTotalRecordCount}[1]{%
 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount}%
 {\csname glo@\glsdetoklabel{#1}@recordcount\endcsname}%
 {0}%
}
```

`sXtrRecordCount`

Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
\newcommand*{\GlsXtrRecordCount}[2]{%
 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2}%
 {\csname glo@\glsdetoklabel{#1}@recordcount.#2\endcsname}%
 {0}%
}
```

`tionRecordCount`

Access record count for a particular counter and location. The first argument is the label. The second argument is the counter name. The third argument is the location. This command shouldn't be used if the location doesn't fully expand unless `\glsxtrdetoklocation` can be set to something sensible.

```
\newcommand*{\GlsXtrLocationRecordCount}[3]{%
 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2.\glsxtrdetoklocation{#3}}%
 {\csname glo@\glsdetoklabel{#1}@recordcount.#2.\glsxtrdetoklocation{#3}\endcsname}%
 {0}%
}
```

`trdetoklocation`

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

`ablerecordcount`

```
\newcommand*{\glsxtrenablerecordcount}{%
  \renewcommand*{\gls}{\rgls}%
  \renewcommand*{\Gls}{\rGls}%
  \renewcommand*{\glspl}{\rglspl}%
  \renewcommand*{\Glspl}{\rGlspl}%
  \renewcommand*{\GLS}{\rGLS}%
  \renewcommand*{\GLSpl}{\rGLSpl}%
}
```

`ordtriggervalue`

The value used by the record trigger test. The argument is the entry's label.

```
\newcommand*{\glsxtrrecordtriggervalue}[1]{%
 \GlsXtrTotalRecordCount{#1}%
}
```

`dCountAttribute`

```
\newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
 \@for\@glsxtr@cat:=#1\do
 {%
    \ifdefempty{\@glsxtr@cat}{}%
   {%
      \glssetcategoryattribute{\@glsxtr@cat}{recordcount}{#2}%
   }%
 }%
}
```

rifrecordtrigger

`\glsxtrifrecordtrigger{⟨label⟩}{⟨trigger format⟩}{⟨normal⟩}`

```
\newcommand*{\glsxtrifrecordtrigger}[3]{%
 \glshasattribute{#1}{recordcount}%
 {%
   \ifnum\glsxtrrecordtriggervalue{#1}>\glsgetattribute{#1}{recordcount}\relax
    #3%
   \else
    #2%
   \fi
 }%
 {#3}%
}
```

strigger@record

Still need a record to ensure that bib2gls selects the entry.

```
\newcommand*{\@glsxtr@rglstrigger@record}[3]{%
  \edef\glslabel{\glsdetoklabel{#2}}%
  \let\@gls@link@label\glslabel
  \def\@glsxtr@thevalue{}%
  \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
  \def\@glsnumberformat{glstriggerrecordformat}%
  \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
  \edef\glstype{\csname glo@\glslabel @type\endcsname}%
  \def\@glsxtr@thevalue{}%
  \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
  \glsxtrinitwrgloss
  \glslinkpresetkeys
  \setkeys{glslink}{#1}%
  \glslinkpostsetkeys
  \ifdefempty{\@glsxtr@thevalue}%
  {%
    \@gls@saveentrycounter
  }%
  {%
    \let\theglsentrycounter\@glsxtr@thevalue
    \def\theHglsentrycounter{\@glsxtr@theHvalue}%
  }%
  \ifglsxtrinitwrglossbefore
    \@do@wrglossary{#2}%
  \fi
  #3%
  \ifglsxtrinitwrglossbefore
  \else
    \@do@wrglossary{#2}%
  \fi
  \ifKV@glslink@local
    \glslocalunset{#2}%
  \else
    \glsunset{#2}%
  \fi
}
```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by `bib2gls`.

```
\newcommand*{\glstriggerrecordformat}[1]{}
```

\rgls

```
\newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}
```

\@rgls

```
\newcommand*{\@rgls}[2][]{%
  \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
}
```

\@rgls@

```
\def\@rgls@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
  }%
  {%
    \@gls@{#1}{#2}[#3]%
  }%
}%
```

\rglspl

```
\newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}
```

\@rglspl

```
\newcommand*{\@rglspl}[2][]{%
  \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
}
```

\@rglspl@

```
\def\@rglspl@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
  }%
  {%
    \@glspl@{#1}{#2}[#3]%
  }%
}%
```

\rGls

```
\newrobustcmd*{\rGls}{\@gls@hyp@opt\@rGls}
```

\@rGls

```
\newcommand*{\@rGls}[2][]{%
  \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
}
```

\@rGls@

```
\def\@rGls@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
  }%
  {%
    \@Gls@{#1}{#2}[#3]%
  }%
}%
```

\rGlspl

```
\newrobustcmd*{\rGlspl}{\@gls@hyp@opt\@rGlspl}
```

\@rGlspl

```
\newcommand*{\@rGlspl}[2][]{%
  \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
}
```

\@rGlspl@

```
\def\@rGlspl@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
  }%
  {%
    \@Glspl@{#1}{#2}[#3]%
  }%
}%
```

\rGLS

```
\newrobustcmd*{\rGLS}{\@gls@hyp@opt\@rGLS}
```

\@rGLS

```
\newcommand*{\@rGLS}[2][]{%
  \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}[]}%
}
```

\@rGLS@

```
\def\@rGLS@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
  }%
  {%
    \@GLS@{#1}{#2}[#3]%
  }%
}%
```

\rGLSpl

```
\newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\@rGLSpl}
```

\@rGLSpl

```
\newcommand*{\@rGLSpl}[2][]{%
  \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2}[]}%
}
```

\@rGLSpl@

```
\def\@rGLSpl@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
  }%
  {%
    \@GLSpl@{#1}{#2}[#3]%
  }%
}%
```

\rglsformat

```
\newcommand*{\rglsformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}#2%
}
```

\rglsplformat

```
\newcommand*{\rglsplformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}#2%
}
```

\rGlsformat

```
\newcommand*{\rGlsformat}[2]{%
  \glsifregular{#1}
  {\Glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}#2%
}
```

\rGlsplformat

```
\newcommand*{\rGlsplformat}[2]{%
```

```
 \glsifregular{#1}
 {\Glsentryfirstplural{#1}}%
 {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}#2%
}
```

\rGLSformat

```
\newcommand*{\rGLSformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\rglsformat{#1}{#2}}%
}
```

\rGLSplformat

```
\newcommand*{\rGLSplformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\rglsplformat{#1}{#2}}%
}
```

1.4 Link Counting

This is different to the entry counting provided by the base package (which counts the number of times the first use flag is unset). Instead, this method hooks into `\@gls@link` (through `\glsxtr@inc@linkcount`) to increment an associated counter. To preserve resources, the counter is only defined if it needs to be incremented. This method is independent of the presence of hyperlinks. (The "link" part of the name refers to `\@gls@link` not `\hyperlink`.)

o@inc@linkcount

This performs the actual incrementing and counter definition. The counter is given by `\c@glsxtr@linkcount@`⟨*label*⟩ where ⟨*label*⟩ is the entry's label. Since this is performed within `\@gls@link` the label can be accessed with `\glslabel`.

```
\newcommand{\@glsxtr@do@inc@linkcount}{%
```

Does this entry have the linkcount attribute set?

```
 \glsifattribute{\glslabel}{linkcount}{true}%
 {%
```

Does the counter exist?

```
   \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
   {%
```

Counter doesn't exist, so define it.

```
     \newcounter{glsxtr@linkcount@\glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
     \glshasattribute{\glslabel}{linkcountmaster}%
     {%
```

Need to ensure values are fully expanded.

```
       \begingroup
        \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
          {\glsgetattribute{\glslabel}{linkcountmaster}}}%
       \x
     }%
     {}%
   }%
```

Increment counter:

```
  \glsxtrinclinkcounter{glsxtr@linkcount@\glslabel}%
 }%
 {}%
}
```

`rinclinkcounter`

May be redefined to use `\refstepcounter` if required.

```
\newcommand*{\glsxtrinclinkcounter}[1]{\stepcounter{#1}}
```

`inkCounterValue`

Expands to the associated link counter register or 0 if not defined.

```
\newcommand*{\GlsXtrLinkCounterValue}[1]{%
 \ifcsundef{c@glsxtr@linkcount@#1}{0}{\csname c@glsxtr@linkcount@#1\endcsname}%
}
```

`rTheLinkCounter`

Expands to the display value of the associated link counter or 0 if not defined.

```
\newcommand*{\GlsXtrTheLinkCounter}[1]{%
 \ifcsundef{theglsxtr@linkcount@#1}{0}%
 {\csname theglsxtr@linkcount@#1\endcsname}%
}
```

`fLinkCounterDef`

Tests if the counter has been defined

```
\newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
 \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
}
```

`LinkCounterName`

Expands to the associated link counter name. (No check for existence.)

```
\newcommand*{\GlsXtrLinkCounterName}[1]{glsxtr@linkcount@#1}
```

`ableLinkCounting`

`\GlsXtrEnableLinkCounting[⟨master counter⟩]{⟨categories⟩}`

Enable link counting for the given categories.

```
\newcommand*{\GlsXtrEnableLinkCounting}[2][]{%
 \let\glsxtr@inc@linkcount\@glsxtr@do@inc@linkcount
 \@for\@glsxtr@label:=#2\do
 {%
  \glssetcategoryattribute{\@glsxtr@label}{linkcount}{true}%
  \ifstrempty{#1}{}%
  {%
    \ifcsundef{c@#1}%
    {\@nocounterr{#1}}%
    {\glssetcategoryattribute{\@glsxtr@label}{linkcountmaster}{#1}}%
  }%
 }%
}
\@onlypreamble\GlsXtrEnableLinkCounting
```

1.5 Integration with glossaries-accsupp

Provide better integration with the glossaries-accsupp package. (Must be loaded before the main code of glossaries-extra either explicitly or through the accsupp package option.)

These commands have their definitions set according to whether or not glossaries-extra has been loaded.

```
\@ifpackageloaded{glossaries-accsupp}
{
```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```
  \newcommand*{\glsaccessname}[1]{%
    \glsnameaccessdisplay
    {%
      \glsentryname{#1}%
    }%
    {#1}%
  }
```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccessname}[1]{%
    \glsnameaccessdisplay
    {%
      \Glsentryname{#1}%
    }%
    {#1}%
  }
```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessname}[1]{%
    \glsnameaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentryname{#1}}%
    }%
    {#1}%
  }
```

`\glsaccesstext` Display the text value (no link and no check for existence).

```
  \newcommand*{\glsaccesstext}[1]{%
    \glstextaccessdisplay
    {%
      \glsentrytext{#1}%
    }%
    {#1}%
  }
```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccesstext}[1]{%
  \glstextaccessdisplay
  {%
    \Glsentrytext{#1}%
  }%
  {#1}%
}
```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccesstext}[1]{%
  \glstextaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrytext{#1}}%
  }%
  {#1}%
}
```

`glsaccessplural` Display the plural value (no link and no check for existence).

```
\newcommand*{\glsaccessplural}[1]{%
  \glspluralaccessdisplay
  {%
    \glsentryplural{#1}%
  }%
  {#1}%
}
```

`Glsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessplural}[1]{%
  \glspluralaccessdisplay
  {%
    \Glsentryplural{#1}%
  }%
  {#1}%
}
```

`GLSaccessplural` Display the plural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessplural}[1]{%
  \glspluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentryplural{#1}}%
  }%
  {#1}%
}
```

`\glsaccessfirst` Display the first value (no link and no check for existence).

```
\newcommand*{\glsaccessfirst}[1]{%
  \glsfirstaccessdisplay
  {%
    \glsentryfirst{#1}%
  }%
  {#1}%
}
```

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessfirst}[1]{%
  \glsfirstaccessdisplay
  {%
    \Glsentryfirst{#1}%
  }%
  {#1}%
}
```

`\GLSaccessfirst` Display the first value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessfirst}[1]{%
  \glsfirstaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentryfirst{#1}}%
  }%
  {#1}%
}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```
\newcommand*{\glsaccessfirstplural}[1]{%
  \glsfirstpluralaccessdisplay
  {%
    \glsentryfirstplural{#1}%
  }%
  {#1}%
}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessfirstplural}[1]{%
  \glsfirstpluralaccessdisplay
  {%
    \Glsentryfirstplural{#1}%
  }%
  {#1}%
}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessfirstplural}[1]{%
```

```
  \glsfirstpluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
  }%
  {#1}%
}
```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```
\newcommand*{\glsaccesssymbol}[1]{%
  \glssymbolaccessdisplay
  {%
    \glsentrysymbol{#1}%
  }%
  {#1}%
}
```

Glsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccesssymbol}[1]{%
  \glssymbolaccessdisplay
  {%
    \Glsentrysymbol{#1}%
  }%
  {#1}%
}
```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccesssymbol}[1]{%
  \glssymbolaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
  }%
  {#1}%
}
```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```
\newcommand*{\glsaccesssymbolplural}[1]{%
  \glssymbolpluralaccessdisplay
  {%
    \glsentrysymbolplural{#1}%
  }%
  {#1}%
}
```

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccesssymbolplural}[1]{%
  \glssymbolpluralaccessdisplay
  {%
    \Glsentrysymbolplural{#1}%
  }%
  {#1}%
}
```

esssymbolplural Display the symbolplural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccesssymbolplural}[1]{%
  \glssymbolpluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
  }%
  {#1}%
}
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
\newcommand*{\glsaccessdesc}[1]{%
  \glsdescriptionaccessdisplay
  {%
    \glsentrydesc{#1}%
  }%
  {#1}%
}
```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessdesc}[1]{%
  \glsdescriptionaccessdisplay
  {%
    \Glsentrydesc{#1}%
  }%
  {#1}%
}
```

\GLSaccessdesc Display the desc value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessdesc}[1]{%
  \glsdescriptionaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrydesc{#1}}%
  }%
  {#1}%
}
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
\newcommand*{\glsaccessdescplural}[1]{%
  \glsdescriptionpluralaccessdisplay
  {%
    \glsentrydescplural{#1}%
  }%
  {#1}%
}
```

ccessdescplural Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessdescplural}[1]{%
  \glsdescriptionpluralaccessdisplay
  {%
    \Glsentrydescplural{#1}%
  }%
  {#1}%
}
```

ccessdescplural Display the descplural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessdescplural}[1]{%
  \glsdescriptionpluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
  }%
  {#1}%
}
```

\glsaccessshort Display the short form (no link and no check for existence).

```
\newcommand*{\glsaccessshort}[1]{%
  \glsshortaccessdisplay
  {%
    \glsentryshort{#1}%
  }%
  {#1}%
}
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
\newcommand*{\Glsaccessshort}[1]{%
  \glsshortaccessdisplay
  {%
    \Glsentryshort{#1}%
  }%
  {#1}%
}
```

\GLSaccessshort Display the short value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessshort}[1]{%
  \glsshortaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentryshort{#1}}%
  }%
  {#1}%
}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```
\newcommand*{\glsaccessshortpl}[1]{%
  \glsshortpluralaccessdisplay
  {%
    \glsentryshortpl{#1}%
  }%
  {#1}%
}
```

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
\newcommand*{\Glsaccessshortpl}[1]{%
  \glsshortpluralaccessdisplay
  {%
    \Glsentryshortpl{#1}%
  }%
  {#1}%
}
```

LSaccessshortpl Display the shortplural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessshortpl}[1]{%
  \glsshortpluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentryshortpl{#1}}%
  }%
  {#1}%
}
```

\glsaccesslong Display the long form (no link and no check for existence).

```
\newcommand*{\glsaccesslong}[1]{%
  \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
}
```

\Glsaccesslong Display the long form (no link and no check for existence).

```

\newcommand*{\Glsaccesslong}[1]{%
  \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
}
```

\GLSaccesslong Display the long value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccesslong}[1]{%
  \glslongaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrylong{#1}}%
  }%
  {#1}%
}
```

glsaccesslongpl Display the long plural form (no link and no check for existence).

```
\newcommand*{\glsaccesslongpl}[1]{%
  \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
}
```

Glsaccesslongpl Display the long plural form (no link and no check for existence).

```

\newcommand*{\Glsaccesslongpl}[1]{%
  \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
}
```

GLSaccesslongpl Display the longplural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccesslongpl}[1]{%
  \glslongpluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
  }%
  {#1}%
}
```

Keys for accessibility support.

```
\define@key{glsxtrabbrv}{access}{%
  \def\@gls@nameaccess{#1}%
}
\define@key{glsxtrabbrv}{textaccess}{%
  \def\@gls@textaccess{#1}%
}
\define@key{glsxtrabbrv}{firstaccess}{%
  \def\@gls@firstaccess{#1}%
}
\define@key{glsxtrabbrv}{shortaccess}{%
  \def\@gls@shortaccess{#1}%
}
\define@key{glsxtrabbrv}{shortpluralaccess}{%
  \def\@gls@shortaccesspl{#1}%
}
```

@initaccesskeys

```
\newcommand*{\@gls@initaccesskeys}{%
  \def\@gls@nameaccess{}%
  \def\@gls@textaccess{}%
  \def\@gls@firstaccess{}%
  \def\@gls@shortaccess{}%
  \def\@gls@shortaccesspl{}%
}
```

essattribute@set

`\gls@ifaccessattribute@set{⟨attribute⟩}{⟨true⟩}{⟨false⟩}`

```
\newcommand*{\@gls@ifaccessattribute@set}[3]{%
  \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
  {#2}%
  {%
    \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
    {#3}%
    {%
      \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
      {#2}%
      {#3}%
    }%
  }%
}
```

lt@short@access

Assign the default value of the shortaccess key. The argument is the short value passed to `\newabbreviation`.

```
  \newcommand{\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```
    \ifdefempty\@gls@shortaccess
    {%
      \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
      {%
        \@glsxtr@insertdots\@gls@shortaccess{#1}%
        \eappto\ExtraCustomAbbreviationFields{%
          shortaccess={\expandonce\@gls@shortaccess},}%
      }%
      {}%
    }%
    {}%
```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```
    \ifdefempty\@gls@shortaccess
    {}%
    {%
      \ifdefempty\@gls@shortaccesspl
      {%
        \@gls@ifaccessattribute@set{aposplural}%
        {%
          \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
            \@gls@shortaccess'\abbrvpluralsuffix}%
        }%
        {%
          \@gls@ifaccessattribute@set{noshortplural}%
          {%
```

```
         \let\@gls@shortaccesspl\@gls@shortaccess
      }%
      {%
        \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
          \@gls@shortaccess\abbrvpluralsuffix}%
      }%
    }%
    \eappto\ExtraCustomAbbreviationFields{%
      shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
  }%
  {}%
}%
```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```
\ifdefempty\@gls@nameaccess
{%
  \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
  {%
```

Do nothing if the shortaccess key hasn't been set.

```
    \ifdefempty\@gls@shortaccess
    {}%
    {%
       \eappto\ExtraCustomAbbreviationFields{%
          access={\expandonce\@gls@shortaccess},%
       }%
    }%
  }%
  {}%
}%
{}%
```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```
\ifdefempty\@gls@textaccess
{%
  \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
  {%
```

Do nothing if the shortaccess key hasn't been set.

```
    \ifdefempty\@gls@shortaccess
    {}%
    {%
       \eappto\ExtraCustomAbbreviationFields{%
          textaccess={\expandonce\@gls@shortaccess},%
       }%
    }%
  }%
  {}%
}%
{}%
```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
    \ifdefempty\@gls@firstaccess
    {%
      \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
      {%
```

Do nothing if the shortaccess key hasn't been set.

```
        \ifdefempty\@gls@shortaccess
        {}%
        {%
           \eappto\ExtraCustomAbbreviationFields{%
              firstaccess={\expandonce\@gls@shortaccess},%
           }%
        }%
      }%
      {}%
    }%
    {}%
  }
```

End of if accsupp part

```
}
{
```

No accessibility support. Just define these commands to do \glsentry⟨*xxx*⟩

\glsaccessname Display the name value (no link and no check for existence).

```
  \newcommand*{\glsaccessname}[1]{\glsentryname{#1}}
```

\Glsaccessname Display the name value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}
```

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.

```
  \newcommand*{\GLSaccessname}[1]{%
   \protect\mfirstucMakeUppercase{\glsentryname{#1}}}
```

\glsaccesstext Display the text value (no link and no check for existence).

```
  \newcommand*{\glsaccesstext}[1]{\glsentrytext{#1}}
```

\Glsaccesstext Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}
```

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.

```
  \newcommand*{\GLSaccesstext}[1]{%
   \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}
```

glsaccessplural Display the plural value (no link and no check for existence).

```
  \newcommand*{\glsaccessplural}[1]{\glsentryplural{#1}}
```

`Glsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}
```

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessplural}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryplural{#1}}}
```

`\glsaccessfirst` Display the first value (no link and no check for existence).

```
\newcommand*{\glsaccessfirst}[1]{\glsentryfirst{#1}}
```

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}
```

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessfirst}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```
\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessfirstplural}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}
```

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

```
\newcommand*{\glsaccesssymbol}[1]{\glsentrysymbol{#1}}
```

`Glsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}
```

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccesssymbol}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}
```

`esssymbolplural` Display the symbolplural value (no link and no check for existence).

```
\newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}
```

`esssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}
```

esssymbolplural Display the symbolplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccesssymbolplural}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
\newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}
```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}
```

\GLSaccessdesc Display the desc value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessdesc}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessdescplural}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrydescplural{#1}}}
```

\glsaccessshort Display the short form (no link and no check for existence).

```
\newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
\newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessshort}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryshort{#1}}}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```
\newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}
```

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
\newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

`LSaccessshortpl` Display the shortplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessshortpl}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}
```

`\glsaccesslong` Display the long form (no link and no check for existence).

```
\newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}
```

`\Glsaccesslong` Display the long form (no link and no check for existence).

```
\newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

`\GLSaccesslong` Display the long value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccesslong}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrylong{#1}}}
```

`glsaccesslongpl` Display the long plural form (no link and no check for existence).

```
\newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}
```

`Glsaccesslongpl` Display the long plural form (no link and no check for existence).

```
\newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

`GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccesslongpl}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrylongpl{#1}}}
```

`@initaccesskeys` This does nothing if there's no accessibility support.

```
\newcommand*{\@gls@initaccesskeys}{}
```

`lt@short@access` This does nothing if there's no accessibility support.

```
\newcommand{\@gls@setup@default@short@access}[1]{}%
```

End of else part

```
}
```

1.6 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```
\glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```
\newcommand{\glsifcategory}[4]{%
\ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
}
```

Categories can have attributes.

`ategoryattribute`

```
\glssetcategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨value⟩}
```

Set (or override if already set) an attribute for the given category.

```
\newcommand*{\glssetcategoryattribute}[3]{%
  \csdef{@glsxtr@categoryattr@@#1@#2}{#3}%
}
```

`ategoryattribute`

```
\glsgetcategoryattribute{⟨category⟩}{⟨attribute-label⟩}
```

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
\newcommand*{\glsgetcategoryattribute}[2]{%
  \csuse{@glsxtr@categoryattr@@#1@#2}%
}
```

`ategoryattribute`

```
\glshascategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨true⟩}{⟨false⟩}
```

Tests if the category has the given attribute set.

```
\newcommand*{\glshascategoryattribute}[4]{%
  \ifcsvoid{@glsxtr@categoryattr@@#1@#2}{#4}{#3}%
}
```

`\glssetattribute`

```
\glssetattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨value⟩}
```

Short cut where the category label is obtained from the entry information.

```
\newcommand*{\glssetattribute}[3]{%
  \glssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
}
```

`\glsgetattribute`

```
\glsgetattribute{⟨entry label⟩}{⟨attribute-label⟩}
```

Short cut where the category label is obtained from the entry information.

```
\newcommand*{\glsgetattribute}[2]{%
  \glsgetcategoryattribute{\glscategory{#1}}{#2}%
}
```

`\glshasattribute`

```
\glshasattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨true⟩}{⟨false⟩}
```

Short cut to test if the given attribute has been set where the category label is obtained from the entry information.

```
\newcommand*{\glshasattribute}[4]{%
  \ifglsentryexists{#1}%
  {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
  {#4}%
}
```

`ategoryattribute`

```
\glsifcategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨value⟩}{⟨true
part⟩}{⟨false part⟩}
```

True if category has the attribute with the given value.

```
\newcommand{\glsifcategoryattribute}[5]{%
 \ifcsundef{@glsxtr@categoryattr@@#1@#2}%
 {#5}%
 {\ifcsstring{@glsxtr@categoryattr@@#1@#2}{#3}{#4}{#5}}%
}
```

`\glsifattribute`

```
\glsifattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨value⟩}{⟨true part⟩}
{⟨false part⟩}
```

Short cut to determine if the given entry has a category with the given attribute set.

```
\newcommand{\glsifattribute}[5]{%
  \ifglsentryexists{#1}%
  {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
  {#5}%
}
```

Set attributes for the default general category:

```
\glssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
\glssetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```
\newcommand*{\glssetregularcategory}[1]{%
 \glssetcategoryattribute{#1}{regular}{true}%
}
```

fregularcategory

```
\glsifregularcategory{⟨category⟩}{⟨true part⟩}{⟨false part⟩}
```

Short cut to determine if a category has the regular attribute explicitly set to true.

```
\newcommand{\glsifregularcategory}[3]{%
  \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
}
```

tregularcategory

```
\glsifnotregularcategory{⟨category⟩}{⟨true part⟩}{⟨false part⟩}
```

Short cut to determine if a category has the regular attribute explicitly set to false.

```
\newcommand{\glsifnotregularcategory}[3]{%
  \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
}
```

\glsifregular

```
\glsifregular{⟨entry label⟩}{⟨true part⟩}{⟨false part⟩}
```

Short cut to determine if an entry has a regular attribute set to true.

```
\newcommand{\glsifregular}[3]{%
  \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
}
```

\glsifnotregular

```
\glsifnotregular{⟨entry label⟩}{⟨true part⟩}{⟨false part⟩}
```

Short cut to determine if an entry has a regular attribute set to false.

```
\newcommand{\glsifnotregular}[3]{%
  \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
}
```

oreachincategory

```
\glsforeachincategory[⟨glossary labels⟩]{⟨category-label⟩}
{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

Iterates through all entries in all the glossaries (or just those listed in ⟨*glossary labels*⟩) and does ⟨*body*⟩ if the category matches ⟨*category-label*⟩. The control sequences ⟨*glossary-cs*⟩ and ⟨*label-cs*⟩ may be used in ⟨*body*⟩ to access the glossary label and entry label for the current iteration.

```
\newcommand{\glsforeachincategory}[5][\@glo@types]{%
```

```
  \forallglossaries[#1]{#3}%
  {%
     \forglsentries[#3]{#4}%
     {%
       \glsifcategory{#4}{#2}{#5}{}%
     }%
  }%
}
```

achwithattribute

```
\glsforeachwithattribute[⟨glossary labels⟩]{⟨attribute-label⟩}
{⟨attribute-value⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

Iterates through all entries in all the glossaries (or just those listed in ⟨*glossary labels*⟩) and does ⟨*body*⟩ if the category attribute ⟨*attribute-label*⟩ matches ⟨*attribute-value*⟩. The control sequences ⟨*glossary-cs*⟩ and ⟨*label-cs*⟩ may be used in ⟨*body*⟩ to access the glossary label and entry label for the current iteration.

```
\newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
  \forallglossaries[#1]{#4}%
  {%
     \forglsentries[#4]{#5}%
     {%
       \glsifattribute{#5}{#2}{#3}{#6}{}%
     }%
  }%
}
```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to index and add `\glsxtrpostdescription`.

```
\ifdef\newterm
{%
```

\newterm

```
  \renewcommand*{\newterm}[2][]{%
    \newglossaryentry{#2}%
    {type={index},category=index,name={#2},%
     description={\glsxtrpostdescription\nopostdesc},#1}%
  }
```

Indexed terms are regular by default.

```
  \glssetcategoryattribute{index}{regular}{true}
```

trpostdescindex

```
  \newcommand*{\glsxtrpostdescindex}{}
```

```
}
{}
```

If the symbols package option was used, define a similar command for symbols, but set the default sort to the label rather than the name as the symbols will typically contain commands that will confuse makeindex and xindy.

```
\ifdef\printsymbols
{%
```

`glsxtrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```
  \newcommand*{\glsxtrnewsymbol}[3][]{%
    \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
  }
```

Symbols are regular by default.

```
  \glssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
  \newcommand*{\glsxtrpostdescsymbol}{}
```

```
}
{}
```

Similar for the numbers option.

```
\ifdef\printnumbers
{%
```

`glsxtrnewnumber`

```
\ifdef\printnumbers
  \newcommand*{\glsxtrnewnumber}[3][]{%
    \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
  }
```

Numbers are regular by default.

```
  \glssetcategoryattribute{number}{regular}{true}
```

`rpostdescnumber`

```
  \newcommand*{\glsxtrpostdescnumber}{}
```

```
}
{}
```

`sxtrsetcategory` Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
\newcommand*{\glsxtrsetcategory}[2]{%
  \@for\@glsxtr@label:=#1\do
  {%
    \glsfieldxdef{\@glsxtr@label}{category}{#2}%
  }%
}
```

`tcategoryforall` Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
\newcommand*{\glsxtrsetcategoryforall}[2]{%
  \forallglossaries[#1]{\@glsxtr@type}{%
    \forglsentries[\@glsxtr@type]{\@glsxtr@label}%
    {%
      \glsfieldxdef{\@glsxtr@label}{category}{#2}%
    }%
  }%
}
```

`trfieldtitlecase`

```
\glsxtrfieldtitlecase{⟨label⟩}{⟨field⟩}
```

Apply title casing to the contents of the given field.

```
\newcommand*{\glsxtrfieldtitlecase}[2]{%
  \expandafter\glsxtrfieldtitlecasecs\expandafter
    {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
}
```

`ieldtitlecasecs` The command used by `\glsxtrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```
\newcommand*{\glsxtrfieldtitlecasecs}[1]{\xcapitalisewords{#1}}
```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

`\glossentrydesc` If the glossdesc attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\glossentrydesc}[1]{%
    \glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossdescfont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossdescfont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          ‘\@glsxtr@attrval’ supplied in glossdescfont attribute
          for entry `#1'. Ignoring}%
          \let\@glsxtr@glossdescfont\@firstofone
        }%
      }%
      {\let\@glsxtr@glossdescfont\@firstofone}%
      \glsifattribute{#1}{glossdesc}{firstuc}%
      {%
        \@glsxtr@glossdescfont{\Glsaccessdesc{#1}}%
      }%
      {%
        \glsifattribute{#1}{glossdesc}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \glsdescriptionaccessdisplay
          {%
            \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
          }%
          {#1}%
        }%
        {%
          \@glsxtr@glossdescfont{\glsaccessdesc{#1}}%
        }%
      }%
    }%
  }
}
{
  \renewcommand*{\glossentrydesc}[1]{%
    \glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glshasattribute{#1}{glossdescfont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossdescfont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          `\@glsxtr@attrval' supplied in glossdescfont attribute
          for entry `#1'. Ignoring}%
          \let\@glsxtr@glossdescfont\@firstofone
        }%
      }%
      {\let\@glsxtr@glossdescfont\@firstofone}%
      \glsifattribute{#1}{glossdesc}{firstuc}%
      {%
        \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
      }%
      {%
        \glsifattribute{#1}{glossdesc}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
        }%
        {%
          \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
        }%
      }%
    }%
  }
}
```

\glossentryname If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossnamefont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
          for entry ‘#1’. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \glsifattribute{#1}{glossname}{firstuc}%
      {%
        \glsnameaccessdisplay
        {%
          \@glsxtr@glossnamefont{\Glsentryname{#1}}%
        }%
        {#1}%
      }%
      {%
        \glsifattribute{#1}{glossname}{title}%
```

```
    {%
     \@glsxtr@do@titlecaps@warn
     \glsnameaccessdisplay
     {%
      \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
     }%
     {#1}%
    }%
    {%
     \glsifattribute{#1}{glossname}{uc}%
     {%
      \glsnameaccessdisplay
      {%
```

Hide the label from the upper-casing command.

```
        \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
        \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
      }%
      {#1}%
     }%
     {%
      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
      \glsnameaccessdisplay
      {%
       \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
      }%
      {#1}%
     }%
    }%
   }%
```

Do post-name hook:

```
   \glsxtrpostnamehook{#1}%
  }%
 }
}
{
 \renewcommand*{\glossentryname}[1]{%
  \@glsdoifexistsorwarn{#1}%
  {%
   \glssetabbrvfmt{\glscategory{#1}}%
   \glshasattribute{#1}{glossnamefont}%
   {%
    \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
    \ifcsdef{\@glsxtr@attrval}%
    {%
     \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
    }%
    {%
     \GlossariesExtraWarning{Unknown control sequence name
```

```
          `\@glsxtr@attrval' supplied in glossnamefont attribute
          for entry `#1'. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \glsifattribute{#1}{glossname}{firstuc}%
      {%
        \@glsxtr@glossnamefont{\Glsentryname{#1}}%
      }%
      {%
        \glsifattribute{#1}{glossname}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
        }%
        {%
          \glsifattribute{#1}{glossname}{uc}%
          {%
```

Hide the label from the upper-casing command.

```
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
          }%
          {%
```

This little trick is used by glossaries to allow the user to redefine `\glsnamefont` to use `\makefirstuc`. Support it even though they can now use the firstuc attribute.

```
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
          }%
        }%
      }%
```

Do post-name hook.

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\Glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossnamefont}%
      {%
```

```
    \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
    \ifcsdef{\@glsxtr@attrval}%
    {%
     \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
    }%
    {%
     \GlossariesExtraWarning{Unknown control sequence name
     ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
     for entry ‘#1’. Reverting to default \string\glsnamefont}%
     \let\@glsxtr@glossnamefont\glsnamefont
    }%
   }%
   {\let\@glsxtr@glossnamefont\glsnamefont}%
   \glsnameaccessdisplay
   {%
    \@glsxtr@glossnamefont{\Glsentryname{#1}}%
   }%
   {#1}%
```

Do post-name hook:

```
   \glsxtrpostnamehook{#1}%
  }%
 }
}
{
 \renewcommand*{\Glossentryname}[1]{%
  \@glsdoifexistsorwarn{#1}%
  {%
   \glssetabbrvfmt{\glscategory{#1}}%
   \glshasattribute{#1}{glossnamefont}%
   {%
    \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
    \ifcsdef{\@glsxtr@attrval}%
    {%
     \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
    }%
    {%
     \GlossariesExtraWarning{Unknown control sequence name
     ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
     for entry ‘#1’. Reverting to default \string\glsnamefont}%
     \let\@glsxtr@glossnamefont\glsnamefont
    }%
   }%
   {\let\@glsxtr@glossnamefont\glsnamefont}%
   \@glsxtr@glossnamefont{\Glsentryname{#1}}%
```

Do post-name hook:

```
   \glsxtrpostnamehook{#1}%
  }%
 }
```

```
}
```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`xtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```
\newcommand*{\glsxtrpostnamehook}[1]{%
  \let\@glsnumberformat\@glsxtr@defaultnumberformat
  \glsxtrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
  \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
  \csuse{glsxtrpostname\glscategory{#1}}%
}
```

`trapostnamehook`

```
\newcommand*{\glsextrapostnamehook}[1]{}%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
\newcommand*{\glsdefpostname}[2]{%
  \csdef{glsxtrpostname#1}{#2}%
}
```

`etaccessdisplay`

```
\@ifpackageloaded{glossaries-accsupp}
{
  \newcommand*{\glsxtr@setaccessdisplay}[1]{%
     \ifcsdef{gls#1accessdisplay}%
     {\letcs\@glsxtr@accessdisplay{gls#1accessdisplay}}%
     {%
```

This is essentially the reverse of `\@gls@fetchfield`, since the field supplied to `\glossentryname` has to be the internal label, but the `\gls`⟨*field*⟩`accessdisplay` commands use the key name.

```
       \edef\@gls@thisval{#1}%
       \@for\@gls@map:=\@gls@keymap\do{%
        \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
        \ifdefequal{\@this@key}{\@gls@thisval}%
        {%
          \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
          \@endfortrue
        }%
        {}%
       }%
       \ifcsdef{gls\@gls@thisval accessdisplay}%
       {\letcs\@glsxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
       {\let\@glsxtr@accessdisplay\@firstoftwo}%
     }%
  }
}
{%
  \newcommand*{\glsxtr@setaccessdisplay}[1]{%
   \let\@glsxtr@accessdisplay\@firstoftwo}
}
```

sentrynameother Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```
\newrobustcmd*{\glossentrynameother}[2]{%
  \@glsdoifexistsorwarn{#1}%
  {%
```

Accessibility support:

```
    \glsxtr@setaccessdisplay{#2}%
```

Set the abbreviation format:

```
    \glssetabbrvfmt{\glscategory{#1}}%
    \glshasattribute{#1}{glossnamefont}%
    {%
      \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
      \ifcsdef{\@glsxtr@attrval}%
      {%
        \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
      }%
      {%
        \GlossariesExtraWarning{Unknown control sequence name
        `\@glsxtr@attrval' supplied in glossnamefont attribute
        for entry `#1'. Reverting to default \string\glsnamefont}%
        \let\@glsxtr@glossnamefont\glsnamefont
      }%
    }%
    {\let\@glsxtr@glossnamefont\glsnamefont}%
    \glsifattribute{#1}{glossname}{firstuc}%
    {%
      \@glsxtr@accessdisplay
      {\@glsxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
      {#1}%
    }%
    {%
      \glsifattribute{#1}{glossname}{title}%
      {%
        \@glsxtr@do@titlecaps@warn
        \@glsxtr@accessdisplay
        {\@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{#2}}}%
        {#1}%
      }%
      {%
        \glsifattribute{#1}{glossname}{uc}%
        {%
         \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
         \@glsxtr@accessdisplay
         {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
         {#1}%
        }%
        {%
         \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
         \@glsxtr@accessdisplay
         {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
         {#1}%
        }%
      }%
    }%
```

Do post-name hook.

```
   \glsxtrpostnamehook{#1}%
 }%
}
```

format@override Determines if the format key should override the indexing attribute value.

```
\newif\if@glsxtr@format@override
\@glsxtr@format@overridefalse
```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

xFormatOverride

```
\@ifpackageloaded{hyperref}
{
```

If hyperref's hyperindex option is on, then hyperref will automatically add `\hyperpage`, so don't add it.

```
 \ifHy@hyperindex
  \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
   \@glsxtr@format@overridetrue
   \appto\theindex{\let\glshypernumber\@firstofone}%
  }
 \else
  \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
   \@glsxtr@format@overridetrue
   \appto\theindex{\let\glshypernumber\hyperpage}%
  }
 \fi
}
{
 \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
  \@glsxtr@format@overridetrue
 }
```

```
}
\@onlypreamble\GlsXtrEnableIndexFormatOverride
```

doautoindexname

```
\newcommand*{\glsxtrdoautoindexname}[2]{%
  \glshasattribute{#1}{#2}%
  {%
```

Escape any makeindex/xindy characters in the value of the name field. Take care with babel as this won't work if the category code has changed for those characters.

```
    \@glsxtr@autoindex@setname{#1}%
```

If the attribute value is simply "true" don't add an encap, otherwise use the value as the encap.

```
    \protected@edef\@glsxtr@attrval{\glsgetattribute{#1}{#2}}%
    \if@glsxtr@format@override

      \ifx\@glsnumberformat\@glsxtr@defaultnumberformat
      \else
        \let\@glsxtr@attrval\@glsnumberformat
      \fi
    \fi
    \ifdefstring{\@glsxtr@attrval}{true}%
    {}%
    {\eappto\@glo@name{\@glsxtr@autoindex@encap\@glsxtr@attrval}}%
    \expandafter\glsxtrautoindex\expandafter{\@glo@name}%
  }%
  {}%
}
```

glsxtrautoindex

```
\newcommand*{\glsxtrautoindex}{\index}
```

xtrautoindexesc

```
\newcommand{\glsxtrautoindexesc}{%
  \@gls@checkmkidxchars\@glo@sort
  \@glsxtr@autoindex@doextra@esc\@glo@sort
}
```

toindex@setname Assign `\@glo@name` for use with indexname attribute.

```
\newcommand*{\@glsxtr@autoindex@setname}[1]{%
  \protected@edef\@glo@name{\glsxtrautoindexentry{#1}}%
  \glsxtrautoindexassignsort{\@glo@sort}{#1}%
  \glsxtrautoindexesc
  \epreto\@glo@name{\@glo@sort\@glsxtr@autoindex@at}%
}
```

rautoindexentry Command used for the actual part when auto-indexing.

```
\newcommand*{\glsxtrautoindexentry}[1]{\string\glsentryname{#1}}
```

indexassignsort Used to assign the sort value when auto-indexing.

```
\newcommand*{\glsxtrautoindexassignsort}[2]{%
  \glsletentryfield{#1}{#2}{sort}%
}
```

dex@doextra@esc

```
\newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@esc\@gls@quotechar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
        \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape actual character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@at\@gls@actualchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
        \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape level character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@level\@gls@levelchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
        \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape encap character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@encap\@gls@encapchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
        \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
}
```

The user commands here have a preamble-only restriction to ensure they are set before required and also to reduce the chances of complications caused by babel's shorthands.

`tr@autoindex@at` Actual character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@at}{}
```

`trSetActualChar` Set the actual character.

```
\newcommand*{\GlsXtrSetActualChar}[1]{%
  \gdef\@glsxtr@autoindex@at{#1}%
  \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
  }%
}
\@onlypreamble\GlsXtrSetActualChar
\makeatother
\GlsXtrSetActualChar{@}
\makeatletter
```

`autoindex@encap` Encap character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@encap}{}
```

`XtrSetEncapChar` Set the encap character.

```
\newcommand*{\GlsXtrSetEncapChar}[1]{%
  \gdef\@glsxtr@autoindex@encap{#1}%
  \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetEncapChar{|}
\@onlypreamble\GlsXtrSetEncapChar
```

`autoindex@level` Level character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@level}{}
```

`XtrSetLevelChar` Set the encap character.

```
\newcommand*{\GlsXtrSetLevelChar}[1]{%
  \gdef\@glsxtr@autoindex@level{#1}%
  \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetLevelChar{!}
\@onlypreamble\GlsXtrSetLevelChar
```

`r@autoindex@esc` Escape character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@esc}{"}
```

`lsXtrSetEscChar` Set the escape character.

```
\newcommand*{\GlsXtrSetEscChar}[1]{%
  \gdef\@glsxtr@autoindex@esc{#1}%
  \def\@glsxtr@autoindex@escquote##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetEscChar{"}
\@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character `\actualchar`:

```
\ifdef\actualchar
 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
 {}
```

Quote character `\quotechar`:

```
\ifdef\quotechar
 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
 {}
```

Level character `\levelchar`:

```
\ifdef\levelchar
 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
 {}
```

Encap character `\encapchar`:

```
\ifdef\encapchar
 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
 {}
```

`leto@endescspch`

```
\def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

`toindex@esc@spch`

```
\@@glsxtr@autoindex@escspch{⟨char⟩}{⟨cs⟩}{⟨pre⟩}{⟨mid⟩}{⟨post⟩}
```

```
\newcommand*{\@@glsxtr@autoindex@escspch}[5]{%
  \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
  \toks@={#3}%
  \ifx\@nnil#3\relax
    \def\@@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
  \else
    \ifx\@nnil#4\relax
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
     \def\@@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
```

```
        #4#5\@glsxtr@endescspch}%
    \else
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
       \@glsxtr@autoindex@esc#1}%
      \def\@@glsxtr@checkspch{#2#5#1\@nnil#1\@glsxtr@endescspch}%
    \fi
  \fi
  \@@glsxtr@checkspch
}
```

`\Glossentrydesc` Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\Glossentrydesc}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \Glsaccessdesc{#1}%
  }%
}
```

`lossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \glsaccesssymbol{#1}%
  }%
}
```

`lossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\Glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \Glsaccesssymbol{#1}%
  }%
}
```

Allow initials to be marked but only use the formatting for the tag in the glossary.

`eInitialTagging` Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```
\newcommand*{\GlsXtrEnableInitialTagging}{%
  \@ifstar\s@glsxtr@enabletagging\@glsxtr@enabletagging
}
\@onlypreamble\GlsXtrEnableInitialTagging
```

`r@enabletagging` Starred version undefines command.

```
\newcommand*{\s@glsxtr@enabletagging}[2]{%
  \undef#2%
  \@glsxtr@enabletagging{#1}{#2}%
}
```

r@enabletagging Internal command.

```
\newcommand*{\@glsxtr@enabletagging}[2]{%
```

Set attributes for categories given in the first argument.

```
  \@for\@glsxtr@cat:=#1\do
  {%
    \ifdefempty\@glsxtr@cat
    {}%
    {\glssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%
  }%
  \newrobustcmd*#2[1]{##1}%
  \def\@glsxtr@taggingcs{#2}%
  \renewcommand*\@glsxtr@activate@initialtagging{%
    \let#2\@glsxtr@tag
  }%
  \ifundef\@gls@preglossaryhook
  {\GlossariesExtraWarning{Initial tagging requires at least
    glossaries.sty v4.19 to work correctly}}%
  {}%
}
```

Are we using an old version of mfirstuc that has a bug in `\capitalisewords`? If so, patch it so we don't have a problem with a combination of tagging and title case.

fu@checkword@do If this command hasn't been defined, then we have pre v2.02 of mfirstuc

```
\ifundef\mfu@checkword@do
{
  \newcommand*{\mfu@checkword@do}[1]{%
   \ifdefstring{\mfu@checkword@arg}{#1}%
   {%
     \let\@mfu@domakefirstuc\@firstofone
     \listbreak
   }%
   {}%
  }
```

\mfu@checkword `\capitalisewords` was introduced in mfirstuc v1.06. If `\mfu@checkword` hasn't been defined mfirstuc is too old to support the title case attribute.

```
  \ifundef\mfu@checkword
  {
    \newcommand{\@glsxtr@do@titlecaps@warn}{%
     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
      support not available}%
```

One warning should suffice.

```
      \let\@glsxtr@do@titlecaps@warn\relax
    }
  }
  {
    \renewcommand*{\mfu@checkword}[1]{%
      \def\mfu@checkword@arg{#1}%
      \let\@mfu@domakefirstuc\makefirstuc
      \forlistloop\mfu@checkword@do\@mfu@nocaplist
    }
  }
}
{}% no patch required
```

`@titlecaps@warn` Do warning if title case not supported.

```
\newcommand*{\@glsxtr@do@titlecaps@warn}{}
```

`@initialtagging` Used in `\printglossary` but at least v4.19 of glossaries required.

```
\newcommand*\@glsxtr@activate@initialtagging{}
```

`\@glsxtr@tag` Definition of tagging command when used in glossary.

```
\newrobustcmd*{\@glsxtr@tag}[1]{%
  \glsifattribute{\glscurrententrylabel}{tagging}{true}%
  {\glsxtrtagfont{#1}}{#1}%
}
```

`\glsxtrtagfont` Used in the glossary.

```
\newcommand*{\glsxtrtagfont}[1]{\underline{#1}}
```

`preglossaryhook` This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

```
\ifdef\@gls@preglossaryhook
{
  \renewcommand*{\@gls@preglossaryhook}{%
    \@glsxtr@activate@initialtagging
```

Since the glossaries are automatically scoped, `\@glsxtr@org@postdescription` shouldn't already be defined, but check anyway just as a precautionary measure.

```
    \ifundef\@glsxtr@org@postdescription
    {%
      \let\@glsxtr@org@postdescription\glspostdescription
      \renewcommand*{\glspostdescription}{%
        \ifglsentryexists{\glscurrententrylabel}%
        {%
          \glsxtrpostdescription
          \@glsxtr@org@postdescription
        }%
        {}%
      }%
    }%
    {}%
```

Enable the options used by `\@@glsxtrp`:

```
    \glossxtrsetpopts
  }%
}
{}
```

`postdescription` This command will only be used if `\@gls@preglossaryhook` is available *and* the glossary style uses `\glspostdescription` without modifying it. (`\nopostdesc` will suppress this.) The glossaries-extra-stylemods package will add the post description hook to all the predefined styles that don't include it.

```
\newcommand*{\glsxtrpostdescription}{%
  \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
}
```

`postdescgeneral`

```
\newcommand*{\glsxtrpostdescgeneral}{}
```

`xtrpostdescterm`

```
\newcommand*{\glsxtrpostdescterm}{}
```

`postdescacronym`

```
\newcommand*{\glsxtrpostdescacronym}{}
```

`escabbreviation`

```
\newcommand*{\glsxtrpostdescabbreviation}{}
```

`\glsdefpostdesc` Provide a convenient command for defining the post-description hook for the given category.

```
\newcommand*{\glsdefpostdesc}[2]{%
  \csdef{glsxtrpostdesc#1}{#2}%
}
```

`glspostlinkhook` Redefine the post link hook used by commands like `\gls` to make it easier for categories or attributes to modify this action. Since this hook occurs outside the existence check of commands like `\gls`, this needs to be checked again here. Do nothing if the entry hasn't been defined.

```
\renewcommand*{\glspostlinkhook}{%
 \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
}
```

`xtrpostlinkhook` The entry label should already be stored in `\glslabel` by `\@gls@link`.

```
\newcommand*{\glsxtrpostlinkhook}{%
 \glsxtrdiscardperiod{\glslabel}%
 {\glsxtrpostlinkendsentence}%
 {\glsxtrifcustomdiscardperiod
  {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}%
  {\glsxtrpostlink}%
 }%
}
```

`omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
\newcommand*{\glsxtrifcustomdiscardperiod}[2]{#2}
```

`\glsxtrpostlink`

```
\newcommand*{\glsxtrpostlink}{%
 \csuse{glsxtrpostlink\glscategory{\glslabel}}%
}
```

`\glsdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glsxtrpostlink`.

```
\newcommand*{\glsdefpostlink}[2]{%
```

`\ifthenelse` is used to ensure that the expanded value is tested. (The category label must be fully expandable.)

```
  \ifthenelse{\equal{#1}{}}%
  {\PackageError{glossaries-extra}
    {Invalid empty category label in \string\glsdefpostlink}{}}%
  {\csdef{glsxtrpostlink#1}{#2}}%
}
```

`linkendsentence` Done by `\glsxtrpostlinkhook` if a full stop is discarded.

```
\newcommand*{\glsxtrpostlinkendsentence}{%
 \ifcsdef{glsxtrpostlink\glscategory{\glslabel}}
 {%
   \csuse{glsxtrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
   .\spacefactor\sfcode`\. \relax
 }%
 {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
   \spacefactor\sfcode`\. \relax
 }%
}
```

`dDescOnFirstUse` Provide a command for appending the description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
\newcommand*{\glsxtrpostlinkAddDescOnFirstUse}{%
  \glsxtrifwasfirstuse{\space\glsxtrparen{\glsaccessdesc{\glslabel}}}{}%
}
```

`ymbolOnFirstUse` Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
\newcommand*{\glsxtrpostlinkAddSymbolOnFirstUse}{%
  \glsxtrifwasfirstuse
  {%
    \ifglshassymbol{\glslabel}%
    {\space\glsxtrparen{\glsaccesssymbol{\glslabel}}}%
    {}%
  }%
  {}%
}
```

`lDescOnFirstUse` Provide a command for appending the symbol (if defined) and description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
\newcommand*{\glsxtrpostlinkAddSymbolDescOnFirstUse}{%
  \glsxtrifwasfirstuse
  {%
    \space\glsxtrparen
    {%
      \ifglshassymbol{\glslabel}%
      {\glsaccesssymbol{\glslabel}, }%
      {}%
      \glsaccessdesc{\glslabel}%
    }%
  }%
  {}%
}
```

`trdiscardperiod` Discard following period (if present) if the discardperiod attribute is true. If a period is discarded, do the second argument otherwise do the third argument. The entry label is in the first argument. Since this is designed for abbreviations that end with a period, check if the plural form was used (which typically won't end with a period).

```
\newcommand*{\glsxtrdiscardperiod}[3]{%
 \glsxtrifwasfirstuse
 {%
   \glsifattribute{#1}{retainfirstuseperiod}{true}%
   {#3}%
   {%
     \glsifattribute{#1}{discardperiod}{true}%
     {%
       \glsifplural
       {%
         \glsifattribute{#1}{pluraldiscardperiod}{true}%
         {\glsxtrifperiod{#2}{#3}}%
         {#3}%
       }%
       {%
         \glsxtrifperiod{#2}{#3}%
```

```
      }%
    }%
    {#3}%
  }%
}%
{%
  \glsifattribute{#1}{discardperiod}{true}%
  {%
    \glsifplural
    {%
      \glsifattribute{#1}{pluraldiscardperiod}{true}%
      {\glsxtrifperiod{#2}{#3}}%
      {#3}%
    }%
    {%
      \glsxtrifperiod{#2}{#3}%
    }%
  }%
  {#3}%
}%
}
```

`\glsxtrifperiod` Make a convenient user command to check if the next character is a full stop (period). Works like `\@ifstar` but uses `\new@ifnextchar` rather than `\@ifnextchar`

```
\newcommand*{\glsxtrifperiod}[1]{\new@ifnextchar.{\@firstoftwo{#1}}}
```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`glsxtr@punclist` List of characters identified as punctuation marks. (Be careful of babel shorthands!) This doesn't allow for punctuation marks made up from multiple characters (such as `''`).

```
\newcommand*{\glsxtr@punclist}{.,:;?!}
```

`punctuationmark` Add character to punctuation list.

```
\newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}
```

`unctuationmarks` Reset the punctuation list.

```
\newcommand*{\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}
```

`\glsxtrifpunc` `\glsxtrifnextpunc{⟨true part⟩}{⟨false part⟩}`

Test if this is followed by a punctuation mark. (Adapted from `\new@ifnextchar`.)

```
\newcommand*{\glsxtrifnextpunc}[2]{%
  \def\reserved@a{#1}%
  \def\reserved@b{#2}%
  \futurelet\@glspunc@token\glsxtr@ifnextpunc
}
```

`sxtr@ifnextpunc`

```
\newcommand*{\glsxtr@ifnextpunc}{%
 \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
 \reserved@b
}
```

`xtr@ifpunctoken` Test if the token given in the first argument is in the punctuation list.

```
\newcommand*{\glsxtr@ifpunctoken}[1]{%
  \expandafter\@glsxtr@ifpunctoken\expandafter#1\glsxtr@punclist\@nnil
}
```

`xtr@ifpunctoken`

```
\def\@glsxtr@ifpunctoken#1#2{%
  \let\reserved@d=#2%
  \ifx\reserved@d\@nnil
    \let\glsxtr@next\@glsxtr@notfoundinlist
  \else
    \ifx#1\reserved@d
     \let\glsxtr@next\@glsxtr@foundinlist
    \else
     \let\glsxtr@next\@glsxtr@ifpunctoken
    \fi
  \fi
  \glsxtr@next#1%
}
```

`xtr@foundinlist`

```
\def\@glsxtr@foundinlist#1\@nnil{\@firstoftwo}
```

`@notfoundinlist`

```
\def\@glsxtr@notfoundinlist#1{\@secondoftwo}
```

`glsxtrdopostpunc`

`\glsxtrdopostpunc{⟨code⟩}`

If this is followed be a punctuation character, do ⟨*code*⟩ after the character otherwise do ⟨*code*⟩ before whatever comes next.

```
\newcommand{\glsxtrdopostpunc}[1]{%
  \glsxtrifnextpunc{\@glsxtr@swaptwo{#1}}{#1}%
}
```

`@glsxtr@swaptwo`

```
\newcommand{\@glsxtr@swaptwo}[2]{#2#1}
```

1.7 Abbreviations

The “acronym” code from glossaries is misnamed as it’s more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there’s a style for the given category, apply it.

```
\define@key{glsxtrabbrv}{category}{%
 \edef\glscategorylabel{#1}%
 \ifcsdef{@glsabbrv@current@#1}%
 {%
```

Warning should already have been issued.

```
    \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
    \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
    \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
    \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
 }%
 {}%
}
```

Save the short plural form. This may be needed before the entry is defined.

```
\define@key{glsxtrabbrv}{shortplural}{%
  \def\@gls@shortpl{#1}%
}
```

Similarly for the long plural form.

```
\define@key{glsxtrabbrv}{longplural}{%
  \def\@gls@longpl{#1}%
}
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

`\glsshortpltok`

```
\newtoks\glsshortpltok
```

`\glslongpltok`

```
\newtoks\glslongpltok
```

`sxtr@insertdots` Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to `\newabbreviation`. Note that explicitly using the short or shortplural keys will override this.

```
\newcommand*{\@glsxtr@insertdots}[2]{%
  \def#1{}%
  \@glsxtr@insert@dots#1#2\@nnil
}
```

xtr@insert@dots

```
\newcommand*{\@glsxtr@insert@dots}[2]{%
  \ifx\@nnil#2\relax
   \let\@glsxtr@insert@dots@next\@gobble
  \else
   \ifx\relax#2\relax
   \else
     \appto#1{#2.}%
   \fi
   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
  \fi
  \@glsxtr@insert@dots@next#1%
}
```

Similarly provide a way of replacing spaces with `\glsxtrwordsep`, which first needs to be defined:

\glsxtrwordsep

```
\newcommand*{\glsxtrwordsep}{\space}
```

Each word is marked with

\glsxtrword

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

tr@markwordseps

```
\newcommand*{\@glsxtr@markwordseps}[2]{%
  \def#1{}%
  \@glsxtr@mark@wordseps#1#2 \@nnil
}
```

r@mark@wordseps

```
\def\@glsxtr@mark@wordseps#1#2 #3{%
  \ifdefempty{#1}%
  {\def#1{\protect\glsxtrword{#2}}}%
  {\appto#1{\protect\glsxtrwordsep\protect\glsxtrword{#2}}}%
  \ifx\@nnil#3\relax
   \let\@glsxtr@mark@wordseps@next\relax
  \else
   \def\@glsxtr@mark@wordseps@next{%
     \@glsxtr@mark@wordseps#1#3}%
  \fi
  \@glsxtr@mark@wordseps@next
}
```

newabbreviation Define a new generic abbreviation.

```
\newcommand*{\newabbreviation}[4][]{%
  \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
}
```

newabbreviation Internal macro. (bib2gls has an option that needs to temporarily redefine `\newabbreviation`. This is just makes it easier to save and restore the original definition.)

```
\newcommand*{\glsxtr@newabbreviation}[4]{%
  \glskeylisttok{#1}%
  \glslabeltok{#2}%
  \glsshorttok{#3}%
  \glslongtok{#4}%
```

Save the original short and long values (before attribute settings modify them).

```
  \def\glsxtrorgshort{#3}%
  \def\glsxtrorglong{#4}%
```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
  \def\ExtraCustomAbbreviationFields{}%
```

Initialise accessibility settings if required.

```
  \@gls@initaccesskeys
```

Get the category.

```
  \def\glscategorylabel{abbreviation}%
  \glsxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
```

Ignore the shortplural and longplural keys.

```
  \setkeys*{glsxtrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
  \def\@gls@longpl{#4\glspluralsuffix}%
  \let\@gls@default@longpl\@gls@longpl
```

Has the markwords attribute been set?

```
  \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
  {%
    \@glsxtr@markwordseps\@gls@long{#4}%
    \expandafter\def\expandafter\@gls@longpl\expandafter
     {\@gls@long\glspluralsuffix}%
    \let\@gls@default@longpl\@gls@longpl
```

Update `\glslongtok`.

```
    \expandafter\glslongtok\expandafter{\@gls@long}%
  }%
  {}%
```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
  \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
  {%
    \@glsxtr@markwordseps\@gls@short{#3}%
  }%
  {%
```

Has the insertdots attribute been set?

```
    \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
    {%
      \@glsxtr@insertdots\@gls@short{#3}%
      \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
    }%
    {\def\@gls@short{#3}}%
  }%
```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
\glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
{%
  \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
    '\abbrvpluralsuffix}%
}%
{%
```

Has the noshortplural attribute been set?

```
  \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
  {%
    \let\@gls@shortpl\@gls@short
  }%
    {%
    \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
      \abbrvpluralsuffix}%
  }%
}%
```

Update `\glsshorttok`:

```
\expandafter\glsshorttok\expandafter{\@gls@short}%
```

Hook for further customisation if required:

```
\glsxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%
```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```
\setkeys*{glsxtrabbrv}[category]{#1}%
```

Has the plural been explicitly set?

```
\ifx\@gls@default@longpl\@gls@longpl
\else
```

Has the markwords attribute been set?

```
  \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
  {%
    \expandafter\@glsxtr@markwordseps\expandafter\@gls@longpl\expandafter
     {\@gls@longpl}%
  }%
  {}%
\fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
\expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
\expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
\@gls@setup@default@short@access{#3}%
```

Do any extra setup provided by hook:

```
  \newabbreviationhook
```

Define this entry:

```
  \protected@edef\@do@newglossaryentry{%
    \noexpand\newglossaryentry{\the\glslabeltok}%
    {%
      type=\glsxtrabbrvtype,%
      category=abbreviation,%
      short={\the\glsshorttok},%
      shortplural={\the\glsshortpltok},%
      long={\the\glslongtok},%
      longplural={\the\glslongpltok},%
      name={\the\glsshorttok},%
      \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
      \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
      \the\glskeylisttok
    }%
  }%
  \@do@newglossaryentry
  \GlsXtrPostNewAbbreviation
}
```

`evpresetkeyhook`

Hook for extra stuff in `\newabbreviation`

```
\newcommand*{\glsxtrnewabbrevpresetkeyhook}[3]{}
```

`NewAbbreviation`

Hook used by abbreviation styles.

```
\newcommand*{\GlsXtrPostNewAbbreviation}{}
```

`bbreviationhook`

Hook for use with `\newabbreviation`.

```
\newcommand*{\newabbreviationhook}{}
```

`reviationFields`

```
\newcommand*{\CustomAbbreviationFields}{}
```

`\glsxtrparen`

For the parenthetical styles.

```
\newcommand*{\glsxtrparen}[1]{(#1)}
```

`lsxtrfullformat`

Full format without case change.

```
\newcommand*{\glsxtrfullformat}[2]{%
  \glsfirstlongfont{\glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
  \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
}
```

lsxtrfullformat

Full format with case change.

```
\newcommand*{\Glsxtrfullformat}[2]{%
  \glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
  \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
}
```

xtrfullplformat

Plural full format without case change.

```
\newcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
  \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
}
```

xtrfullplformat

Plural full format with case change.

```
\newcommand*{\Glsxtrfullplformat}[2]{%
  \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
  \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
}
```

\glsxtrfullsep

Separator used by full format is a space by default. The argument is the entry's label.

```
\newcommand*{\glsxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with `\glsentryfull` (for example, first use suppresses the long form or uses a footnote).

nlinefullformat

Full format without case change.

```
\newcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}
```

nlinefullformat

Full format with case change.

```
\newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

xtrfullplformat

Plural full format without case change.

```
\newcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}
```

inefullplformat

Plural full format with case change.

```
\newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}
```

Redefine `\glsentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\glsxtrfull` set of commands instead.

\glsentryfull

```
\renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}
```

\Glsentryfull

```
\renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

\glsentryfullpl

```
\renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}
```

\Glsentryfullpl

```
\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

sfirstabbrvfont Font changing command used for the abbreviation on first use or in the full format.

```
\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}
```

bbrvdefaultfont Font changing command used for the abbreviation on first use or in the full format.

```
\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}
```

\glsabbrvfont Font changing command used for the abbreviation on subsequent use.

```
\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}
```

bbrvdefaultfont

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

\glslongfont Font changing command used for the long form in commands like `\glsxtrlong`.

```
\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}
```

longdefaultfont Default font changing command used for the long form in commands like `\glsxtrlong`.

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

lsfirstlongfont Font changing command used for the long form on first use or in the full format.

```
\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}
```

longdefaultfont

```
\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}
```

brvpluralsuffix Default plural suffix. Allow an alternative default suffix for abbreviations.

```
\newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}
```

brvpluralsuffix Default plural suffix.

```
\newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtrfull Full form (no case-change).

```
\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}
\newcommand*\ns@glsxtrfull[2][]{%
  \new@ifnextchar[{\@glsxtr@full{#1}{#2}}%
                  {\@glsxtr@full{#1}{#2}[]}%
}
```

\@glsxtr@full Low-level macro:

```
\def\@glsxtr@full#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\glsxtrinlinefullformat{#2}{#3}}%
```

What should `\glsxtrifwasfirstuse` be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`trsetupfulldefs`

```
\newcommand*{\glsxtrsetupfulldefs}{%
  \let\glsxtrifwasfirstuse\@firstoftwo
}
```

`\Glsxtrfull` Full form (first letter uppercase).

```
\newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
\newcommand*\ns@Glsxtrfull[2][]{%
  \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
                  {\@Glsxtr@full{#1}{#2}[]}%
}
```

`\@Glsxtr@full` Low-level macro:

```
\def\@Glsxtr@full#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrfull Full form (all uppercase).

```
\newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
\newcommand*\ns@GLSxtrfull[2][]{%
  \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}%
                  {\@GLSxtr@full{#1}{#2}[]}%
}
```

\@GLSxtr@full Low-level macro:

```
\def\@GLSxtr@full#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\glsxtrfullpl Plural full form (no case-change).

```
\newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
\newcommand*\ns@glsxtrfullpl[2][]{%
  \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}%
                  {\@glsxtr@fullpl{#1}{#2}[]}%
}
```

\@glsxtr@fullpl Low-level macro:

```
\def\@glsxtr@fullpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```
\newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\ns@Glsxtrfullpl}
\newcommand*\ns@Glsxtrfullpl[2][]{%
  \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
                  {\@Glsxtr@fullpl{#1}{#2}[]}%
}
```

`\@Glsxtr@fullpl` Low-level macro:

```
\def\@Glsxtr@fullpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\GLSxtrfullpl` Plural full form (all upper case).

```
\newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\ns@GLSxtrfullpl}
\newcommand*\ns@GLSxtrfullpl[2][]{%
  \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}%
                  {\@GLSxtr@fullpl{#1}{#2}[]}%
}
```

`\@GLSxtr@fullpl` Low-level macro:

```
\def\@GLSxtr@fullpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```
\newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\ns@glsxtrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@glsxtrshort}[2][]{%
  \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@glsxtrshort#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
```

Need to make sure `\glsabbrvfont` is set correctly.

```
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glsabbrvfont{\glsaccessshort{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\Glsxtrshort`

```
\newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\ns@Glsxtrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@Glsxtrshort}[2][]{%
  \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Glsxtrshort#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glsabbrvfont{\Glsaccessshort{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\GLSxtrshort`

```
\newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\ns@GLSxtrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@GLSxtrshort}[2][]{%
  \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@GLSxtrshort#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase
      {\glsabbrvfont{\glsaccessshort{#2}\ifglsxtrinsertinside#3\fi}%
        \ifglsxtrinsertinside\else#3\fi
      }%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\glsxtrlong

```
\newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\ns@glsxtrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@glsxtrlong}[2][]{%
  \new@ifnextchar[{\@glsxtrlong{#1}{#2}}{\@glsxtrlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@glsxtrlong#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glslongfont{\glsaccesslong{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\Glsxtrlong

```
\newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\ns@Glsxtrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@Glsxtrlong}[2][]{%
  \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Glsxtrlong#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glslongfont{\Glsaccesslong{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrlong

```
\newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\ns@GLSxtrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@GLSxtrlong}[2][]{%
  \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@GLSxtrlong#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
     \mfirstucMakeUppercase
     {\glslongfont{\glsaccesslong{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
     }%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

Plural short forms:

\glsxtrshortpl

```
\newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\ns@glsxtrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@glsxtrshortpl}[2][]{%
```

```
  \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@glsxtrshortpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\Glsxtrshortpl`

```
\newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\ns@Glsxtrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@Glsxtrshortpl}[2][]{%
  \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Glsxtrshortpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\GLSxtrshortpl`

```
\newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@GLSxtrshortpl}[2][]{%
  \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@GLSxtrshortpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase
      {\glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
       \ifglsxtrinsertinside\else#3\fi
      }%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

Plural long forms:

`\glsxtrlongpl`

```
\newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@glsxtrlongpl}[2][]{%
  \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@glsxtrlongpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glslongfont{\glsaccesslongpl{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\Glsxtrlongpl`

```
\newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\ns@Glsxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@Glsxtrlongpl}[2][]{%
  \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Glsxtrlongpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glslongfont{\Glsaccesslongpl{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\GLSxtrlongpl`

```
\newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\ns@GLSxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@GLSxtrlongpl}[2][]{%
  \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@GLSxtrlongpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase
      {\glslongfont{\glsaccesslongpl{#2}\ifglsxtrinsertinside#3\fi}%
       \ifglsxtrinsertinside\else#3\fi
      }%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\glssetabbrvfmt Set the current format for the given category (or the abbreviation category if unset).

```
\newcommand*{\glssetabbrvfmt}[1]{%
  \ifcsdef{@glsabbrv@current@#1}%
  {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
  {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
}
```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```
\newrobustcmd*{\glsuseabbrvfont}[2]{{\glssetabbrvfmt{#2}\glsabbrvfont{#1}}}
```

\glsuselongfont Provide a way to use the long font for a given category for arbitrary text.

```
\newrobustcmd*{\glsuselongfont}[2]{{\glssetabbrvfmt{#2}\glslongfont{#1}}}
```

sxtrgenabbrvfmt Similar to `\glsgenacfmt`, but for abbreviations.

```
\newcommand*{\glsxtrgenabbrvfmt}{%
  \ifdefempty\glscustomtext
  {%
    \ifglsused\glslabel
    {%
```

Subsequent use:

```
      \glsifplural
      {%
```

Subsequent plural form:

```
        \glscapscase
        {%
```

Subsequent plural form, don't adjust case:

```
          \glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
        }%
        {%
```

Subsequent plural form, make first letter upper case:

```
          \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
        }%
        {%
```

Subsequent plural form, all caps:

```
          \mfirstucMakeUppercase
            {\glsxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
        }%
      }%
      {%
```

Subsequent singular form

```
        \glscapscase
        {%
```

Subsequent singular form, don't adjust case:

```
          \glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
        }%
        {%
```

Subsequent singular form, make first letter upper case:

```
          \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
        }%
        {%
```

Subsequent singular form, all caps:

```
          \mfirstucMakeUppercase
            {\glsxtrsubsequentfmt{\glslabel}{\glsinsert}}%
        }%
      }%
    }%
    {%
```

First use:

```
      \glsifplural
      {%
```

First use plural form:

```
        \glscapscase
        {%
```

First use plural form, don't adjust case:

```
          \glsxtrfullplformat{\glslabel}{\glsinsert}%
        }%
        {%
```

First use plural form, make first letter upper case:

```
          \Glsxtrfullplformat{\glslabel}{\glsinsert}%
        }%
        {%
```

First use plural form, all caps:

```
          \mfirstucMakeUppercase
            {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
        }%
      }%
      {%
```

First use singular form

```
        \glscapscase
        {%
```

First use singular form, don't adjust case:

```
          \glsxtrfullformat{\glslabel}{\glsinsert}%
        }%
        {%
```

First use singular form, make first letter upper case:

```
          \Glsxtrfullformat{\glslabel}{\glsinsert}%
        }%
        {%
```

First use singular form, all caps:

```
          \mfirstucMakeUppercase
           {\glsxtrfullformat{\glslabel}{\glsinsert}}%
        }%
      }%
    }%
  }%
  {%
```

User supplied text.

```
    \glscustomtext
  }%
}
```

trsubsequentfmt Subsequent use format (singular no case change).

```
\newcommand*{\glsxtrsubsequentfmt}[2]{%
  \glsabbrvfont{\glsaccessshort{#1}\ifglsxtrinsertinside #2\fi}%
  \ifglsxtrinsertinside \else#2\fi
}
\let\glsxtrdefaultsubsequentfmt\glsxtrsubsequentfmt
```

`subsequentplfmt` Subsequent use format (plural no case change).

```
\newcommand*{\glsxtrsubsequentplfmt}[2]{%
  \glsabbrvfont{\glsaccessshortpl{#1}\ifglsxtrinsertinside #2\fi}%
  \ifglsxtrinsertinside \else#2\fi
}
\let\glsxtrdefaultsubsequentplfmt\glsxtrsubsequentplfmt
```

`trsubsequentfmt` Subsequent use format (singular, first letter uppercase).

```
\newcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glsabbrvfont{\Glsaccessshort{#1}\ifglsxtrinsertinside #2\fi}%
  \ifglsxtrinsertinside \else#2\fi
}
\let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt
```

`subsequentplfmt` Subsequent use format (plural, first letter uppercase).

```
\newcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glsabbrvfont{\Glsaccessshortpl{#1}\ifglsxtrinsertinside #2\fi}%
  \ifglsxtrinsertinside \else#2\fi
}
\let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt
```

1.7.1 Abbreviation Styles Setup

`breviationstyle`

```
\newcommand*{\setabbreviationstyle}[2][abbreviation]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#2}
  {%
    \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
  }%
  {%
```

Have abbreviations already been defined for this category?

```
      \ifcsstring{@glsabbrv@current@#1}{#2}%
      {%
```

Style already set.

```
      }%
      {%
        \def\@glsxtr@dostylewarn{}%
        \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
        {%
          \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
            style has been switched \MessageBreak
            for category ‘#1’, \MessageBreak
            but there have already been entries \MessageBreak
            defined for this category. Unwanted \MessageBreak
            side-effects may result}}%
          \@endfortrue
        }%
        \@glsxtr@dostylewarn
```

Set up the style for the given category.

```
       \csdef{@glsabbrv@current@#1}{#2}%
       \glsxtr@applyabbrvstyle{#2}%
     }%
  }%
}
```

applyabbrvstyle Apply the abbreviation style without existence check.

```
\newcommand*{\glsxtr@applyabbrvstyle}[1]{%
  \csuse{@glsabbrv@dispstyle@setup@#1}%
  \csuse{@glsabbrv@dispstyle@fmts@#1}%
}
```

r@applyabbrvfmt Only apply the style formats.

```
\newcommand*{\glsxtr@applyabbrvfmt}[1]{%
  \csuse{@glsabbrv@dispstyle@fmts@#1}%
}
```

breviationstyle This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```
\newcommand*{\newabbreviationstyle}[3]{%
  \ifcsdef{@glsabbrv@dispstyle@setup@#1}
  {%
    \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
     defined}{}%
  }%
  {%
    \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
      \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
     #2}%
    \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
    \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
    \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
    \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
    \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```
    \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
    \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
    \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
    \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
    #3}%
  }%
}
```

breviationstyle

```
\newcommand*{\renewabbreviationstyle}[3]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#1}
  {%
    \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
  }%
  {%
    \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
     #2}%
    \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
    \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
    \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
    \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
    \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
    #3}%
  }%
}
```

breviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
\newcommand*{\letabbreviationstyle}[2]{%
  \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
  \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
}
```

ecated@abbrstyle `\@glsxtr@deprecated@abbrstyle{⟨old-name⟩}{⟨new-name⟩}`

Define a synonym for a deprecated abbreviation style.

```
\newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
  \csdef{@glsabbrv@dispstyle@setup@#1}{%
    \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
    \csuse{@glsabbrv@dispstyle@setup@#2}%
  }%
  \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
}
```

ecatedAbbrStyle Generate warning for deprecated style use.

```
\newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
  \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
  use ‘#2’ instead}%
}
```

eAbbrStyleSetup

```
\newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
  {%
     \PackageError{glossaries-extra}%
     {Unknown abbreviation style definitions ‘#1’}{}%
  }%
  {%
     \csname @glsabbrv@dispstyle@setup@#1\endcsname
  }%
}
```

seAbbrStyleFmts

```
\newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
  \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
  {%
     \PackageError{glossaries-extra}%
     {Unknown abbreviation style formats ‘#1’}{}%
  }%
  {%
     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
  }%
}
```

1.7.2 Predefined Styles (Default Font)

Define some common styles. These will set the first, firstplural, text and plural keys, even if the regular attribute isn't set to "true". If this attribute is set, commands like `\gls` will use them as per a regular entry, otherwise those keys will be ignored unless explicitly invoked by the user with commands like `\glsfirst`. In order for the first letter uppercase versions to work correctly, `\glsxtrfullformat` needs to be expanded when those keys are set. The final optional argument of `\glsfirst` will behave differently to the final optional argument of `\gls` with some styles.

xtrinsertinside Switch to determine if the insert text should be inside or outside the font changing command. The default is outside.

```
\newif\ifglsxtrinsertinside
\glsxtrinsertinsidefalse
```

trlongshortname

```
\newcommand*{\glsxtrlongshortname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

long-short

```
\newabbreviationstyle{long-short}%
{%
```

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
}
```

Set this as the default style for general abbreviations:

```
\setabbreviationstyle{long-short}
```

ngshortdescsort

```
\newcommand*{\glsxtrlongshortdescsort}{%
 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
}
```

ngshortdescname

```
\newcommand*{\glsxtrlongshortdescname}{%
  \protect\glslongfont{\the\glslongtok}
  \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
}
```

long-short-desc User supplies description. The long form is included in the name.

```
\newabbreviationstyle{long-short-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
```

The text key should only have the short form.

```
    text={\protect\glsabbrvfont{\the\glsshorttok}},%

    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-short}%
}
```

trshortlongname

```
\newcommand*{\glsxtrshortlongname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

short-long Short form followed by long form in parenthesis on first use.

```
\newabbreviationstyle{short-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
}
```

`ortlongdescsort`

```
\newcommand*{\glsxtrshortlongdescsort}{\the\glsshorttok}
```

`ortlongdescname`

```
\newcommand*{\glsxtrshortlongdescname}{%
  \protect\glsabbrvfont{\the\glsshorttok}
  \glsxtrparen{\protect\glslongfont{\the\glslongtok}}%
}
```

`short-long-desc` User supplies description. The long form is included in the name.

```
\newabbreviationstyle{short-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

```
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-long}%
}
```

`ongfootnotefont` Only used by the “footnote” styles.

```
\newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

`ongfootnotefont` Only used by the “footnote” styles.

```
\newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

`xtrabbrvfootnote` `\glsxtrabbrvfootnote{⟨label⟩}{⟨long⟩}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument ⟨*long*⟩ includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```
\newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}
```

`xtrfootnotename`

```
\newcommand*{\glsxtrfootnotename}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

`footnote` Short form followed by long form in footnote on first use.

```
\newabbreviationstyle{footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%

    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

    plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

`short-footnote`

```
\letabbreviationstyle{short-footnote}{footnote}
```

`postfootnote` Similar to the above but the footnote is placed afterwards, outside the link. This avoids nested links and can also move the footnote marker after any following punctuation mark. Pre v1.07 included `\footnote` in the first keys, which was incorrect as it caused duplicate footnotes.

```
\newabbreviationstyle{postfootnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
        \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```
  \renewcommand*{\glsxtrsetupfulldefs}{%
    \let\glsxtrifwasfirstuse\@secondoftwo
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
```

```
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

rt-postfootnote

```
\letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```
\newcommand*{\glsxtrshortnolongname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

short Provide a style that only displays the short form on first use, but the short and long form can be displayed with the “full” commands that use the inline format. If the user supplies a description, the long form won’t be displayed in the predefined glossary styles, but the post description hook can be employed to automatically insert it.

```
\newabbreviationstyle{short}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortnolongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
    text={\protect\glsabbrvfont{\the\glsshorttok}},
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}%
```

```
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

Set this as the default style for acronyms:

```
\setabbreviationstyle[acronym]{short}
```

`short-nolong`

```
\letabbreviationstyle{short-nolong}{short}
```

`rt-nolong-noreg` Like short-nolong but doesn't set the regular attribute.

```
\newabbreviationstyle{short-nolong-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-nolong}%
}
```

trshortdescname

```
\newcommand*{\glsxtrshortdescname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

short-desc The user must supply the description in this style. The long form is added to the name. The short style (possibly with the post-description hooks set) might be a better option.

```
\newabbreviationstyle{short-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
    text={\protect\glsabbrvfont{\the\glsshorttok}},
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`ort-nolong-desc`

```
\letabbreviationstyle{short-nolong-desc}{short-desc}
```

`long-desc-noreg` Like short-nolong-desc but doesn't set the regular attribute.

```
\newabbreviationstyle{short-nolong-desc-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

```
  \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
}
```

`nolong-short` Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```
\newabbreviationstyle{nolong-short}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
  }%
}
```

`ong-short-noreg` Like nolong-short but doesn't set the regular attribute.

```
\newabbreviationstyle{nolong-short-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{nolong-short}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{nolong-short}%
}
```

noshortdescname

```
\newcommand*{\glsxtrlongnoshortdescname}{%
  \protect\glslongfont{\the\glslongtok}%
}
```

`long-desc` Provide a style that only displays the long form, but the long and short form can be displayed with the “full” commands that use the inline format. The predefined glossary styles won’t show the short form. The user must supply a description for this style.

```
\newabbreviationstyle{long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortdescname},
    sort={\the\glslongtok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
    text={\glslongfont{\the\glslongtok}},
    plural={\glslongfont{\the\glslongpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`ng-noshort-desc` Provide a synonym that matches similar styles.

```
\letabbreviationstyle{long-noshort-desc}{long-desc}
```

`hort-desc-noreg` Like long-noshort-desc but doesn't set the regular attribute.

```
\newabbreviationstyle{long-noshort-desc-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
}
```

`longnoshortname`

```
\newcommand*{\glsxtrlongnoshortname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

`long` It doesn't really make a great deal of sense to have a long-only style that doesn't have a description (unless no glossary is required), but the best course of action here is to use the short form as the name and the long form as the description.

```
\newabbreviationstyle{long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
    text={\glslongfont{\the\glslongtok}},
    plural={\glslongfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-desc}%
}
```

`long-noshort` Provide a synonym that matches similar styles.

```
\letabbreviationstyle{long-noshort}{long}
```

`g-noshort-noreg` Like long-noshort but doesn't set the regular attribute.

```
\newabbreviationstyle{long-noshort-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-noshort}%
}
```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glsxtrscfont` Maintained for backward-compatibility.

```
\newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
\newcommand*{\glsabbrvscfont}{\glsxtrscfont}
```

`sxtrfirstscfont` Maintained for backward-compatibility.

```
\newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{#1}}
```

`irstabbrvscfont` Added for consistent naming.

```
\newcommand*{\glsfirstabbrvscfont}{\glsxtrfirstscfont}
```

and for the default short form suffix:

`\glsxtrscsuffix`

```
\newcommand*{\glsxtrscsuffix}{\glstextup{\glsxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
\newabbreviationstyle{long-short-sc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
```

```
      \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
```

Use the default long fonts.

```
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
}
```

g-short-sc-desc

```
\newabbreviationstyle{long-short-sc-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
```

```
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As long-short-sc style:

```
  \GlsXtrUseAbbrStyleFmts{long-short-sc}%
}
```

Now the short (long) version

```
\newabbreviationstyle{short-sc-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
```

```
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
}
```

As before but user provides description

```
\newabbreviationstyle{short-sc-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As short-sc-long style:

```
  \GlsXtrUseAbbrStyleFmts{short-sc-long}%
}
```

`short-sc`

```
\newabbreviationstyle{short-sc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortnolongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
    text={\protect\glsabbrvscfont{\the\glsshorttok}},
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvscfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`short-sc-nolong`

```
\letabbreviationstyle{short-sc-nolong}{short-sc}
```

`short-sc-desc`

```
\newabbreviationstyle{short-sc-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
    text={\protect\glsabbrvscfont{\the\glsshorttok}},
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
```

```
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
}
```

-sc-nolong-desc

```
\letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

nolong-short-sc

```
\newabbreviationstyle{nolong-short-sc}%
{%
  \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
}
```

long-noshort-sc

The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsxtrshort`.

```
\newabbreviationstyle{long-noshort-sc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
    text={\protect\glslongdefaultfont{\the\glslongtok}},
    plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
 }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
 \renewcommand*{\glsxtrfullformat}[2]{%
   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi
 }%
 \renewcommand*{\glsxtrfullplformat}[2]{%
   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi
 }%
 \renewcommand*{\Glsxtrfullformat}[2]{%
   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi
 }%
 \renewcommand*{\Glsxtrfullplformat}[2]{%
   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi
 }%
}
```

`long-sc` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-sc}{long-noshort-sc}
```

`noshort-sc-desc` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
\newabbreviationstyle{long-noshort-sc-desc}%
{%
 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
   \ifglsxtrinsertinside \else##2\fi
 }%
 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
   \ifglsxtrinsertinside \else##2\fi
 }%
```

```
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`long-desc-sc` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

`ort-sc-footnote`

```
\newabbreviationstyle{short-sc-footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

`footnote-sc` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```

`sc-postfootnote`

```
\newabbreviationstyle{short-sc-postfootnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
        \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```
  \renewcommand*{\glsxtrsetupfulldefs}{%
    \let\glsxtrifwasfirstuse\@secondoftwo
  }%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
```

```
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

postfootnote-sc Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}
```

1.7.4 Predefined Styles (Fake Small Capitals)

These styles require the relsize package, which must be loaded by the user. These styles all use:

\glsxtrsmfont Maintained for backward compatibility.

```
\newcommand*{\glsxtrsmfont}[1]{\textsmaller{#1}}
```

\glsabbrvsmfont Added for consistent naming.

```
\newcommand*{\glsabbrvsmfont}{\glsxtrsmfont}
```

sxtrfirstsmfont Maintained for backward compatibility.

```
\newcommand*{\glsxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}
```

irstabbrvsmfont Added for consistent naming.

```
\newcommand*{\glsfirstabbrvsmfont}{\glsxtrfirstsmfont}
```

and for the default short form suffix:

\glsxtrsmsuffix

```
\newcommand*{\glsxtrsmsuffix}{\glsxtrabbrvpluralsuffix}
```

long-short-sm

```
\newabbreviationstyle{long-short-sm}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
```

Use the default long fonts.

```
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
}
```

g-short-sm-desc

```
\newabbreviationstyle{long-short-sm-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As long-short-sm style:

```
  \GlsXtrUseAbbrStyleFmts{long-short-sm}%
}
```

short-sm-long Now the short (long) version

```
\newabbreviationstyle{short-sm-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
```

```
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
}
```

`rt-sm-long-desc` As before but user provides description

```
\newabbreviationstyle{short-sm-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As short-sm-long style:

```
  \GlsXtrUseAbbrStyleFmts{short-sm-long}%
}
```

`short-sm`

```
\newabbreviationstyle{short-sm}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortnolongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}%
```

```
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%

  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`short-sm-nolong`

```
\letabbreviationstyle{short-sm-nolong}{short-sm}
```

`short-sm-desc`

```
\newabbreviationstyle{short-sm-desc}%
```

```
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
}
```

-sm-nolong-desc

```
\letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}
```

nolong-short-sm

```
\newabbreviationstyle{nolong-short-sm}%
{%
  \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
}
```

`long-noshort-sm` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
\newabbreviationstyle{long-noshort-sm}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
    text={\protect\glslongdefaultfont{\the\glslongtok}},
    plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`long-sm` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}
```

`noshort-sm-desc` The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
\newabbreviationstyle{long-noshort-sm-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
\renewcommand*{\glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
```

The inline full form displays the long format followed by the short form in parentheses.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
}%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
```

```
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

long-desc-sm Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}
```

ort-sm-footnote

```
\newabbreviationstyle{short-sm-footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
```

```
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

`footnote-sm` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}
```

`sm-postfootnote`

```
\newabbreviationstyle{short-sm-postfootnote}%
{%
```

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
        \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```
  \renewcommand*{\glsxtrsetupfulldefs}{%
    \let\glsxtrifwasfirstuse\@secondoftwo
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

```
\renewcommand*{\Glsxtrfullformat}[2]{%
  \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullplformat}[2]{%
  \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
```

The first use full form and the inline full form use the short (long) style.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
}%
}
```

`postfootnote-sm` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}
```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```
\newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%
```

`irstabbrvemfont`

```
\newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%
```

The default short form suffix:

`\glsxtremsuffix`

```
\newcommand*{\glsxtremsuffix}{\glsxtrabbrvpluralsuffix}
```

`firstlongemfont` Only used by the “long-em” styles.

```
\newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%
```

`\glslongemfont` Only used by the “long-em” styles.

```
\newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em` The long form is just set in the default long font.

```
\newabbreviationstyle{long-short-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
```

Use the default long fonts.

```
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
```

```
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
}
```

g-short-em-desc

```
\newabbreviationstyle{long-short-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As long-short-em style:

```
  \GlsXtrUseAbbrStyleFmts{long-short-em}%
}
```

ong-em-short-em

```
\newabbreviationstyle{long-em-short-em}%
{%
```

`\glslongemfont` is used in the description since `\glsdesc` doesn't set the style.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongemfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
    description={\protect\glslongemfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
}
```

m-short-em-desc

```
\newabbreviationstyle{long-em-short-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongemfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
}
```

short-em-long Now the short (long) version

```
\newabbreviationstyle{short-em-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

Mostly as short-long style:

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
}
```

`rt-em-long-desc` As before but user provides description

```
\newabbreviationstyle{short-em-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-em-long}%
}
```

hort-em-long-em

```
\newabbreviationstyle{short-em-long-em}%
{%
```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\protect\glslongemfont{\the\glslongtok}},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
  }%
}
```

em-long-em-desc

```
\newabbreviationstyle{short-em-long-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},%
    sort={\glsxtrshortlongdescsort},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

```
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
}
```

short-em

```
\newabbreviationstyle{short-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortnolongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
    text={\protect\glsabbrvemfont{\the\glsshorttok}},
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvemfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%

  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvemfont{\Glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won’t be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`short-em-nolong`

```
\letabbreviationstyle{short-em-nolong}{short-em}
```

`short-em-desc`

```
\newabbreviationstyle{short-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
    text={\protect\glsabbrvemfont{\the\glsshorttok}},
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
```

```
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`-em-nolong-desc`

```
\letabbreviationstyle{short-em-nolong-desc}{short-em-desc}
```

`nolong-short-em`

```
\newabbreviationstyle{nolong-short-em}%
{%
  \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-em-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
}
```

`long-noshort-em` The short form is explicitly invoked through commands like `\glsshort`.

```
\newabbreviationstyle{long-noshort-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
    text={\protect\glslongdefaultfont{\the\glslongtok}},
    plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
\renewcommand*{\glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
```

The inline full form displays the long format followed by the short form in parentheses.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
}%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
```

```
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`long-em` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-em}{long-noshort-em}
```

`g-em-noshort-em` The short form is explicitly invoked through commands like `\glsshort`.

```
\newabbreviationstyle{long-em-noshort-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongemfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
    text={\protect\glslongemfont{\the\glslongtok}},
    plural={\protect\glslongemfont{\the\glslongpltok}},%
    description={\protect\glslongemfont{\the\glslongtok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
```

```
 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
 \renewcommand*{\glsxtrinlinefullformat}[2]{%
   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
 }%
 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
 }%
 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
 }%
 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
 }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
 \renewcommand*{\glsxtrfullformat}[2]{%
   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi
 }%
 \renewcommand*{\glsxtrfullplformat}[2]{%
   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi
 }%
 \renewcommand*{\Glsxtrfullformat}[2]{%
   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi
 }%
 \renewcommand*{\Glsxtrfullplformat}[2]{%
   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi
 }%
}
```

`oshort-em-noreg` Like long-em-noshort-em but doesn't set the regular attribute.

```
\newabbreviationstyle{long-em-noshort-em-noreg}%
{%
 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
}
```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
\newabbreviationstyle{long-noshort-em-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
```

```
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`long-desc-em` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

`noshort-em-desc` The short form is explicitly invoked through commands like `\glsshort`. The long form is emphasized.

```
\newabbreviationstyle{long-em-noshort-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortdescname},
    sort={\the\glslongtok},
    first={\protect\glsfirstlongemfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
    text={\glslongemfont{\the\glslongtok}},
    plural={\glslongemfont{\the\glslongpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`t-em-desc-noreg` Like long-em-noshort-em-desc but doesn't set the regular attribute.

```
\newabbreviationstyle{long-em-noshort-em-desc-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
}
```

`ort-em-footnote`

```
\newabbreviationstyle{short-em-footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
```

```
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
```

```
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

`footnote-em` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

`em-postfootnote`

```
\newabbreviationstyle{short-em-postfootnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
        \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
```

```
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```
  \renewcommand*{\glsxtrsetupfulldefs}{%
    \let\glsxtrifwasfirstuse\@secondoftwo
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

postfootnote-em Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}
```

1.7.6 Predefined Styles (User Parentheses Hook)

These styles allow the user to adjust the parenthetical forms. These styles all test for the existence of the field given by:

glsxtruserfield Default is the useri field.

```
\newcommand*{\glsxtruserfield}{useri}
```

glsxtruserparen The format of the parenthetical information. The first argument is the long/short form. The second argument is the entry's label. If `\glscurrentfieldvalue` has been defined, then we have at least glossaries v4.23, which makes it easier for the user to adjust this.

```
\ifdef\glscurrentfieldvalue
{
  \newcommand*{\glsxtruserparen}[2]{%
    \glsxtrfullsep{#2}%
    \glsxtrparen
     {#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
  }
}
{
  \newcommand*{\glsxtruserparen}[2]{%
    \glsxtrfullsep{#2}%
    \glsxtrparen
      {#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{}}%
  }
}
```

Font used for short form:

lsabbrvuserfont

```
\newcommand*{\glsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}
```

Font used for short form on first use:

stabbrvuserfont

```
\newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
\newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
\newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
\newcommand*{\glsxtrusersuffix}{\glsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
\newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
\newabbreviationstyle{long-short-user}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}%
     \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
      {\the\glslabeltok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
     \protect\glsxtruserparen
      {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
    description={\protect\glsuserdescription{\the\glslongtok}%
     {\the\glslabeltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
  }%
}
```

`-postshort-user` Like long-short-user but defers the parenthetical matter to after the link.

```
\newabbreviationstyle{long-postshort-user}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
    description={\protect\glsuserdescription{\the\glslongtok}%
     {\the\glslabeltok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

In-line format:

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
  }%
}
```

ortuserdescname

```
\newcommand*{\glsxtrlongshortuserdescname}{%
  \protect\glslonguserfont{\the\glslongtok}%
  \protect\glsxtruserparen
   {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
}
```

short-user-desc Like long-postshort-user but the user supplies the description.

```
\newabbreviationstyle{long-postshort-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortuserdescname},
    sort={\the\glslongtok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
}
```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```
\newabbreviationstyle{short-postlong-user}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
    description={\protect\glsuserdescription{\the\glslongtok}%
     {\the\glslabeltok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

In-line format:

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
  }%
}
```

`onguserdescname`

```
\newcommand*{\glsxtrshortlonguserdescname}{%
 \protect\glsabbrvuserfont{\the\glsshorttok}%
 \protect\glsxtruserparen
   {\protect\glslonguserfont{\the\glslongpltok}}%
   {\the\glslabeltok}%
}
```

`tlong-user-desc` Like short-postlong-user but leaves the user to specify the description.

```
\newabbreviationstyle{short-postlong-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlonguserdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

    text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
}
```

short-user-desc

```
\newabbreviationstyle{long-short-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortuserdescname},
    sort={\glsxtrlongshortdescsort},%

    first={\protect\glsfirstlonguserfont{\the\glslongtok}%
     \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
      {\the\glslabeltok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
     \protect\glsxtruserparen
      {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-short-user}%
}
```

short-long-user

```
\newabbreviationstyle{short-long-user}%
{%
```

`\glslonguserfont` is used in the description since `\glsdesc` doesn't set the style. (Now in `\glsuserdescription`.)

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\protect\glsuserdescription{\the\glslongtok}%
      {\the\glslabeltok}},%
    first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
       {\the\glslabeltok}},%
    firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
       {\the\glslabeltok}},%

    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
  }%
}
```

-long-user-desc

```
\newabbreviationstyle{short-long-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlonguserdescname},
    sort={\glsxtrshortlongdescsort},%
    first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
       {\the\glslabeltok}},%
    firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
       {\the\glslabeltok}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-long-user}%
}
```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the markwords attribute. They check if the inserted material (provided by the final optional argument of commands like `\gls`) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like `\glsxtrlong` set `\glsinsert` to empty with the entire link-text stored in `\glscustomtext`.

trifhyphenstart Checks if the argument starts with a hyphen. The argument may be `\glsinsert` so check for that and expand.

```
\newrobustcmd*{\glsxtrifhyphenstart}[3]{%
  \ifx\glsinsert#1\relax
   \expandafter\@glsxtrifhyphenstart#1\relax\relax
     \@end@glsxtrifhyphenstart{#2}{#3}%
  \else
   \@glsxtrifhyphenstart#1\relax\relax\@end@glsxtrifhyphenstart{#2}{#3}%
  \fi
}
```

trifhyphenstart

```
\def\@glsxtrifhyphenstart#1#2\@end@glsxtrifhyphenstart#3#4{%
  \ifx-#1\relax#3\else #4\fi
}
```

rlonghyphenshort

```
\glsxtrlonghyphenshort{⟨label⟩}{⟨long⟩}{⟨short⟩}{⟨insert⟩}
```

The ⟨*long*⟩ and ⟨*short*⟩ arguments may be the plural form. The ⟨*long*⟩ argument may also be the first letter uppercase form.

```
\newcommand*{\glsxtrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
```

If ⟨*insert*⟩ starts with a hyphen, redefine `\glsxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glsxtrwordsep` if ⟨*insert*⟩ doesn't start with a hyphen.

```
   \glsxtrifhyphenstart{#4}{\def\glsxtrwordsep{-}}{}%
   \glsfirstlonghyphenfont{#2\ifglsxtrinsertinside{#4}\fi}%
   \ifglsxtrinsertinside\else{#4}\fi
   \glsxtrfullsep{#1}%
   \glsxtrparen{\glsfirstabbrvhyphenfont{#3\ifglsxtrinsertinside{#4}\fi}%
    \ifglsxtrinsertinside\else{#4}\fi}%
 }%
}
```

abbrvhyphenfont

```
\newcommand*{\glsabbrvhyphenfont}{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
\newcommand*{\glsfirstabbrvhyphenfont}{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
\newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%
```

tlonghyphenfont

```
\newcommand*{\glsfirstlonghyphenfont}{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
\newcommand*{\glsxtrhyphensuffix}{\glsxtrabbrvpluralsuffix}
```

`en-short-hyphen` Designed for use with the markwords attribute.

```
\newabbreviationstyle{long-hyphen-short-hyphen}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
    description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsxtrlonghyphenshort{##1}{\glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsxtrlonghyphenshort{##1}{\glsaccesslongpl{##1}}%
     {\glsaccessshortpl{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
     {\glsaccessshortpl{##1}}{##2}%
  }%
}
```

`ort-hyphen-desc` Like long-hyphen-short-hyphen but the description must be supplied by the user.

```
\newabbreviationstyle{long-hyphen-short-hyphen-desc}%
```

```
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
}
```

onghyphennoshort

```
\glsxtrlonghyphennoshort{⟨label⟩}{⟨long⟩}{⟨insert⟩}
```

```
\newcommand*{\glsxtrlonghyphennoshort}[3]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
```

If ⟨*insert*⟩ starts with a hyphen, redefine `\glsxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glsxtrwordsep` if ⟨*insert*⟩ doesn't start with a hyphen.

```
   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
   \glsfirstlonghyphenfont{#2\ifglsxtrinsertinside{#3}\fi}%
   \ifglsxtrinsertinside\else{#3}\fi
 }%
}
```

hort-desc-noreg

This version doesn't show the short form (except explicitly with `\glsxtrshort`). Since `\glsxtrshort` doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough.

```
\newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortdescname},
    sort={\expandonce\glsxtrorglong},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
  }%
}
```

n-noshort-noreg

It doesn't really make a great deal of sense to have a long-only style that doesn't have a description (unless no glossary is required), but the best course of action here is to use the short form as the name and the long form as the description.

```
\newabbreviationstyle{long-hyphen-noshort-noreg}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    text={\protect\glslonghyphenfont{\the\glslongtok}},%
    plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-desc}%
}
```

glsxtrlonghyphen

\glsxtrlonghyphen{⟨*long*⟩}{⟨*label*⟩}{⟨*insert*⟩}

Used by long-hyphen-postshort-hyphen. The ⟨*insert*⟩ is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```
\newcommand*{\glsxtrlonghyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
   \glsfirstlonghyphenfont{#1}%
 }%
}
```

rposthyphenshort

\glsxtrposthyphenshort{⟨*label*⟩}{⟨*insert*⟩}

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glsxtrlonghyphenshort` but omits the ⟨*long*⟩ part. This always uses the singular short form.

```
\newcommand*{\glsxtrposthyphenshort}[2]{%
 {%
   \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
   \ifglsxtrinsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
   \glsxtrfullsep{#1}%
   \glsxtrparen
   {\glsfirstabbrvhyphenfont{\glsentryshort{#1}\ifglsxtrinsertinside{#2}\fi}%
    \ifglsxtrinsertinside\else{#2}\fi
   }%
 }%
}
```

hyphensubsequent

\glsxtrposthyphensubsequent{⟨*label*⟩}{⟨*insert*⟩}

Format in the post-link hook for subsequent use. The label is ignored by default.

```
\newcommand*{\glsxtrposthyphensubsequent}[2]{%
  \glsabbrvfont{\ifglsxtrinsertinside {#2}\fi}%
  \ifglsxtrinsertinside \else{#2}\fi
}
```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
\newabbreviationstyle{long-hyphen-postshort-hyphen}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
    description={\protect\glslonghyphenfont{\the\glslongtok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtrposthyphenshort{\glslabel}{\glsinsert}%
      }%
      {%
```

Put the insertion into the post-link:

```
        \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
      }%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

Subsequent use needs to omit the insertion:

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glsabbrvfont{\glsaccessshort{##1}}%
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glsabbrvfont{\glsaccessshortpl{##1}}%
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
```

```
    \glsabbrvfont{\Glsaccessshort{##1}}%
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glsabbrvfont{\Glsaccessshortpl{##1}}%
  }%
```

First use full form:

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
  }%
```

In-line format.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlonghyphenfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside{##2}\fi}%
    \ifglsxtrinsertinside \else{##2}\fi
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlonghyphenfont{\glsaccesslongpl{##1}%
      \ifglsxtrinsertinside{##2}\fi}%
    \ifglsxtrinsertinside \else{##2}\fi
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlonghyphenfont{\Glsaccesslong{##1}%
      \ifglsxtrinsertinside{##2}\fi}%
    \ifglsxtrinsertinside \else{##2}\fi
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}%
      \ifglsxtrinsertinside{##2}\fi}%
    \ifglsxtrinsertinside \else{##2}\fi
  }%
}
```

`ort-hyphen-desc` Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```
\newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtrposthyphenshort{\glslabel}{\glsinsert}%
      }%
      {%
```

Put the insertion into the post-link:

```
        \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
      }%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
}
```

rshorthyphenlong `\glsxtrshorthyphenlong{⟨label⟩}{⟨short⟩}{⟨long⟩}{⟨insert⟩}`

The ⟨*long*⟩ and ⟨*short*⟩ arguments may be the plural form. The ⟨*long*⟩ argument may also be the first letter uppercase form.

```
\newcommand*{\glsxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
```

If ⟨*insert*⟩ starts with a hyphen, redefine `\glsxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.)

```
   \glsxtrifhyphenstart{#4}{\def\glsxtrwordsep{-}}{}%
   \glsfirstabbrvhyphenfont{#2\ifglsxtrinsertinside{#4}\fi}%
   \ifglsxtrinsertinside\else{#4}\fi
   \glsxtrfullsep{#1}%
   \glsxtrparen{\glsfirstlonghyphenfont{#3\ifglsxtrinsertinside{#4}\fi}%
    \ifglsxtrinsertinside\else{#4}\fi}%
 }%
}
```

hen-long-hyphen Designed for use with the markwords attribute.

```
\newabbreviationstyle{short-hyphen-long-hyphen}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
    description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsxtrshorthyphenlong{##1}%
     {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsxtrshorthyphenlong{##1}%
     {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
  }%
}
```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```
\newabbreviationstyle{short-hyphen-long-hyphen-desc}%
```

```
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
}
```

lsxtrshorthyphen

`\glsxtrshorthyphen{⟨short⟩}{⟨label⟩}{⟨insert⟩}`

Used by short-hyphen-postlong-hyphen. The ⟨*insert*⟩ is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```
\newcommand*{\glsxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
   \glsfirstabbrvhyphenfont{#1}%
 }%
}
```

trposthyphenlong

`\glsxtrposthyphenlong{⟨label⟩}{⟨insert⟩}`

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glsxtrshorthyphenlong` but omits the ⟨*short*⟩ part. This always uses the singular long form.

```
\newcommand*{\glsxtrposthyphenlong}[2]{%
 {%
   \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
   \ifglsxtrinsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
   \glsxtrfullsep{#1}%
   \glsxtrparen
   {\glsfirstlonghyphenfont{\glsentrylong{#1}\ifglsxtrinsertinside{#2}\fi}%
     \ifglsxtrinsertinside\else{#2}\fi
   }%
 }%
}
```

`postlong-hyphen` Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
\newabbreviationstyle{short-hyphen-postlong-hyphen}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
    description={\protect\glslonghyphenfont{\the\glslongtok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtrposthyphenlong{\glslabel}{\glsinsert}%
      }%
      {%
```

Put the insertion into the post-link:

```
        \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
      }%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

Subsequent use needs to omit the insertion:

```
\renewcommand*{\glsxtrsubsequentfmt}[2]{%
  \glsabbrvfont{\glsaccessshort{##1}}%
}%
\renewcommand*{\glsxtrsubsequentplfmt}[2]{%
  \glsabbrvfont{\glsaccessshortpl{##1}}%
}%
\renewcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glsabbrvfont{\Glsaccessshort{##1}}%
}%
\renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glsabbrvfont{\Glsaccessshortpl{##1}}%
}%
```

First use full form:

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
  \glsxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
}%
\renewcommand*{\Glsxtrfullplformat}[2]{%
  \glsxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
}%
```

In-line format. Commands like `\glsxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstabbrvhyphenfont{\glsaccessshort{##1}%
    \ifglsxtrinsertinside{##2}\fi}%
  \ifglsxtrinsertinside \else{##2}\fi
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}%
    \ifglsxtrinsertinside{##2}\fi}%
  \ifglsxtrinsertinside \else{##2}\fi
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}%
    \ifglsxtrinsertinside{##2}\fi}%
  \ifglsxtrinsertinside \else{##2}\fi
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}%
    \ifglsxtrinsertinside{##2}\fi}%
  \ifglsxtrinsertinside \else{##2}\fi
}%
```

```
}
```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```
\newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},%
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtrposthyphenlong{\glslabel}{\glsinsert}%
      }%
      {%
```

Put the insertion into the post-link:

```
        \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
      }%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
}
```

1.7.8 Predefined Styles (No Short on First Use)

These styles show only the long form on first use and only the short form on subsequent use.

lsabbrvonlyfont

```
\newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%
```

stabbrvonlyfont

```
\newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

glslongonlyfont

```
\newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

rstlongonlyfont

```
\newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
\newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```
\newcommand*{\glsxtronlyname}{%
  \protect\glsabbrvonlyfont{\the\glsshorttok}%
}
```

only-short-only

```
\newabbreviationstyle{long-only-short-only}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtronlyname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
    description={\protect\glslongonlyfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
```

The first use full form doesn't show the short form.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
```

```
    \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The inline full form does show the short form.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
  }%
}
```

`xtronlydescsort`

```
\newcommand*{\glsxtronlydescsort}{\the\glslongtok}
```

`xtronlydescname`

```
\newcommand*{\glsxtronlydescname}{%
  \protect\glslongfont{\the\glslongtok}%
}
```

`short-only-desc`

```
\newabbreviationstyle{long-only-short-only-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtronlydescname},
    sort={\glsxtronlydescsort},%
    first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
    text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
}
```

1.8 Using Entries in Headings

There are four main problems with using entries in sectioning commands: they can mess with the first use flag if they end up in the table of contents, they can add unwanted numbers to the entry's location list, the label is corrupted if used inside `\MakeUppercase` (which is used by the default headings style) and they need to be expandable for PDF bookmarks. The glossaries package therefore recommends the use of the expandable commands, such as `\glsentryshort`, instead but this doesn't reflect the formatting since it doesn't include `\glsabbrvfont`. The commands below are an attempt to get around these problems.

The PDF bookmark issue can easily be fixed with hyperref's `\texorpdfstring` which can simply use the expandable command in the PDF string case. The TeX string case can now use `\glsxtrshort` with the noindex key set, which prevents the unwanted additions to the location list, and the hyper key set to false, which prevents the problem of nested links. This just leaves one thing left that needs to be dealt with, and that's what to do if the heading style uses `\MakeUppercase`.

Note that glossaries automatically loads textcase, so the label can be protected from case change with textcase's `\NoCaseChange`. This means that we don't have a problem provided the page style uses `\MakeTextUppercase`, but the default heading page style uses `\MakeUppercase`.

To get around this, save the original definition of `\markboth` and `\markright` and adjust it so that `\MakeUppercase` is temporarily redefined to `\MakeTextUppercase`. Some packages or classes redefine these commands, so we can't just assume they still have the original kernel definition.

`\markright` Save original definition:

```
\let\@glsxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```
\renewcommand*{\markright}[1]{%
 \glsxtrmarkhook
 \@glsxtr@org@markright{\@glsxtrinmark#1\@glsxtrnotinmark}%
 \glsxtrrestoremarkhook
}
```

\markboth Save original definition:

```
\let\@glsxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
\renewcommand*{\markboth}[2]{%
 \glsxtrmarkhook
 \@glsxtr@org@markboth
   {\@glsxtrinmark#1\@glsxtrnotinmark}%
   {\@glsxtrinmark#2\@glsxtrnotinmark}%
 \glsxtrrestoremarkhook
}
```

Also do this for \@starttoc

\@starttoc Save original definition:

```
\let\@glsxtr@org@@starttoc\@starttoc
```

Redefine:

```
\renewcommand*{\@starttoc}[1]{%
 \glsxtrmarkhook
 \@glsxtrinmark
 \@glsxtr@org@@starttoc{#1}%
 \@glsxtrnotinmark
 \glsxtrrestoremarkhook
}
```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```
\newcommand*{\glsxtrRevertMarks}{%
  \let\markright\@glsxtr@org@markright
  \let\markboth\@glsxtr@org@markboth
  \let\@starttoc\@glsxtr@org@@starttoc
}
```

rRevertTocMarks Just restores \@starttoc.

```
\newcommand*{\glsxtrRevertTocMarks}{%
  \let\@starttoc\@glsxtr@org@@starttoc
}
```

\glsxtrifinmark

```
\newcommand*{\glsxtrifinmark}[2]{#2}
```

\@glsxtrinmark

```
\newrobustcmd*{\@glsxtrinmark}{%
  \let\glsxtrifinmark\@firstoftwo
}
```

glsxtrnotinmark

```
\newrobustcmd*{\@glsxtrnotinmark}{%
  \let\glsxtrifinmark\@secondoftwo
}
```

eorpdforheading

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
}
{
  \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
}
```

\glsxtrmarkhook Hook used in new definition of `\markboth` and `\markright` to make some changes to apply to the marks:

```
\newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
  \let\@glsxtr@org@MakeUppercase\MakeUppercase
  \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
  \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
  \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
  \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
  \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
  \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
  \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
  \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
  \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
  \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
  \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
  \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
  \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
  \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
  \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
  \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
  \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
  \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
  \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
  \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
  \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
  \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
  \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
  \let\glsxtrifinmark\@firstoftwo
  \let\MakeUppercase\MakeTextUppercase
  \let\glsxtrtitleorpdforheading\@thirdofthree
  \let\glsxtrtitleshort\glsxtrheadshort
  \let\glsxtrtitleshortpl\glsxtrheadshortpl
  \let\Glsxtrtitleshort\Glsxtrheadshort
  \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
  \let\glsxtrtitlename\glsxtrheadname
  \let\Glsxtrtitlename\Glsxtrheadname
  \let\glsxtrtitletext\glsxtrheadtext
  \let\Glsxtrtitletext\Glsxtrheadtext
  \let\glsxtrtitleplural\glsxtrheadplural
  \let\Glsxtrtitleplural\Glsxtrheadplural
  \let\glsxtrtitlefirst\glsxtrheadfirst
  \let\Glsxtrtitlefirst\Glsxtrheadfirst
  \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
  \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
  \let\glsxtrtitlelong\glsxtrheadlong
  \let\glsxtrtitlelongpl\glsxtrheadlongpl
  \let\Glsxtrtitlelong\Glsxtrheadlong
  \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
  \let\glsxtrtitlefull\glsxtrheadfull
  \let\glsxtrtitlefullpl\glsxtrheadfullpl
  \let\Glsxtrtitlefull\Glsxtrheadfull
  \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
}
```

restoremarkhook Hook used in new definition of \markboth and \markright to restore the modified definitions. (This is in case the original \markboth and \markright shouldn't be grouped for some reason. There already is some grouping within those original definitions, but some of the code lies outside that grouping, and possibly there's a reason for it.)

```
\newcommand*{\glsxtrrestoremarkhook}{%
  \let\glsxtrifinmark\@secondoftwo
  \let\MakeUppercase\@glsxtr@org@MakeUppercase
  \let\glsxtrtitleorpdforheading\@glsxtr@org@glsxtrtitleorpdforheading
  \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
  \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
  \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
  \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
  \let\glsxtrtitlename\@glsxtr@org@glsxtrtitlename
  \let\Glsxtrtitlename\@glsxtr@org@Glsxtrtitlename
  \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
  \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
  \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
  \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
  \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
  \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
  \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural
```

```
 \let\Glsxtrtitlefirstplural\@glsxtr@org@Glsxtrtitlefirstplural
 \let\glsxtrtitlelong\@glsxtr@org@glsxtrtitlelong
 \let\glsxtrtitlelongpl\@glsxtr@org@glsxtrtitlelongpl
 \let\Glsxtrtitlelong\@glsxtr@org@Glsxtrtitlelong
 \let\Glsxtrtitlelongpl\@glsxtr@org@Glsxtrtitlelongpl
 \let\glsxtrtitlefull\@glsxtr@org@glsxtrtitlefull
 \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
 \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
 \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
}
```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

`glsxtrheadshort` Command used to display short form in the page header.

```
\newcommand*{\glsxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshort[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrshort[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitleshort}[1]{%
  \glsxtrshort[noindex,hyper=false]{#1}[]%
}
```

`sxtrheadshortpl` Command used to display plural short form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrshortpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadshortpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitleshortpl}[1]{%
  \glsxtrshortpl[noindex,hyper=false]{#1}[]%
}
```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshort[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrshort[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitleshort}[1]{%
  \Glsxtrshort[noindex,hyper=false]{#1}[]%
}
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadshortpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
  \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
}
```

`\glsxtrheadname` As above but for the name value.

```
\newcommand*{\glsxtrheadname}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSname[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsname[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`glsxtrtitlename` Command to display name value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlename}[1]{%
  \glsname[noindex,hyper=false]{#1}[]%
}
```

`\Glsxtrheadname` First letter converted to upper case

```
\newcommand*{\Glsxtrheadname}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSname[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsname[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`Glsxtrtitlename` Command to display name value in section title and table of contents with the first letter changed to upper case.

```
%\changes{1.21}{2017-11-03}{new}
\newrobustcmd*{\Glsxtrtitlename}[1]{%
  \Glsname[noindex,hyper=false]{#1}[]%
}
```

`\glsxtrheadtext` As above but for the text value.

```
\newcommand*{\glsxtrheadtext}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLStext[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glstext[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

glsxtrtitletext Command to display text value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitletext}[1]{%
  \glstext[noindex,hyper=false]{#1}[]%
}
```

\Glsxtrheadtext First letter converted to upper case

```
\newcommand*{\Glsxtrheadtext}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLStext[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glstext[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitletext}[1]{%
  \Glstext[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadplural As above but for the plural value.

```
\newcommand*{\glsxtrheadplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitleplural Command to display plural value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitleplural}[1]{%
  \glsplural[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadplural` Convert first letter to upper case.

```
\newcommand*{\Glsxtrheadplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`sxtrtitleplural` Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitleplural}[1]{%
  \Glsplural[noindex,hyper=false]{#1}[]%
}
```

`glsxtrheadfirst` As above but for the first value.

```
\newcommand*{\glsxtrheadfirst}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirst[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsfirst[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`lsxtrtitlefirst` Command to display first value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefirst}[1]{%
  \glsfirst[noindex,hyper=false]{#1}[]%
}
```

`Glsxtrheadfirst` First letter converted to upper case

```
\newcommand*{\Glsxtrheadfirst}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirst[noindex,hyper=false]{#1}[]%
   }%
   {%
```

```
     \Glsfirst[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitlefirst}[1]{%
  \Glsfirst[noindex,hyper=false]{#1}[]%
}
```

headfirstplural As above but for the firstplural value.

```
\newcommand*{\glsxtrheadfirstplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirstplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsfirstplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

itlefirstplural Command to display firstplural value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefirstplural}[1]{%
  \glsfirstplural[noindex,hyper=false]{#1}[]%
}
```

headfirstplural First letter converted to upper case

```
\newcommand*{\Glsxtrheadfirstplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirstplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsfirstplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

itlefirstplural Command to display first value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
  \Glsfirstplural[noindex,hyper=false]{#1}[]%
}
```

`\glsxtrheadlong` Command used to display long form in the page header.

```
\newcommand*{\glsxtrheadlong}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlong[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrlong[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlelong}[1]{%
  \glsxtrlong[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadlongpl` Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrlongpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadlongpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
      \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
   {%
      \glsxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlelongpl}[1]{%
  \glsxtrlongpl[noindex,hyper=false]{#1}[]%
}
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadlong}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlong[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrlong[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlelong}[1]{%
  \Glsxtrlong[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadlongpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
  \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
}
```

`\glsxtrheadfull` Command used to display full form in the page header.

```
\newcommand*{\glsxtrheadfull}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrfull[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrfull[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefull}[1]{%
  \glsxtrfull[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadfullpl` Command used to display plural full form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrfullpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadfullpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
      \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
   {%
      \glsxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefullpl}[1]{%
  \glsxtrfullpl[noindex,hyper=false]{#1}[]%
}
```

`\Glsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadfull}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
      \GLSxtrfull[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrfull[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlefull}[1]{%
  \Glsxtrfull[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadfullpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
  \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
}
```

`\glsfmtshort` Provide a way of using the formatted short form in section headings. If hyperref has been loaded, use `\texorpdfstring` for convenience in PDF bookmarks.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtshort}[1]{%
    \texorpdfstring
      {\glsxtrtitleshort{#1}}%
      {\glsentryshort{#1}}%
  }
}
{
  \newcommand*{\glsfmtshort}[1]{%
    \glsxtrtitleshort{#1}}
}
```

Similarly for the plural version.

`\glsfmtshortpl`

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtshortpl}[1]{%
    \texorpdfstring
      {\glsxtrtitleshortpl{#1}}%
      {\glsentryshortpl{#1}}%
  }
}
{
  \newcommand*{\glsfmtshortpl}[1]{%
```

```
   \glsxtrtitleshortpl{#1}}
}
```

The case-changing version isn't suitable for PDF bookmarks, so the PDF alternative uses the non-case-changing version.

`\Glsfmtshort` Singular form (first letter uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtshort}[1]{%
    \texorpdfstring
      {\Glsxtrtitleshort{#1}}%
      {\glsentryshort{#1}}%
  }
}
{
  \newcommand*{\Glsfmtshort}[1]{%
   \Glsxtrtitleshort{#1}}
}
```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtshortpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitleshortpl{#1}}%
    {\glsentryshortpl{#1}}%
  }
}
{
  \newcommand*{\Glsfmtshortpl}[1]{%
   \Glsxtrtitleshortpl{#1}}
}
```

`\glsfmtname` As above but for the name value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtname}[1]{%
    \texorpdfstring
    {\glsxtrtitlename{#1}}%
    {\glsentryname{#1}}%
  }
}
{
  \newcommand*{\glsfmtname}[1]{%
   \glsxtrtitlename{#1}}
}
```

`\Glsfmtname` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtname}[1]{%
    \texorpdfstring
    {\Glsxtrtitlename{#1}}%
    {\glsentryname{#1}}%
  }
}
{
  \newcommand*{\Glsfmtname}[1]{%
   \Glsxtrtitlename{#1}}
}
```

`\glsfmttext` As above but for the text value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmttext}[1]{%
    \texorpdfstring
    {\glsxtrtitletext{#1}}%
    {\glsentrytext{#1}}%
  }
}
{
  \newcommand*{\glsfmttext}[1]{%
   \glsxtrtitletext{#1}}
}
```

`\Glsfmttext` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmttext}[1]{%
    \texorpdfstring
    {\Glsxtrtitletext{#1}}%
    {\glsentrytext{#1}}%
  }
}
{
  \newcommand*{\Glsfmttext}[1]{%
   \Glsxtrtitletext{#1}}
}
```

`\glsfmtplural` As above but for the plural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtplural}[1]{%
    \texorpdfstring
    {\glsxtrtitleplural{#1}}%
    {\glsentryplural{#1}}%
  }
}
{
  \newcommand*{\glsfmtplural}[1]{%
   \glsxtrtitleplural{#1}}
}
```

`\Glsfmtplural` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtplural}[1]{%
    \texorpdfstring
    {\Glsxtrtitleplural{#1}}%
    {\glsentryplural{#1}}%
  }
}
{
  \newcommand*{\Glsfmtplural}[1]{%
   \Glsxtrtitleplural{#1}}
}
```

`\glsfmtfirst` As above but for the first value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfirst}[1]{%
    \texorpdfstring
    {\glsxtrtitlefirst{#1}}%
    {\glsentryfirst{#1}}%
  }
}
{
  \newcommand*{\glsfmtfirst}[1]{%
   \glsxtrtitlefirst{#1}}
}
```

`\Glsfmtfirst` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfirst}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefirst{#1}}%
    {\glsentryfirst{#1}}%
  }
}
{
  \newcommand*{\Glsfmtfirst}[1]{%
   \Glsxtrtitlefirst{#1}}
}
```

`\glsfmtfirstpl` As above but for the firstplural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfirstpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlefirstplural{#1}}%
    {\glsentryfirstplural{#1}}%
  }
}
{
  \newcommand*{\glsfmtfirstpl}[1]{%
   \glsxtrtitlefirstplural{#1}}
}
```

`\Glsfmtfirstpl` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfirstpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefirstplural{#1}}%
    {\glsentryfirstplural{#1}}%
  }
}
{
  \newcommand*{\Glsfmtfirstpl}[1]{%
   \Glsxtrtitlefirstplural{#1}}
}
```

`\glsfmtlong` As above but for the long value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtlong}[1]{%
    \texorpdfstring
    {\glsxtrtitlelong{#1}}%
    {\glsentrylong{#1}}%
  }
}
{
  \newcommand*{\glsfmtlong}[1]{%
   \glsxtrtitlelong{#1}}
}
```

`\Glsfmtlong` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtlong}[1]{%
    \texorpdfstring
    {\Glsxtrtitlelong{#1}}%
    {\glsentrylong{#1}}%
  }
}
{
  \newcommand*{\Glsfmtlong}[1]{%
   \Glsxtrtitlelong{#1}}
}
```

`\glsfmtlongpl` As above but for the longplural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtlongpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlelongpl{#1}}%
    {\glsentrylongpl{#1}}%
  }
}
{
  \newcommand*{\glsfmtlongpl}[1]{%
   \glsxtrtitlelongpl{#1}}
}
```

`\Glsfmtlongpl` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtlongpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitlelongpl{#1}}%
    {\glsentrylongpl{#1}}%
  }
}
{
  \newcommand*{\Glsfmtlongpl}[1]{%
   \Glsxtrtitlelongpl{#1}}
}
```

`\glsfmtfull` In-line full format.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfull}[1]{%
    \texorpdfstring
    {\glsxtrtitlefull{#1}}%
    {\glsxtrinlinefullformat{#1}{}}%
  }
}
{
  \newcommand*{\glsfmtfull}[1]{%
   \glsxtrtitlefull{#1}}
}
```

`\Glsfmtfull` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfull}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefull{#1}}%
    {\Glsxtrinlinefullformat{#1}{}}%
  }
}
{
  \newcommand*{\Glsfmtfull}[1]{%
   \Glsxtrtitlefull{#1}}
}
```

`\glsfmtfullpl` In-line full plural format.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfullpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlefullpl{#1}}%
    {\glsxtrinlinefullplformat{#1}{}}%
  }
}
{
  \newcommand*{\glsfmtfullpl}[1]{%
   \glsxtrtitlefullpl{#1}}
}
```

`\Glsfmtfullpl` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfullpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefullpl{#1}}%
    {\Glsxtrinlinefullplformat{#1}{}}%
  }
}
{
  \newcommand*{\Glsfmtfullpl}[1]{%
   \Glsxtrtitlefullpl{#1}}
}
```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```
\newcommand*{\RequireGlossariesExtraLang}[1]{%
```

```
  \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
}
```

sariesExtraLang

```
\newcommand*{\ProvidesGlossariesExtraLang}[1]{%
  \ProvidesFile{glossariesxtr-#1.ldf}%
}
```

Load any required language modules that are available. This doesn't generate any warning if none are found, since they're not essential. (The only command that really needs defining for the document is `\abbreviationsname`, which can simply be redefined. However, with `bib2gls` it might be useful to provide custom rules for a particular locale.)

xtr@loaddialect The dialect label should be stored in `\this@dialect` before using this command.

```
\newcommand{\glsxtr@loaddialect}{%
  \IfTrackedLanguageFileExists{\this@dialect}%
  {glossariesxtr-}% prefix
  {.ldf}%
  {%
    \RequireGlossariesExtraLang{\CurrentTrackedTag}%
  }%
  {}% not found
```

If glossaries-extra-bib2gls has been loaded, `\@glsxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```
  \@glsxtrdialecthook
}
```

```
\@ifpackageloaded{tracklang}
{%
  \AnyTrackedLanguages
  {%
    \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
  }%
  {}%
}
{}
```

Load glossaries-extra-stylemods if required.

```
\@glsxtr@redefstyles
```

and set the style:

```
\@glsxtr@do@style
```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the record option.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra-bib2gls}[2018/12/01 1.38 (NLCT)]
```

These are some convenient macros for use with custom rules.

\glshex

```
\newcommand*{\glshex}{\string\u}
```

lscapturedgroup

```
\newcommand*{\glscapturedgroup}{\string\$}
```

nZeroChildCount For use with bib2gls's save-child-count resource option.

```
\newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
  \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
}
```

rprovidecommand For use in @preamble, this behaves like \providecommand in the document but like \renewcommand in bib2gls.

```
\newcommand*{\glsxtrprovidecommand}{\providecommand}
```

glsrenewcommand Like \renewcommand but only generates a warning rather than an error if the command isn't defined.

```
\newcommand*{\glsrenewcommand}{\@star@or@long\glsxtr@renewcommand}
```

tr@renewcommand

```
\newcommand*{\glsxtr@renewcommand}[1]{%
 \begingroup \escapechar\m@ne\xdef\@gtempa{{\string#1}}\endgroup
 \expandafter\@ifundefined\@gtempa
   {%
     \GlossariesExtraWarning{can't redefine \noexpand#1(not already defined)}%
   }%
   \relax
  \relax
 \let\@ifdefinable\@rc@ifdefinable
 \new@command#1%
}
```

lossarylocation For use with indexcounter and bib2gls.

```
\newcommand*{\glsxtr@wrglossarylocation}[2]{#1}
```

IndexCounterLink

\GlsXtrIndexCounterLink{⟨*text*⟩}{⟨*label*⟩}

For use with indexcounter and bib2gls.

```
\ifdef\hyperref
{%
  \newcommand*{\GlsXtrIndexCounterLink}[2]{%
```

```
    \glsxtrifhasfield{indexcounter}{#2}%
    {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
    {#1}%
  }
}
{
  \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
}
```

`\GlsXtrDualField`

`\GlsXtrDualField`

The internal field used to store the dual label. The `dual-field` defaults to `dual` if no value is supplied so that's used as the default.

```
\newcommand*{\GlsXtrDualField}{dual}
```

`sXtrDualBackLink`

`\GlsXtrDualBackLink{⟨text⟩}{⟨label⟩}`

Adds a hyperlink to the dual entry.

```
\newcommand*{\GlsXtrDualBackLink}[2]{%
  \glsxtrifhasfield{\GlsXtrDualField}{#2}%
  {\glshyperlink[#1]{\glscurrentfieldvalue}}%
  {#2}%
}
```

`TeXEntryAliases` Convenient shortcut for use with `entry-type-aliases` to alias standard BIBTEX entry types to `@bibtexentry`.

```
\newcommand*{\GlsXtrBibTeXEntryAliases}{%
  article=bibtexentry,
  book=bibtexentry,
  booklet=bibtexentry,
  conference=bibtexentry,
  inbook=bibtexentry,
  incollection=bibtexentry,
  inproceedings=bibtexentry,
  manual=bibtexentry,
  mastersthesis=bibtexentry,
  misc=bibtexentry,
  phdthesis=bibtexentry,
  proceedings=bibtexentry,
  techreport=bibtexentry,
  unpublished=bibtexentry
}
```

`ideBibTeXFields` Convenient shortcut to define the standard BIBTEX fields.

```
\newcommand*{\GlsXtrProvideBibTeXFields}{%
  \glsaddstoragekey{address}{}{\glsxtrbibaddress}%
  \glsaddstoragekey{author}{}{\glsxtrbibauthor}%
  \glsaddstoragekey{booktitle}{}{\glsxtrbibbooktitle}%
  \glsaddstoragekey{chapter}{}{\glsxtrbibchapter}%
  \glsaddstoragekey{edition}{}{\glsxtrbibedition}%
  \glsaddstoragekey{howpublished}{}{\glsxtrbibhowpublished}%
  \glsaddstoragekey{institution}{}{\glsxtrbibinstitution}%
  \glsaddstoragekey{journal}{}{\glsxtrbibjournal}%
  \glsaddstoragekey{month}{}{\glsxtrbibmonth}%
  \glsaddstoragekey{note}{}{\glsxtrbibnote}%
  \glsaddstoragekey{number}{}{\glsxtrbibnumber}%
  \glsaddstoragekey{organization}{}{\glsxtrbiborganization}%
  \glsaddstoragekey{pages}{}{\glsxtrbibpages}%
  \glsaddstoragekey{publisher}{}{\glsxtrbibpublisher}%
  \glsaddstoragekey{school}{}{\glsxtrbibschool}%
  \glsaddstoragekey{series}{}{\glsxtrbibseries}%
  \glsaddstoragekey{title}{}{\glsxtrbibtitle}%
  \glsaddstoragekey{bibtextype}{}{\glsxtrbibtype}%
  \glsaddstoragekey{volume}{}{\glsxtrbibvolume}%
}
```

Multiple supplementary references are only supported with `bib2gls`.

`ltisupplocation` This is like `\glsxtrsupphypernumber` but the second argument is the external file name (which isn't obtained from the externallocation attribute). The third argument is the formatting (encap) control sequence *name*. This is ignored by default, but is set by `bib2gls` to the original encap in case it's required.

```
\newcommand*{\glsxtrmultisupplocation}[3]{%
 {%
   \def\glsxtrsupplocationurl{#2}%
   \glshypernumber{#1}%
 }%
}
```

`trdisplaysupploc`

```
\glsxtrdisplaysupploc{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨src⟩}{⟨location⟩}
```

This is like `\glsnoidxdisplayloc` but is used for supplementary locations and so requires an extra argument.

```
\newcommand*\glsxtrdisplaysupploc[5]{%
  \setentrycounter[#1]{#2}%
  \glsxtrmultisupplocation{#5}{#4}{#3}%
}
```

splaylocnameref `\glsxtrdisplaylocnameref{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}{⟨name⟩}{⟨href⟩}{⟨hcounter⟩}{⟨external file⟩}` Used with the [nameref]record package option. The ⟨*href*⟩ argument was obtained from `\@currentHref` and the ⟨*hcounter*⟩ argument was obtained from `\theHentrycounter`, which is more reliable. If hyperref hasn't been loaded, this just behaves like `\glsnoidxdisplayloc`.

```
\ifundef\hyperlink
{
  \newcommand*{\glsxtrdisplaylocnameref}[8]{%
    \glsnoidxdisplayloc{#1}{#2}{#3}{#4}%
  }
}
{
```

Default action uses ⟨*hcounter*⟩. Equations and pages typically don't have a title, so check the counter name.

```
  \newcommand*{\glsxtrdisplaylocnameref}[8]{%
    \ifstrequal{#2}{equation}%
    {\glsxtrnamereflink{#3}{(#4)}{#2.#7}{#8}}%
    {%
      \ifstrempty{#5}%
      {%
```

No title, so just use the location as the link text.

```
        \glsxtrnamereflink{#3}{#4}{#2.#7}{#8}%
      }%
      {%
        \ifstrequal{#2}{page}%
        {\glsxtrnamereflink{#3}{#4}{#2.#7}{#8}}%
        {\glsxtrnamereflink{#3}{#5}{#2.#7}{#8}}%
      }%
    }%
  }
}
```

lsxtrnamereflink `\glsxtrfmtnamereflink{⟨format⟩}{⟨title⟩}{⟨href⟩}{⟨external file⟩}`

```
\newcommand*{\glsxtrnamereflink}[4]{%
```

Locally change `\glshypernumber` to `\@firstofone` to remove the normal location hyperlink.

```
  \begingroup
    \let\glshypernumber\@firstofone
```

If the ⟨*external file*⟩ argument is empty, an internal link is used, otherwise an external one is needed.

```
    \ifstrempty{#4}%
    {\glsxtrfmtinternalnameref{#3}{#1}{#2}}%
    {\glsxtrfmtexternalnameref{#3}{#1}{#2}{#4}}%
  \endgroup
}
```

lsxtrnameloclink

`\glsxtrnamerefloclink{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}{⟨text⟩}{⟨external file⟩}`

Like `\@gls@numberlink`, this creates a hyperlink to the target obtained from the prefix, counter and location but uses ⟨*text*⟩ as the hyperlink text. As with regular indexing, this will fail if the target name can't be formed by prefixing the location value.

```
\newcommand{\glsxtrnameloclink}[6]{%
 \begingroup
  \setentrycounter[#1]{#2}%
  \def\glsxtr@locationhypertext{#5}%
  \let\glshypernumber\@firstofone
  \def\@glsnumberformat{#3}%
  \def\glsxtrsupplocationurl{#6}%
  \toks@={}%
  \@glsxtr@bibgls@removespaces#4 \@nil
 \endgroup
}
```

ls@removespaces

```
\def\@glsxtr@bibgls@removespaces#1 #2\@nil{%
 \toks@=\expandafter{\the\toks@#1}%
 \ifx\\#2\\%
   \edef\x{\the\toks@}%
   \ifx\x\empty
   \else
     \protected@edef\x{\glsentrycounter\@glo@counterprefix\the\toks@}%
     \ifdefvoid\glsxtrsupplocationurl
     {%
       \expandafter\glsxtrfmtinternalnameref\expandafter{\x}%
       {\@glsnumberformat}{\glsxtr@locationhypertext}%
     }%
     {%
       \expandafter\glsxtrfmtexternalnameref\expandafter{\x}%
       {\@glsnumberformat}{\glsxtr@locationhypertext}{\glsxtrsupplocationurl}%
     }%
   \fi
 \else
   \@gls@ReturnAfterFi{%
     \@glsxtr@bibgls@removespaces#2\@nil
   }%
 \fi
}
```

tinternalnameref

`\glsxtrfmtinternalnameloc{⟨target⟩}{⟨format⟩}{⟨title⟩}`

```
\newcommand*{\glsxtrfmtinternalnameref}[3]{%
  \csuse{#2}{\glsdohyperlink{#1}{#3}}%
}
```

texternalnameref

`\glsxtrfmtexternalnameloc{⟨target⟩}{⟨format⟩}{⟨title⟩}{⟨file⟩}`

```
\newcommand*{\glsxtrfmtexternalnameref}[4]{%
  \csuse{#2}{\hyperref{#4}{}{#1}{#3}}%
}
```

\glsxtrSetWidest

`\glsxtrSetWidest{⟨type⟩}{⟨level⟩}{⟨text⟩}`

As from bib2gls v1.8, this is used by the `set-widest` resource option for the alttree and the styles provided by the glossary-longextra package.

```
\newcommand*{\glsxtrSetWidest}[3]{%
```

Check which style options have been provided. (The style packages may not have been loaded.)

```
  \ifdef\glsupdatewidest
  {%
    \ifdef\glslongextraUpdateWidest
    {%
```

Relevant style packages all loaded. If the ⟨*type*⟩ has been given, append to glossary preamble.

```
      \ifstrempty{#1}
      {%
        \glsupdatewidest[#2]{#3}%
        \ifnum#2=0\relax
          \glslongextraUpdateWidest{#3}%
        \else
          \glslongextraUpdateWidestChild{#2}{#3}%
        \fi
      }%
      {%
        \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%
        \ifnum#2=0\relax
          \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
        \else
          \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
        \fi
```

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

Only alttree.

```
    \ifstrempty{#1}
    {%
      \glsupdatewidest[#2]{#3}%
    }%
    {%
      \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%
    }%
  }%
}%
{%
```

`\glsupdatewidest` hasn't been defined. This could just mean that the glossaries-extra-stylemods package hasn't been loaded.

```
  \ifdef\glssetwidest
  {%
    \ifdef\glslongextraUpdateWidest
    {%
```

Relevant glossary-tree and glossary-longextra have been loaded. If the ⟨*type*⟩ has been given, append to glossary preamble.

```
      \ifstrempty{#1}
      {%
        \glssetwidest[#2]{#3}%
        \ifnum#2=0\relax
          \glslongextraUpdateWidest{#3}%
        \else
          \glslongextraUpdateWidestChild{#2}{#3}%
        \fi
      }%
      {%
        \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
        \ifnum#2=0\relax
          \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
        \else
          \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
        \fi
      }%
    }%
    {%
```

Only alttree.

```
      \ifstrempty{#1}
      {%
        \glssetwidest[#2]{#3}%
      }%
      {%
```

```
          \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
        }%
      }%
    }%
    {%
      \ifdef\glslongextraUpdateWidest
      {%
```

glossary-longextra has been loaded.

```
        \ifstrempty{#1}
        {%
          \ifnum#2=0\relax
            \glslongextraUpdateWidest{#3}%
          \else
            \glslongextraUpdateWidestChild{#2}{#3}%
          \fi
        }%
        {%
          \ifnum#2=0\relax
            \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
          \else
            \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
          \fi
        }%
      }%
```

Neither glossary-tree nor glossary-longextra have been loaded. Do nothing.

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

etWidestFallback

`\glsxtrSetWidestFallback{⟨max depth⟩}{⟨list⟩}`

Used when bib2gls can't determine the widest name. The ⟨*list*⟩ argument is a comma-separated list of glossary labels. The ⟨*max depth*⟩ refers to the maximum hierarchical depth. This will either be 0 (only top-level entries) or 2 (up to two child-levels).

```
\newcommand*{\glsxtrSetWidestFallback}[2]{%
  \ifnum#1=0\relax
   \ifdef\glsFindWidestTopLevelName
   {%
     \glsFindWidestTopLevelName[#2]%
   }%
   {%
     \GlossariesExtraWarning{You need stylemods={tree} to
       provide a fallback for set-widest}%
   }%
  \else
   \ifdef\glsFindWidestLevelTwo
   {%
     \glsFindWidestLevelTwo[#2]%
     \ifdef\glslongextraUpdateWidestChild
     {%
      \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnamei}}%
      \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnameii}}%
     }%
     {}%
   }%
   {%
     \GlossariesExtraWarning{You need stylemods={tree} to
       provide a fallback for set-widest}%
   }%
  \fi
}
```

`r@labelprefixes` List of label prefixes.

```
\newcommand*{\@glsxtr@labelprefixes}{}
```

`arlabelprefixes` List of label prefixes.

```
\newcommand*{\glsxtrclearlabelprefixes}{%
  \renewcommand*{\@glsxtr@labelprefixes}{}%
}
```

`raddlabelprefix` Add prefix to the list. These should be added in the order of precedence with the last one as a fallback. This doesn't check against duplicates as it may be useful to replicate a prefix at the end as the fallback.

```
\newcommand*{\glsxtraddlabelprefix}[1]{%
  \ifstrempty{#1}%
  {\glsxtraddlabelprefix{\empty}}%
  {%
    \ifdefempty\@glsxtr@labelprefixes
    {\def\@glsxtr@labelprefixes{#1}}%
    {\appto\@glsxtr@labelprefixes{,#1}}%
  }%
}
```

`pendlabelprefix` Inserts at the start of the list.

```
\newcommand*{\glsxtrprependlabelprefix}[1]{%
  \ifstrempty{#1}%
  {\glsxtrprependlabelprefix{\empty}}%
  {%
    \ifdefempty\@glsxtr@labelprefixes
    {\def\@glsxtr@labelprefixes{#1}}%
    {\preto\@glsxtr@labelprefixes{#1,}}%
  }%
}
```

nlabelprefixlist

```
\glsxtrifinlabelprefixlist{⟨prefix⟩}{⟨true⟩}{⟨false⟩}
```

Test if the given prefix is in the list.

```
\newcommand*{\glsxtrifinlabelprefixlist}[3]{%
  \ifstrempty{#1}%
  {\glsxtrifinlabelprefixlist{\empty}{#2}{#3}}%
  {%
    \DTLifinlist{#1}{\@glsxtr@labelprefixes}{#2}{#3}%
  }%
}
```

prefixlabellist This is provided for the benefit of bib2gls. It's possible that the user may add more prefixes after the start of the document, but that can lead to inconsistencies. The final element of the list (the fallback) is the only prefix of interest for bib2gls.

```
\AtBeginDocument{%
 \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@prefixlabellist}[1]{}}%
 \protected@write\@auxout{}{\string\@glsxtr@prefixlabellist{\@glsxtr@labelprefixes}}%
}
```

t@prefixedlabel Iterate through all the prefixes and find the first prefix and label combination that exists. If none found, this could mean that it's the first LaTeX run, so the last prefix in the list needs to be the fallback one. Grouping is used in case of a nested for loop.

```
\newcommand*{\@glsxtr@get@prefixedlabel}[1]{%
 \begingroup
```

Initialise to the unprefixed label in the event that the list is empty.

```
 \edef\@gls@thislabel{#1}%
 \@for\@glsxtr@prefix:=\@glsxtr@labelprefixes\do
 {%
   \edef\@gls@thislabel{\@glsxtr@prefix#1}%
   \ifglsentryexists{\@gls@thislabel}{\@endfortrue}{}%
 }%
 \edef\x{\endgroup\noexpand\def\noexpand\@gls@thislabel{\@gls@thislabel}}\x
}
```

\dgls Like \gls but tries the prefixes. (Can't use \pgls as that's provided by glossaries-prefix.) Since this command is designed for bib2gls's dual entry system, the "d" stands for "dual".

```
\newrobustcmd*{\dgls}{\@gls@hyp@opt\@dgls}
```

\@dgls

```
\newcommand*{\@dgls}[2][]{%
  \@glsxtr@get@prefixedlabel{#2}%
  \new@ifnextchar[{\@gls@{#1}{\@gls@thislabel}}{\@gls@{#1}{\@gls@thislabel}[]}%
}
```

\dglspl

```
\newrobustcmd*{\dglspl}{\@gls@hyp@opt\@dglspl}
```

\@dglspl

```
\newcommand*{\@dglspl}[2][]{%
  \@glsxtr@get@prefixedlabel{#2}%
  \new@ifnextchar[{\@glspl@{#1}{\@gls@thislabel}}{\@glspl@{#1}{\@gls@thislabel}[]}%
}
```

\dGls

```
\newrobustcmd*{\dGls}{\@gls@hyp@opt\@dGls}
```

\@dGls

```
\newcommand*{\@dGls}[2][]{%
  \@glsxtr@get@prefixedlabel{#2}%
  \new@ifnextchar[{\@Gls@{#1}{\@gls@thislabel}}{\@Gls@{#1}{\@gls@thislabel}[]}%
}
```

\dGlspl

```
\newrobustcmd*{\dGlspl}{\@gls@hyp@opt\@dGlspl}
```

\@dGlspl

```
\newcommand*{\@dGlspl}[2][]{%
  \@glsxtr@get@prefixedlabel{#2}%
  \new@ifnextchar[{\@Glspl@{#1}{\@gls@thislabel}}{\@Glspl@{#1}{\@gls@thislabel}[]}%
}
```

\dGLS

```
\newrobustcmd*{\dGLS}{\@gls@hyp@opt\@dGLS}
```

\@dGLS

```
\newcommand*{\@dGLS}[2][]{%
  \@glsxtr@get@prefixedlabel{#2}%
  \new@ifnextchar[{\@GLS@{#1}{\@gls@thislabel}}{\@GLS@{#1}{\@gls@thislabel}[]}%
}
```

\dGLSpl

```
\newrobustcmd*{\dGLSpl}{\@gls@hyp@opt\@dGLSpl}
```

\@dGLSpl

```
\newcommand*{\@dGLSpl}[2][]{%
  \@glsxtr@get@prefixedlabel{#2}%
  \new@ifnextchar[{\@GLSpl@{#1}{\@gls@thislabel}}{\@GLSpl@{#1}{\@gls@thislabel}[]}%
}
```

`\dglslink` Like `\glslink` but tries the prefixes.

```
\newrobustcmd*{\dglslink}[3][]{%
  \@glsxtr@get@prefixedlabel{#2}%
  \glslink[#1]{\@gls@thislabel}{#3}%
}
```

`\dglsdisp` Like `\glsdisp` but tries the prefixes.

```
\newrobustcmd*{\dglsdisp}[3][]{%
  \@glsxtr@get@prefixedlabel{#2}%
  \glsdisp[#1]{\@gls@thislabel}{#3}%
}
```

Provide missing Greek letters for use in maths mode. These commands are recognised by `bib2gls` and will be mapped to the Mathematical Greek Italic letters. This ensures that the Greek letters that have the same shape as Latin letters are kept with the other mathematical Greek letters for sorting purposes. The LaTeX version of these commands (provided here) use an upright font for capitals and italic for lower case to provide a better match with the other Greek symbols provided by the kernel.

`\Alpha`

```
\providecommand*{\Alpha}{\mathrm{A}}
```

`\Beta`

```
\providecommand*{\Beta}{\mathrm{B}}
```

`\Epsilon`

```
\providecommand*{\Epsilon}{\mathrm{E}}
```

`\Zeta`

```
\providecommand*{\Zeta}{\mathrm{Z}}
```

`\Eta`

```
\providecommand*{\Eta}{\mathrm{H}}
```

`\Iota`

```
\providecommand*{\Iota}{\mathrm{I}}
```

`\Kappa`

```
\providecommand*{\Kappa}{\mathrm{K}}
```

`\Mu`

```
\providecommand*{\Mu}{\mathrm{M}}
```

`\Nu`

```
\providecommand*{\Nu}{\mathrm{N}}
```

`\Omicron`

```
\providecommand*{\Omicron}{\mathrm{O}}
```

`\Rho`

```
\providecommand*{\Rho}{\mathrm{P}}
```

`\Tau`

```
\providecommand*{\Tau}{\mathrm{T}}
```

`\Chi`

```
\providecommand*{\Chi}{\mathrm{X}}
```

`\Digamma`

```
\providecommand*{\Digamma}{\mathrm{F}}
```

`\omicron`

```
\providecommand*{\omicron}{\mathit{o}}
```

Provide corresponding upright characters if upgreek has been loaded. (The upper case characters are the same as above.)

```
\@ifpackageloaded{upgreek}%
{
```

`\Upalpha`

```
  \providecommand*{\Upalpha}{\mathrm{A}}
```

`\Upbeta`

```
  \providecommand*{\Upbeta}{\mathrm{B}}
```

`\Upepsilon`

```
  \providecommand*{\Upepsilon}{\mathrm{E}}
```

`\Upzeta`

```
  \providecommand*{\Upzeta}{\mathrm{Z}}
```

`\Upeta`

```
  \providecommand*{\Upeta}{\mathrm{H}}
```

`\Upiota`

```
  \providecommand*{\Upiota}{\mathrm{I}}
```

`\Upkappa`

```
  \providecommand*{\Upkappa}{\mathrm{K}}
```

`\Upmu`

```
  \providecommand*{\Upmu}{\mathrm{M}}
```

`\Upnu`

```
  \providecommand*{\Upnu}{\mathrm{N}}
```

`\Upomicron`

```
  \providecommand*{\Upomicron}{\mathrm{O}}
```

\Uprho

```
\providecommand*{\Uprho}{\mathrm{P}}
```

\Uptau

```
\providecommand*{\Uptau}{\mathrm{T}}
```

\Upchi

```
\providecommand*{\Upchi}{\mathrm{X}}
```

\upomicron

```
\providecommand*{\upomicron}{\mathrm{o}}
```

```
}%
{}% upgreek.sty not loaded
```

This package provides some basic rules, but it's not intended for complete coverage of all locales. The CLDR should provide the appropriate locale-sensitive rules. These macros are primarily to help construct custom rules to include, for example, Greek maths symbols mixed with Latin. For the full rule syntax, see the Java API for RuleBaseCollator

If you want to provide a rule-block for a particular locale to allow for customization within that locale, create a file called `glossariesxtr-`⟨*tag*⟩`.ldf` (where ⟨*tag*⟩ identifies the locale) and add similar commands. See the description of `\IfTrackedLanguageFileExists` in the tracklang manual for the allowed forms of ⟨*tag*⟩. The simplest is to just use the root language label or ISO code. The file will then be automatically loaded by glossaries-extra if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```
sort-rule={\glsxtrcontrolrules
 ;\glsxtrspacerules
 ;\glsxtrnonprintablerules
 ;\glsxtrcombiningdiacriticrules
 ,\glsxtrhyphenrules
 <\glsxtrgeneralpuncrules
 <\glsxtrdigitrules
 <\glsxtrfractionrules
 <\glsxtrGeneralLatinIVrules
 <\glsxtrMathItalicGreekIrules
}
```

xtrcontrolrules These are control characters that are usually placed at the start of a rule in the 'ignored characters' section. These control characters are unlikely to appear in any entry fields but are provided for completeness. `\string` is used for punctuation characters in case they've been made active.

```
\newcommand*{\glsxtrcontrolrules}{%
 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
0010\string'\string=\glshex 0011
 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
}
```

`lsxtrspacerules` These are space characters.

```
\newcommand*{\glsxtrspacerules}{%
 \string' \string'\string;
 \string'\glshex 00A0\string'\string;
 \string'\glshex 2000\string'\string;
 \string'\glshex 2001\string'\string;
 \string'\glshex 2002\string'\string;
 \string'\glshex 2003\string'\string;
 \string'\glshex 2004\string'\string;
 \string'\glshex 2005\string'\string;
 \string'\glshex 2006\string'\string;
 \string'\glshex 2007\string'\string;
 \string'\glshex 2008\string'\string;
 \string'\glshex 2009\string'\string;
 \string'\glshex 200A\string'\string;
 \string'\glshex 3000\string'
}
```

`nprintablerules` These are non-printable characters (BOM, tabs, line feed and carriage return).

```
\newcommand*{\glsxtrnonprintablerules}{%
 \string'\glshex FEFF\string'\string;
 \string'\glshex 000A\string'\string;
 \string'\glshex 0009\string'\string;
 \string'\glshex 000C\string'\string;
 \string'\glshex 000B\string'
}
```

`gdiacriticrules` Combining diacritic marks. This is split into multiple macros.

```
\newcommand*{\glsxtrcombiningdiacriticrules}{%
 \glsxtrcombiningdiacriticIrules\string;
 \glsxtrcombiningdiacriticIIrules\string;
 \glsxtrcombiningdiacriticIIIrules\string;
 \glsxtrcombiningdiacriticIVrules
}
```

diacriticIrules First set of combining diacritic marks.

```
\newcommand*{\glsxtrcombiningdiacriticIrules}{%
 \glshex 0301\string;% combining acute
 \glshex 0300\string;% combining grave
 \glshex 0306\string;% combining breve
 \glshex 0302\string;% combining circumflex
 \glshex 030C\string;% combining caron
 \glshex 030A\string;% combining ring
 \glshex 030D\string;% combining vertical line above
 \glshex 0308\string;% combining diaeresis
 \glshex 030B\string;% combining double acute
 \glshex 0303\string;% combining tilde
 \glshex 0307\string;% combining dot above
 \glshex 0304% combining macron
}
```

iacriticIIrules Second set of combining diacritic marks.

```
\newcommand*{\glsxtrcombiningdiacriticIIrules}{%
 \glshex 0337\string;% combining short solidus overlay
 \glshex 0327\string;% combining cedilla
 \glshex 0328\string;% combining ogonek
 \glshex 0323\string;% combining dot below
 \glshex 0332\string;% combining low line
 \glshex 0305\string;% combining overline
 \glshex 0309\string;% combining hook above
 \glshex 030E\string;% combining double vertical line above
 \glshex 030F\string;% combining double grave accent
 \glshex 0310\string;% combining candrabindu
 \glshex 0311\string;% combining inverted breve
 \glshex 0312\string;% combining turned comma above
 \glshex 0313\string;% combining comma above
 \glshex 0314\string;% combining reversed comma above
 \glshex 0315\string;% combining comma above right
 \glshex 0316\string;% combining grave accent below
 \glshex 0317% combining acute accent below
}
```

acriticIIIrules Third set of combining diacritic marks.

```
\newcommand*{\glsxtrcombiningdiacriticIIIrules}{%
 \glshex 0318\string;% combining left tack below
 \glshex 0319\string;% combining right tack below
 \glshex 031A\string;% combining left angle above
 \glshex 031B\string;% combining horn
 \glshex 031C\string;% combining left half ring below
 \glshex 031D\string;% combining up tack below
 \glshex 031E\string;% combining down tack below
 \glshex 031F\string;% combining plus sign below
 \glshex 0320\string;% combining minus sign below
 \glshex 0321\string;% combining palatalized hook below
 \glshex 0322\string;% combining retroflex hook below
 \glshex 0324\string;% combining diaresis below
 \glshex 0325\string;% combining ring below
 \glshex 0326\string;% combining comma below
 \glshex 0329\string;% combining vertical line below
 \glshex 032A\string;% combining bridge below
 \glshex 032B\string;% combining inverted double arch below
 \glshex 032C\string;% combining caron below
 \glshex 032D\string;% combining circumflex accent below
 \glshex 032E\string;% combining breve below
 \glshex 032F\string;% combining inverted breve below
 \glshex 0330\string;% combining tilde below
 \glshex 0331\string;% combining macron below
 \glshex 0333\string;% combining double low line
 \glshex 0334\string;% combining tilde overlay
 \glshex 0335\string;% combining short stroke overlay
 \glshex 0336\string;% combining long stroke overlay
 \glshex 0338\string;% combining long solidus overlay
 \glshex 0339\string;% combining combining right half ring below
 \glshex 033A\string;% combining inverted bridge below
 \glshex 033B\string;% combining square below
 \glshex 033C\string;% combining seagull below
 \glshex 033D\string;% combining x above
 \glshex 033E\string;% combining vertical tilde
 \glshex 033F\string;% combining double overline
 \glshex 0342\string;% combining Greek perispomeni
 \glshex 0344\string;% combining Greek dialytika tonos
 \glshex 0345\string;% combining Greek ypogegrammeni
 \glshex 0360\string;% combining double tilde
 \glshex 0361\string;% combining double inverted breve
 \glshex 0483\string;% combining Cyrillic titlo
 \glshex 0484\string;% combining Cyrillic palatalization
 \glshex 0485\string;% combining Cyrillic dasia pneumata
 \glshex 0486% combining Cyrillic psili pneumata
}
```

`iacriticIVrules` Fourth set of combining diacritic marks.

```
\newcommand*{\glsxtrcombiningdiacriticIVrules}{%
 \glshex 20D0\string;% combining left harpoon above
 \glshex 20D1\string;% combining right harpoon above
 \glshex 20D2\string;% combining long vertical line overlay
 \glshex 20D3\string;% combining short vertical line overlay
 \glshex 20D4\string;% combining anticlockwise arrow above
 \glshex 20D5\string;% combining clockwise arrow above
 \glshex 20D6\string;% combining left arrow above
 \glshex 20D7\string;% combining right arrow above
 \glshex 20D8\string;% combining ring overlay
 \glshex 20D9\string;% combining clockwise ring overlay
 \glshex 20DA\string;% combining anticlockwise ring overlay
 \glshex 20DB\string;% combining three dots above
 \glshex 20DC\string;% combining four dots above
 \glshex 20DD\string;% combining enclosing circle
 \glshex 20DE\string;% combining enclosing square
 \glshex 20DF\string;% combining enclosing diamond
 \glshex 20E0\string;% combining enclosing circle backslash
 \glshex 20E1%  combining left right arrow above
}
```

`sxtrhyphenrules` Hyphens.

```
\newcommand*{\glsxtrhyphenrules}{%
 \string’\string-\string’\string;% ASCII hyphen
 \glshex 00AD\string;% soft hyphen
 \glshex 2010\string;% hyphen
 \glshex 2011\string;% non-breaking hyphen
 \glshex 2012\string;% figure dash
 \glshex 2013\string;% en dash
 \glshex 2014\string;% em dash
 \glshex 2015\string;% horizontal bar
 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
}
```

`eneralpuncrules` General punctuation.

```
\newcommand*{\glsxtrgeneralpuncrules}{%
  \glsxtrgeneralpuncIrules
  \string<\glsxtrcurrencyrules
  \string<\glsxtrgeneralpuncIIrules
}
```

`neralpuncIrules` First set of general punctuation.

```
\newcommand*{\glsxtrgeneralpuncIrules}{%
 \string’\glshex 005F\string’% underscore
 \string<\glshex 00AF% macron
 \string<\string’\glshex 002C\string’% comma
 \string<\string’\glshex 003B\string’% semi-colon
 \string<\string’\glshex 003A\string’% colon
 \string<\string’\glshex 0021\string’% exclamation mark
 \string<\glshex 00A1% inverted exclamation mark
 \string<\string’\glshex 003F\string’% question mark
 \string<\glshex 00BF% inverted question mark
 \string<\string’\glshex 002F\string’% solidus
 \string<\string’\glshex 002E\string’% full stop
 \string<\glshex 00B4% acute accent
 \string<\string’\glshex 0060\string’% grave accent
\string<\string'\glshex 005E\string'% circumflex accent
\string<\glshex 00A8% diaersis
\string<\string'\glshex 007E\string'% tilde
\string<\glshex 00B7% middle dot
\string<\glshex 00B8% cedilla
\string<\string'\glshex 0027\string'% straight apostrophe
\string<\string'\glshex 0022\string'% straight double quote
\string<\glshex 00AB% left guillemet
\string<\glshex 00BB% right guillemet
\string<\string'\glshex 0028\string'% left parenthesis
 \string=\glshex 207D\string=\glshex 208D% super/subscript left parenthesis
\string<\string'\glshex 0029\string'% right parenthesis
 \string=\glshex 207E\string=\glshex 208E% super/subscript right parenthesis
\string<\string'\glshex 005B\string'% left square bracket
\string<\string'\glshex 005D\string'% right square bracket
\string<\string'\glshex 007B\string'% left curly bracket
\string<\string'\glshex 007D\string'% right curly bracket
\string<\glshex 00A7% section sign
\string<\glshex 00B6% pilcrow sign
\string<\glshex 00A9% copyright sign
\string<\glshex 00AE% registered sign
\string<\string'\glshex 0040\string'% at sign
}
```

`trcurrencyrules` General punctuation.

```
\newcommand*{\glsxtrcurrencyrules}{%
\glshex 00A4% currency sign
\string<\glshex 0E3F% Thai currency symbol baht
\string<\glshex 00A2% cent sign
\string<\glshex 20A1% colon sign
\string<\glshex 20A2% cruzeiro sign
\string<\string'\glshex 0024\string'% dollar sign
\string<\glshex 20AB% dong sign
\string<\glshex 20AC% euro sign
\string<\glshex 20A3% French franc sign
\string<\glshex 20A4% lira sign
\string<\glshex 20A5% mill sign
\string<\glshex 20A6% naira sign
\string<\glshex 20A7% peseta sign
\string<\glshex 00A3% pound sign
\string<\glshex 20A8% rupee sign
\string<\glshex 20AA% new sheqel sign
\string<\glshex 20A9% won sign
\string<\glshex 00A5% yen sign
}
```

`eralpuncIIrules` Second set of general punctuation.

```
\newcommand*{\glsxtrgeneralpuncIIrules}{%
\string'\glshex 002A\string'% asterisk
\string<\string'\glshex 005C\string'% backslash
\string<\string'\glshex 0026\string'% ampersand
\string<\string'\glshex 0023\string'% hash sign
\string<\string'\glshex 0025\string'% percent sign
\string<\string'\glshex 002B\string'% plus sign
 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
\string<\glshex 00B1% plus-minus sign
\string<\glshex 00F7% division sign
\string<\glshex 00D7% multiplication sign
\string<\string'\glshex 003C\string'% less-than sign
\string<\string'\glshex 003D\string'% equals sign
\string<\string'\glshex 003E\string'% greater-than sign
\string<\glshex 00AC% not sign
\string<\string'\glshex 007C\string'% vertical bar (pipe)
\string<\glshex 00A6% broken bar
\string<\glshex 00B0% degree sign
\string<\glshex 00B5% micron sign
}
```

`eralLatinIrules` Basic Latin alphabet.

```
\newcommand*{\glsxtrGeneralLatinIrules}{%
\glsxtrLatinA
\string<b,B%
\string<c,C%
\string<d,D%
\string<\glsxtrLatinE
\string<f,F%
\string<g,G%
\string<\glsxtrLatinH
\string<\glsxtrLatinI
\string<j,J%
\string<\glsxtrLatinK
\string<\glsxtrLatinL
\string<\glsxtrLatinM
\string<\glsxtrLatinN
\string<\glsxtrLatinO
\string<\glsxtrLatinP
\string<q,Q%
\string<r,R%
\string<\glsxtrLatinS
\string<\glsxtrLatinT
\string<u,U%
\string<v,V%
\string<w,W%
\string<\glsxtrLatinX
\string<y,Y%
\string<z,Z
}
```

ralLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```
\newcommand*{\glsxtrGeneralLatinIIrules}{%
 \glsxtrLatinA
 \string<b,B%
 \string<c,C%
 \string<d,D%
 \string<\glsxtrLatinEth
 \string<\glsxtrLatinE
 \string<f,F%
 \string<g,G%
 \string<\glsxtrLatinH
 \string<\glsxtrLatinI
 \string<j,J%
 \string<\glsxtrLatinK
 \string<\glsxtrLatinL
 \string<\glsxtrLatinM
 \string<\glsxtrLatinN
 \string<\glsxtrLatinO
 \string<\glsxtrLatinP
 \string<q,Q%
 \string<r,R%
 \string<\glsxtrLatinS
 \string& SS \string, \glsxtrLatinEszettSs
 \string<\glsxtrLatinT
 \string<u,U%
 \string<v,V%
 \string<w,W%
 \string<\glsxtrLatinX
 \string<y,Y%
 \string<z,Z%
}
```

alLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```
\newcommand*{\glsxtrGeneralLatinIIIrules}{%
 \glsxtrLatinA
 \string<b,B%
 \string<c,C%
 \string<d,D%
 \string<\glsxtrLatinEth
 \string<\glsxtrLatinE
 \string<f,F%
 \string<g,G%
 \string<\glsxtrLatinH
 \string<\glsxtrLatinI
 \string<j,J%
 \string<\glsxtrLatinK
 \string<\glsxtrLatinL
 \string<\glsxtrLatinM
 \string<\glsxtrLatinN
 \string<\glsxtrLatinO
 \string<\glsxtrLatinP
 \string<q,Q%
 \string<r,R%
 \string<\glsxtrLatinS
 \string& SZ, \glsxtrLatinEszettSz
 \string<\glsxtrLatinT
 \string<u,U%
 \string<v,V%
 \string<w,W%
 \string<\glsxtrLatinX
 \string<y,Y%
 \string<z,Z%
}
```

ralLatinIVrules General Latin alphabet (Æ treated as AE and Œtreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```
\newcommand*{\glsxtrGeneralLatinIVrules}{%
\glsxtrLatinA
\string& AE , \glsxtrLatinAELigature
\string<b,B%
\string<c,C%
\string<d,D%
\string<\glsxtrLatinEth
\string<\glsxtrLatinE
\string<f,F%
\string<g,G%
\string<\glsxtrLatinH
\string<\glsxtrLatinI
\string<j,J%
\string<\glsxtrLatinK
\string<\glsxtrLatinL
\string<\glsxtrLatinM
\string<\glsxtrLatinN
\string<\glsxtrLatinO
\string& OE , \glsxtrLatinOELigature
\string<\glsxtrLatinP
\string<q,Q%
\string<r,R%
\string<\glsxtrLatinS
\string& SS , \glsxtrLatinEszettSs
\string<\glsxtrLatinT
\string& th =\glshex 00DE
\string& TH =\glshex 00FE
\string<u,U%
\string<v,V%
\string<w,W%
\string<\glsxtrLatinX
\string<y,Y%
```

```
 \string<z,Z%
}
```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```
\newcommand*{\glsxtrGeneralLatinVrules}{%
 \glsxtrLatinA
 \string<b,B%
 \string<c,C%
 \string<d,D%
 \string<\glsxtrLatinEth
 \string<\glsxtrLatinE
 \string<f,F%
 \string<g,G%
 \string<\glsxtrLatinH
 \string<\glsxtrLatinI
 \string<j,J%
 \string<\glsxtrLatinK
 \string<\glsxtrLatinL
 \string<\glsxtrLatinM
 \string<\glsxtrLatinN
 \string<\glsxtrLatinO
 \string<\glsxtrLatinP
 \string<q,Q%
 \string<r,R%
 \string<\glsxtrLatinS
 \string& SS , \glsxtrLatinEszettSs
 \string<\glsxtrLatinT
 \string& th =\glshex 00DE
 \string& TH =\glshex 00FE
 \string<u,U%
 \string<v,V%
 \string<w,W%
 \string<\glsxtrLatinX
 \string<y,Y%
 \string<z,Z%
}
```

ralLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```
\newcommand*{\glsxtrGeneralLatinVIrules}{%
 \glsxtrLatinA
 \string<b,B%
 \string<c,C%
 \string<d,D%
 \string<\glsxtrLatinEth
 \string<\glsxtrLatinE
 \string<f,F%
 \string<g,G%
 \string<\glsxtrLatinH
 \string<\glsxtrLatinI
 \string<j,J%
 \string<\glsxtrLatinK
 \string<\glsxtrLatinL
 \string<\glsxtrLatinM
 \string<\glsxtrLatinN
 \string<\glsxtrLatinO
 \string<\glsxtrLatinP
 \string<q,Q%
 \string<r,R%
 \string<\glsxtrLatinS
 \string& SZ , \glsxtrLatinEszettSz
 \string<\glsxtrLatinT
 \string& th =\glshex 00DE
 \string& TH =\glshex 00FE
 \string<u,U%
 \string<v,V%
 \string<w,W%
 \string<\glsxtrLatinX
 \string<y,Y%
 \string<z,Z%
}
```

`alLatinVIIrules` General Latin alphabet (Æ between A and B, eth between D and E, insular G as G, Œ between O and P, long S equivalent to S, Þ between T and U and wynn as W).

```
\newcommand*{\glsxtrGeneralLatinVIIrules}{%
\glsxtrLatinA
\string<\glsxtrLatinAELigature
\string<b,B%
\string<c,C%
\string<d,D%
\string<\glsxtrLatinEth
\string<\glsxtrLatinE
\string<f,F%
\string<\glsxtrLatinInsularG
\string<\glsxtrLatinH
\string<\glsxtrLatinI
\string<j,J%
\string<\glsxtrLatinK
\string<\glsxtrLatinL
\string<\glsxtrLatinM
\string<\glsxtrLatinN
\string<\glsxtrLatinO
\string<\glsxtrLatinOELigature
\string<\glsxtrLatinP
\string<q,Q%
\string<r,R%
\string<\glshex 017F=\glsxtrLatinS % s and long s
\string<\glsxtrLatinT
\string<\glsxtrLatinThorn
\string<u,U%
\string<v,V%
\string< w\string=\glshex 01BF, W\string=\glshex 01F7
\string<\glsxtrLatinX
\string<y,Y%
\string<z,Z%
}
```

lLatinVIIIrules General Latin alphabet (Æ treated as AE and Œtreated as OE, Þtreated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).

```
\newcommand*{\glsxtrGeneralLatinVIIIrules}{%
\glsxtrLatinA
\string& AE , \glsxtrLatinAELigature
\string<b,B%
\string<c,C%
\string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
\string<\glsxtrLatinE
\string<f,F%
\string<g,G%
\string<\glsxtrLatinH
\string<\glsxtrLatinI
\string<j,J%
\string<\glsxtrLatinK
\string<\glshex 0142\string=\glsxtrLatinL\string=\glshex 0141% L and \L
\string<\glsxtrLatinM
\string<\glsxtrLatinN
\string<\glshex 00F8\string=\glsxtrLatinO\string=\glshex 00D8% O and \O
\string& OE , \glsxtrLatinOELigature
\string<\glsxtrLatinP
\string<q,Q%
\string<r,R%
\string<\glsxtrLatinS
\string& SS , \glsxtrLatinEszettSs
\string<\glsxtrLatinT
\string& th =\glshex 00DE
\string& TH =\glshex 00FE
\string<u,U%
\string<v,V%
\string<w,W%
\string<\glsxtrLatinX
\string<y,Y%
\string<z,Z%
}
```

\glsxtrLatinA

```
\newcommand*{\glsxtrLatinA}{%
  a\string=\glshex 00AA\string=\glshex 2090,A
}
```

\glsxtrLatinE

```
\newcommand*{\glsxtrLatinE}{%
  e\string=\glshex 2091,E
}
```

\glsxtrLatinH

```
\newcommand*{\glsxtrLatinH}{%
  h\string=\glshex 2095,H
}
```

\glsxtrLatinI

```
\newcommand*{\glsxtrLatinI}{%
  i\string=\glshex 2071,I
}
```

\glsxtrLatinK

```
\newcommand*{\glsxtrLatinK}{%
  k\string=\glshex 2096,K
}
```

\glsxtrLatinL

```
\newcommand*{\glsxtrLatinL}{%
  l\string=\glshex 2097,L
}
```

\glsxtrLatinM

```
\newcommand*{\glsxtrLatinM}{%
  m\string=\glshex 2098,M
}
```

\glsxtrLatinN

```
\newcommand*{\glsxtrLatinN}{%
  n\string=\glshex 207F\string=\glshex 2099,N
}
```

\glsxtrLatinO

```
\newcommand*{\glsxtrLatinO}{%
  o\string=\glshex 00BA\string=\glshex 2092,O
}
```

\glsxtrLatinP

```
\newcommand*{\glsxtrLatinP}{%
  p\string=\glshex 209A,P
}
```

\glsxtrLatinS

```
\newcommand*{\glsxtrLatinS}{%
  s\string=\glshex 209B,S
}
```

\glsxtrLatinT

```
\newcommand*{\glsxtrLatinT}{%
  t\string=\glshex 209C,T
}
```

\glsxtrLatinX

```
\newcommand*{\glsxtrLatinX}{%
  x\string=\glshex 2093,X
}
```

lsxtrLatinSchwa Latin schwa (lower case, subscript and upper case).

```
\newcommand*{\glsxtrLatinSchwa}{%
  \glshex 0259\string=\glshex 2094,\glshex 018F
}
```

trLatinEszettSs

```
\newcommand*{\glsxtrLatinEszettSs}{%
 \glshex 00DF% eszett
 \string=\glshex 017Fs % long S s
}
```

trLatinEszettSz

```
\newcommand*{\glsxtrLatinEszettSz}{%
 \glshex 00DF% eszett
 \string= \glshex 017Fz % long S z
}
```

\glsxtrLatinEth

```
\newcommand*{\glsxtrLatinEth}{%
 \glshex 00F0,\glshex 00D0% eth
}
```

lsxtrLatinThorn

```
\newcommand*{\glsxtrLatinThorn}{%
 \glshex 00FE,\glshex 00DE% thorn
}
```

LatinAELigature

```
\newcommand*{\glsxtrLatinAELigature}{%
 \glshex 00E6,\glshex 00C6% AE-ligature
}
```

LatinOELigature

```
\newcommand*{\glsxtrLatinOELigature}{%
 \glshex 0153,\glshex 0152% OE-ligature
}
```

\glsxtrLatinAA

```
\newcommand*{\glsxtrLatinAA}{%
 \glshex 00E5=a\glshex 030A,% \aa
 \glshex 00C5=A\glshex 030A% \AA
}
```

glsxtrLatinWynn

```
\newcommand*{\glsxtrLatinWynn}{%
 \glshex 01BF,\glshex 01F7% wynn
}
```

trLatinInsularG

```
\newcommand*{\glsxtrLatinInsularG}{%
 \glshex 1D79,\glshex A77D% insular G
 \string; g, G
}
```

sxtrLatinOslash

```
\newcommand*{\glsxtrLatinOslash}{%
 \glshex 00F8,\glshex 00D8% \o, \O
}
```

sxtrLatinLslash

```
\newcommand*{\glsxtrLatinLslash}{%
 \glshex 0142,\glshex 0141% \l, \L
}
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
\newcommand*{\glsxtrMathUpGreekIrules}{%
 \glsxtrUpAlpha
 \string<\glsxtrUpBeta
 \string<\glsxtrUpGamma
 \string<\glsxtrUpDelta
 \string<\glsxtrUpEpsilon
 \string<\glsxtrUpDigamma
 \string<\glsxtrUpZeta
 \string<\glsxtrUpEta
 \string<\glsxtrUpTheta
 \string<\glsxtrUpIota
 \string<\glsxtrUpKappa
 \string<\glsxtrUpLambda
 \string<\glsxtrUpMu
 \string<\glsxtrUpNu
 \string<\glsxtrUpXi
 \string<\glsxtrUpOmicron
 \string<\glsxtrUpPi
 \string<\glsxtrUpRho
 \string<\glsxtrUpSigma
 \string<\glsxtrUpTau
 \string<\glsxtrUpUpsilon
 \string<\glsxtrUpPhi
 \string<\glsxtrUpChi
 \string<\glsxtrUpPsi
 \string<\glsxtrUpOmega
}
```

hUpGreekIIrules Doesn’t include digamma.

```
\newcommand*{\glsxtrMathUpGreekIIrules}{%
 \glsxtrUpAlpha
 \string<\glsxtrUpBeta
 \string<\glsxtrUpGamma
 \string<\glsxtrUpDelta
 \string<\glsxtrUpEpsilon
 \string<\glsxtrUpZeta
 \string<\glsxtrUpEta
 \string<\glsxtrUpTheta
 \string<\glsxtrUpIota
 \string<\glsxtrUpKappa
 \string<\glsxtrUpLambda
 \string<\glsxtrUpMu
 \string<\glsxtrUpNu
 \string<\glsxtrUpXi
 \string<\glsxtrUpOmicron
 \string<\glsxtrUpPi
 \string<\glsxtrUpRho
 \string<\glsxtrUpSigma
 \string<\glsxtrUpTau
 \string<\glsxtrUpUpsilon
 \string<\glsxtrUpPhi
 \string<\glsxtrUpChi
 \string<\glsxtrUpPsi
 \string<\glsxtrUpOmega
}
```

alicGreekIrules Includes (upright) digamma between epsilon and zeta (there isn’t an italic digamma), so don’t mix with `\glsxtrMathUpGreekIrules` or there may be unexpected results.

```
\newcommand*{\glsxtrMathItalicGreekIrules}{%
 \glsxtrMathItalicAlpha
 \string<\glsxtrMathItalicBeta
 \string<\glsxtrMathItalicGamma
 \string<\glsxtrMathItalicDelta
 \string<\glsxtrMathItalicEpsilon
 \string<\glsxtrUpDigamma
 \string<\glsxtrMathItalicZeta
 \string<\glsxtrMathItalicEta
 \string<\glsxtrMathItalicTheta
 \string<\glsxtrMathItalicIota
 \string<\glsxtrMathItalicKappa
 \string<\glsxtrMathItalicLambda
 \string<\glsxtrMathItalicMu
 \string<\glsxtrMathItalicNu
 \string<\glsxtrMathItalicXi
 \string<\glsxtrMathItalicOmicron
 \string<\glsxtrMathItalicPi
 \string<\glsxtrMathItalicRho
 \string<\glsxtrMathItalicSigma
 \string<\glsxtrMathItalicTau
 \string<\glsxtrMathItalicUpsilon
 \string<\glsxtrMathItalicPhi
 \string<\glsxtrMathItalicChi
 \string<\glsxtrMathItalicPsi
 \string<\glsxtrMathItalicOmega
}
```

`licGreekIIrules` Doesn’t include digamma.

```
\newcommand*{\glsxtrMathItalicGreekIIrules}{%
 \glsxtrMathItalicAlpha
 \string<\glsxtrMathItalicBeta
 \string<\glsxtrMathItalicGamma
 \string<\glsxtrMathItalicDelta
 \string<\glsxtrMathItalicEpsilon
 \string<\glsxtrMathItalicZeta
 \string<\glsxtrMathItalicEta
 \string<\glsxtrMathItalicTheta
 \string<\glsxtrMathItalicIota
 \string<\glsxtrMathItalicKappa
 \string<\glsxtrMathItalicLambda
 \string<\glsxtrMathItalicMu
 \string<\glsxtrMathItalicNu
 \string<\glsxtrMathItalicXi
 \string<\glsxtrMathItalicOmicron
 \string<\glsxtrMathItalicPi
 \string<\glsxtrMathItalicRho
 \string<\glsxtrMathItalicSigma
 \string<\glsxtrMathItalicTau
 \string<\glsxtrMathItalicUpsilon
 \string<\glsxtrMathItalicPhi
 \string<\glsxtrMathItalicChi
 \string<\glsxtrMathItalicPsi
 \string<\glsxtrMathItalicOmega
}
```

`pperGreekIrules` Upper case only (includes upright digamma).

```
\newcommand*{\glsxtrMathItalicUpperGreekIrules}{%
 \glshex 1D6E2% upper case alpha (maths italic)
 \string<\glshex 1D6E3% upper case beta (maths italic)
 \string<\glshex 1D6E4% upper case gamma (maths italic)
 \string<\glshex 1D6E5% upper case delta (maths italic)
 \string<\glshex 1D6E6% upper case epsilon (maths italic)
 \string<\glshex 03DC% upper case digamma
 \string<\glshex 1D6E7% upper case zeta (maths italic)
 \string<\glshex 1D6E8% upper case eta (maths italic)
 \string<\glshex 1D6E9% upper case theta (maths italic)
 \string=\glshex 1D6F3% upper case theta variant (maths italic)
 \string<\glshex 1D6EA% upper case iota (maths italic)
 \string<\glshex 1D6EB% upper case kappa (maths italic)
 \string<\glshex 1D6EC% upper case lambda (maths italic)
 \string<\glshex 1D6ED% upper case mu (maths italic)
 \string<\glshex 1D6EE% upper case nu (maths italic)
 \string<\glshex 1D6EF% upper case xi (maths italic)
 \string<\glshex 1D6F0% upper case omicron (maths italic)
 \string<\glshex 1D6F1% upper case pi (maths italic)
 \string<\glshex 1D6F2% upper case rho (maths italic)
 \string<\glshex 1D6F4% upper case sigma (maths italic)
 \string<\glshex 1D6F5% upper case tau (maths italic)
 \string<\glshex 1D6F6% upper case upsilon (maths italic)
 \string<\glshex 1D6F7% upper case phi (maths italic)
 \string<\glshex 1D6F8% upper case chi (maths italic)
 \string<\glshex 1D6F9% upper case psi (maths italic)
 \string<\glshex 1D6FA% upper case omega (maths italic)
}
```

`perGreekIIrules` Upper case only (doesn't include upright digamma).

```
\newcommand*{\glsxtrMathItalicUpperGreekIIrules}{%
 \glshex 1D6E2% upper case alpha (maths italic)
 \string<\glshex 1D6E3% upper case beta (maths italic)
 \string<\glshex 1D6E4% upper case gamma (maths italic)
 \string<\glshex 1D6E5% upper case delta (maths italic)
 \string<\glshex 1D6E6% upper case epsilon (maths italic)
 \string<\glshex 1D6E7% upper case zeta (maths italic)
 \string<\glshex 1D6E8% upper case eta (maths italic)
 \string<\glshex 1D6E9% upper case theta (maths italic)
 \string=\glshex 1D6F3% upper case theta variant (maths italic)
 \string<\glshex 1D6EA% upper case iota (maths italic)
 \string<\glshex 1D6EB% upper case kappa (maths italic)
 \string<\glshex 1D6EC% upper case lambda (maths italic)
 \string<\glshex 1D6ED% upper case mu (maths italic)
 \string<\glshex 1D6EE% upper case nu (maths italic)
 \string<\glshex 1D6EF% upper case xi (maths italic)
 \string<\glshex 1D6F0% upper case omicron (maths italic)
 \string<\glshex 1D6F1% upper case pi (maths italic)
 \string<\glshex 1D6F2% upper case rho (maths italic)
 \string<\glshex 1D6F4% upper case sigma (maths italic)
 \string<\glshex 1D6F5% upper case tau (maths italic)
 \string<\glshex 1D6F6% upper case upsilon (maths italic)
 \string<\glshex 1D6F7% upper case phi (maths italic)
 \string<\glshex 1D6F8% upper case chi (maths italic)
 \string<\glshex 1D6F9% upper case psi (maths italic)
 \string<\glshex 1D6FA% upper case omega (maths italic)
}
```

`owerGreekIrules` Lower case only (includes upright digamma).

```
\newcommand*{\glsxtrMathItalicLowerGreekIrules}{%
 \glshex 1D6FC% lower case alpha (maths italic)
 \string<\glshex 1D6FD% lower case beta (maths italic)
 \string<\glshex 1D6FE% lower case gamma (maths italic)
 \string<\glshex 1D6FF% lower case delta (maths italic)
 \string<\glshex 1D700% lower case epsilon (maths italic)
 \string=\glshex 1D716% lower case epsilon variant (maths italic)
 \string<\glshex 03DD% lower case digamma
 \string<\glshex 1D701% lower case zeta (maths italic)
 \string<\glshex 1D702% lower case eta (maths italic)
 \string<\glshex 1D703% lower case theta (maths italic)
 \string=\glshex 1D717% lower case theta variant (maths italic)
 \string<\glshex 1D704% lower case iota (maths italic)
 \string<\glshex 1D705% lower case kappa (maths italic)
 \string=\glshex 1D718% lower case kappa variant (maths italic)
 \string<\glshex 1D706% lower case lambda (maths italic)
 \string<\glshex 1D707% lower case mu (maths italic)
 \string<\glshex 1D708% lower case nu (maths italic)
 \string<\glshex 1D709% lower case xi (maths italic)
 \string<\glshex 1D70A% lower case omicron (maths italic)
 \string<\glshex 1D70B% lower case pi (maths italic)
 \string=\glshex 1D71B% lower case pi variant (maths italic)
 \string<\glshex 1D70C% lower case rho (maths italic)
 \string=\glshex 1D71A% lower case rho variant (maths italic)
 \string<\glshex 1D70D% lower case final sigma (maths italic)
 \string=\glshex 1D70E% lower case sigma (maths italic)
 \string<\glshex 1D70F% lower case tau (maths italic)
 \string<\glshex 1D710% lower case upsilon (maths italic)
 \string<\glshex 1D711% lower case phi (maths italic)
 \string=\glshex 1D719% lower case phi variant (maths italic)
 \string<\glshex 1D712% lower case chi (maths italic)
 \string<\glshex 1D713% lower case psi (maths italic)
 \string<\glshex 1D714% lower case omega (maths italic)
}
```

`werGreekIIrules` Lower case only (doesn’t includes upright digamma).

```
\newcommand*{\glsxtrMathItalicLowerGreekIIrules}{%
 \glshex 1D6FC% lower case alpha (maths italic)
 \string<\glshex 1D6FD% lower case beta (maths italic)
 \string<\glshex 1D6FE% lower case gamma (maths italic)
 \string<\glshex 1D6FF% lower case delta (maths italic)
 \string<\glshex 1D700% lower case epsilon (maths italic)
\string=\glshex 1D716% lower case epsilon variant (maths italic)
\string<\glshex 1D701% lower case zeta (maths italic)
\string<\glshex 1D702% lower case eta (maths italic)
\string<\glshex 1D703% lower case theta (maths italic)
\string=\glshex 1D717% lower case theta variant (maths italic)
\string<\glshex 1D704% lower case iota (maths italic)
\string<\glshex 1D705% lower case kappa (maths italic)
\string=\glshex 1D718% lower case kappa variant (maths italic)
\string<\glshex 1D706% lower case lambda (maths italic)
\string<\glshex 1D707% lower case mu (maths italic)
\string<\glshex 1D708% lower case nu (maths italic)
\string<\glshex 1D709% lower case xi (maths italic)
\string<\glshex 1D70A% lower case omicron (maths italic)
\string<\glshex 1D70B% lower case pi (maths italic)
\string=\glshex 1D71B% lower case pi variant (maths italic)
\string<\glshex 1D70C% lower case rho (maths italic)
\string=\glshex 1D71A% lower case rho variant (maths italic)
\string<\glshex 1D70D% lower case final sigma (maths italic)
\string=\glshex 1D70E% lower case sigma (maths italic)
\string<\glshex 1D70F% lower case tau (maths italic)
\string<\glshex 1D710% lower case upsilon (maths italic)
\string<\glshex 1D711% lower case phi (maths italic)
\string=\glshex 1D719% lower case phi variant (maths italic)
\string<\glshex 1D712% lower case chi (maths italic)
\string<\glshex 1D713% lower case psi (maths italic)
\string<\glshex 1D714% lower case omega (maths italic)
}
```

`MathGreekIrules` Includes both upright and italic with digamma between epsilon and zeta.

```
\newcommand*{\glsxtrMathGreekIrules}{%
\glsxtrMathItalicAlpha
\string;\glsxtrUpAlpha
\string<\glsxtrMathItalicBeta
\string;\glsxtrUpBeta
\string<\glsxtrMathItalicGamma
\string;\glsxtrUpGamma
\string<\glsxtrMathItalicDelta
\string;\glsxtrUpDelta
\string<\glsxtrMathItalicEpsilon
\string;\glsxtrUpEpsilon
\string<\glsxtrUpDigamma
\string<\glsxtrMathItalicZeta
\string;\glsxtrUpZeta
\string<\glsxtrMathItalicEta
\string;\glsxtrUpEta
\string<\glsxtrMathItalicTheta
\string;\glsxtrUpTheta
\string<\glsxtrMathItalicIota
\string;\glsxtrUpIota
```

```
 \string<\glsxtrMathItalicKappa
 \string;\glsxtrUpKappa
 \string<\glsxtrMathItalicLambda
 \string;\glsxtrUpLambda
 \string<\glsxtrMathItalicMu
 \string;\glsxtrUpMu
 \string<\glsxtrMathItalicNu
 \string;\glsxtrUpNu
 \string<\glsxtrMathItalicXi
 \string;\glsxtrUpXi
 \string<\glsxtrMathItalicOmicron
 \string;\glsxtrUpOmicron
 \string<\glsxtrMathItalicPi
 \string;\glsxtrUpPi
 \string<\glsxtrMathItalicRho
 \string;\glsxtrUpRho
 \string<\glsxtrMathItalicSigma
 \string;\glsxtrUpSigma
 \string<\glsxtrMathItalicTau
 \string;\glsxtrUpTau
 \string<\glsxtrMathItalicUpsilon
 \string;\glsxtrUpUpsilon
 \string<\glsxtrMathItalicPhi
 \string;\glsxtrUpPhi
 \string<\glsxtrMathItalicChi
 \string;\glsxtrUpChi
 \string<\glsxtrMathItalicPsi
 \string;\glsxtrUpPsi
 \string<\glsxtrMathItalicOmega
 \string;\glsxtrUpOmega
}
```

`athGreekIIrules` Includes both upright and italic (digamma not included).

```
\newcommand*{\glsxtrMathGreekIIrules}{%
 \glsxtrMathItalicAlpha
 \string;\glsxtrUpAlpha
 \string<\glsxtrMathItalicBeta
 \string;\glsxtrUpBeta
 \string<\glsxtrMathItalicGamma
 \string;\glsxtrUpGamma
 \string<\glsxtrMathItalicDelta
 \string;\glsxtrUpDelta
 \string<\glsxtrMathItalicEpsilon
 \string;\glsxtrUpEpsilon
 \string<\glsxtrMathItalicZeta
 \string;\glsxtrUpZeta
 \string<\glsxtrMathItalicEta
 \string;\glsxtrUpEta
 \string<\glsxtrMathItalicTheta
 \string;\glsxtrUpTheta
 \string<\glsxtrMathItalicIota
 \string;\glsxtrUpIota
 \string<\glsxtrMathItalicKappa
 \string;\glsxtrUpKappa
 \string<\glsxtrMathItalicLambda
 \string;\glsxtrUpLambda
 \string<\glsxtrMathItalicMu
 \string;\glsxtrUpMu
 \string<\glsxtrMathItalicNu
 \string;\glsxtrUpNu
 \string<\glsxtrMathItalicXi
 \string;\glsxtrUpXi
 \string<\glsxtrMathItalicOmicron
 \string;\glsxtrUpOmicron
 \string<\glsxtrMathItalicPi
 \string;\glsxtrUpPi
 \string<\glsxtrMathItalicRho
 \string;\glsxtrUpRho
 \string<\glsxtrMathItalicSigma
 \string;\glsxtrUpSigma
 \string<\glsxtrMathItalicTau
 \string;\glsxtrUpTau
 \string<\glsxtrMathItalicUpsilon
 \string;\glsxtrUpUpsilon
 \string<\glsxtrMathItalicPhi
 \string;\glsxtrUpPhi
 \string<\glsxtrMathItalicChi
 \string;\glsxtrUpChi
 \string<\glsxtrMathItalicPsi
 \string;\glsxtrUpPsi
 \string<\glsxtrMathItalicOmega
 \string;\glsxtrUpOmega
}
```

\glsxtrUpAlpha

```
\newcommand*{\glsxtrUpAlpha}{%
 \glshex 03B1,% lower case alpha
 \glshex 0391% upper case alpha
}
```

\glsxtrUpBeta

```
\newcommand*{\glsxtrUpBeta}{%
 \glshex 03B2,% lower case beta
 \glshex 0392% upper case beta
}
```

\glsxtrUpGamma

```
\newcommand*{\glsxtrUpGamma}{%
```

```
 \glshex 03B3,% lower case gamma
 \glshex 0393% upper case gamma
}
```

\glsxtrUpDelta

```
\newcommand*{\glsxtrUpDelta}{%
 \glshex 03B4,% lower case delta
 \glshex 0394% upper case delta
}
```

glsxtrUpEpsilon

```
\newcommand*{\glsxtrUpEpsilon}{%
 \glshex 03B5% lower case epsilon
 \string=\glshex 03F5,% lower case epsilon variant
 \glshex 0395% upper case epsilon
}
```

glsxtrUpDigamma

```
\newcommand*{\glsxtrUpDigamma}{%
 \glshex 03DD,% lower case digamma
 \glshex 03DC% upper case digamma
}
```

\glsxtrUpZeta

```
\newcommand*{\glsxtrUpZeta}{%
 \glshex 03B6,% lower case zeta
 \glshex 0396% upper case zeta
}
```

\glsxtrUpEta

```
\newcommand*{\glsxtrUpEta}{%
 \glshex 03B7,% lower case eta
 \glshex 0397% upper case eta
}
```

\glsxtrUpTheta

```
\newcommand*{\glsxtrUpTheta}{%
 \glshex 03B8% lower case theta
 \string=\glshex 03D1,% lower case theta variant
 \glshex 0398% upper case theta
}
```

\glsxtrUpIota

```
\newcommand*{\glsxtrUpIota}{%
 \glshex 03B9,% lower case iota
 \glshex 0399% upper case iota
}
```

\glsxtrUpKappa

```
\newcommand*{\glsxtrUpKappa}{%
 \glshex 03BA% lower case kappa
 \string=\glshex 03F0,% lower case kappa variant
 \glshex 039A% upper case kappa
}
```

\glsxtrUpLambda

```
\newcommand*{\glsxtrUpLambda}{%
 \glshex 03BB,% lower lambda
 \glshex 039B% upper case lambda
}
```

\glsxtrUpMu

```
\newcommand*{\glsxtrUpMu}{%
 \glshex 03BC,% lower case mu
 \glshex 039C% upper case mu
}
```

\glsxtrUpNu

```
\newcommand*{\glsxtrUpNu}{%
 \glshex 03BD,% lower case nu
 \glshex 039D% upper case nu
}
```

\glsxtrUpXi

```
\newcommand*{\glsxtrUpXi}{%
 \glshex 03BE,% lower case xi
 \glshex 039E% upper case xi
}
```

glsxtrUpOmicron

```
\newcommand*{\glsxtrUpOmicron}{%
 \glshex 03BF,% lower case omicron
 \glshex 039F% upper case omicron
}
```

\glsxtrUpPi

```
\newcommand*{\glsxtrUpPi}{%
 \glshex 03C0% lower case pi
 \string=\glshex 03D6,% lower case pi variant
 \glshex 03A0% upper case pi
}
```

\glsxtrUpRho

```
\newcommand*{\glsxtrUpRho}{%
 \glshex 03C1% lower case rho
 \string=\glshex 03F1,% lower case rho variant
 \glshex 03A1% upper case rho
}
```

\glsxtrUpSigma

```
\newcommand*{\glsxtrUpSigma}{%
 \glshex 03C2% lower case sigma
 \string=\glshex 03C3,% lower case sigma
 \glshex 03A3% upper case sigma
}
```

\glsxtrUpTau

```
\newcommand*{\glsxtrUpTau}{%
 \glshex 03C4,% lower case tau
 \glshex 03A4% upper case tau
}
```

glsxtrUpUpsilon

```
\newcommand*{\glsxtrUpUpsilon}{%
 \glshex 03C5,% lower case upsilon
 \glshex 03A5% upper case upsilon
}
```

\glsxtrUpPhi

```
\newcommand*{\glsxtrUpPhi}{%
 \glshex 03C6% lower case phi
 \string=\glshex 03D5,% lower case phi variant
 \glshex 03A6% upper case phi
}
```

\glsxtrUpChi

```
\newcommand*{\glsxtrUpChi}{%
 \glshex 03C7,% lower case chi
 \glshex 03A7% upper case chi
}
```

\glsxtrUpPsi

```
\newcommand*{\glsxtrUpPsi}{%
 \glshex 03C8,% lower case psi
 \glshex 03A8% upper case psi
}
```

\glsxtrUpOmega

```
\newcommand*{\glsxtrUpOmega}{%
 \glshex 03C9,% lower case omega
 \glshex 03A9% upper case omega
}
```

MathItalicAlpha

```
\newcommand*{\glsxtrMathItalicAlpha}{%
 \glshex 1D6FC,% lower case alpha (maths italic)
 \glshex 1D6E2% upper case alpha (maths italic)
}
```

rMathItalicBeta

```
\newcommand*{\glsxtrMathItalicBeta}{%
 \glshex 1D6FD,% lower case beta (maths italic)
 \glshex 1D6E3% upper case beta (maths italic)
}
```

MathItalicGamma

```
\newcommand*{\glsxtrMathItalicGamma}{%
 \glshex 1D6FE,% lower case gamma (maths italic)
 \glshex 1D6E4% upper case gamma (maths italic)
}
```

MathItalicDelta

```
\newcommand*{\glsxtrMathItalicDelta}{%
 \glshex 1D6FF,% lower case delta (maths italic)
 \glshex 1D6E5% upper case delta (maths italic)
}
```

thItalicEpsilon

```
\newcommand*{\glsxtrMathItalicEpsilon}{%
 \glshex 1D700% lower case epsilon (maths italic)
 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
 \glshex 1D6E6% upper case epsilon (maths italic)
}
```

rMathItalicZeta

```
\newcommand*{\glsxtrMathItalicZeta}{%
 \glshex 1D701,% lower case zeta (maths italic)
 \glshex 1D6E7% upper case zeta (maths italic)
}
```

trMathItalicEta

```
\newcommand*{\glsxtrMathItalicEta}{%
 \glshex 1D702,% lower case eta (maths italic)
 \glshex 1D6E8% upper case eta (maths italic)
}
```

MathItalicTheta

```
\newcommand*{\glsxtrMathItalicTheta}{%
 \glshex 1D703% lower case theta (maths italic)
 \string=\glshex 1D717,% lower case theta variant (maths italic)
 \glshex 1D6E9% upper case theta (maths italic)
 \string=\glshex 1D6F3% upper case theta variant (maths italic)
}
```

rMathItalicIota

```
\newcommand*{\glsxtrMathItalicIota}{%
 \glshex 1D704,% lower case iota (maths italic)
 \glshex 1D6EA% upper case iota (maths italic)
}
```

MathItalicKappa

```
\newcommand*{\glsxtrMathItalicKappa}{%
 \glshex 1D705% lower case kappa (maths italic)
 \string=\glshex 1D718,% lower case kappa variant (maths italic)
 \glshex 1D6EB% upper case kappa (maths italic)
}
```

athItalicLambda

```
\newcommand*{\glsxtrMathItalicLambda}{%
 \glshex 1D706,% lower case lambda (maths italic)
 \glshex 1D6EC% upper case lambda (maths italic)
}
```

xtrMathItalicMu

```
\newcommand*{\glsxtrMathItalicMu}{%
 \glshex 1D707,% lower case mu (maths italic)
 \glshex 1D6ED% upper case mu (maths italic)
}
```

xtrMathItalicNu

```
\newcommand*{\glsxtrMathItalicNu}{%
 \glshex 1D708,% lower case nu (maths italic)
 \glshex 1D6EE% upper case nu (maths italic)
}
```

xtrMathItalicXi

```
\newcommand*{\glsxtrMathItalicXi}{%
 \glshex 1D709,% lower case xi (maths italic)
 \glshex 1D6EF% upper case xi (maths italic)
}
```

thItalicOmicron

```
\newcommand*{\glsxtrMathItalicOmicron}{%
 \glshex 1D70A,% lower case omicron (maths italic)
 \glshex 1D6F0% upper case omicron (maths italic)
}
```

xtrMathItalicPi

```
\newcommand*{\glsxtrMathItalicPi}{%
```

```
 \glshex 1D70B% lower case pi (maths italic)
 \string=\glshex 1D71B,% lower case pi variant (maths italic)
 \glshex 1D6F1% upper case pi (maths italic)
}
```

trMathItalicRho

```
\newcommand*{\glsxtrMathItalicRho}{%
 \glshex 1D70C% lower case rho (maths italic)
 \string=\glshex 1D71A,% lower case rho variant (maths italic)
 \glshex 1D6F2% upper case rho (maths italic)
}
```

MathItalicSigma

```
\newcommand*{\glsxtrMathItalicSigma}{%
 \glshex 1D70D% lower case final sigma (maths italic)
 \string=\glshex 1D70E,% lower case sigma (maths italic)
 \glshex 1D6F4% upper case sigma (maths italic)
}
```

trMathItalicTau

```
\newcommand*{\glsxtrMathItalicTau}{%
 \glshex 1D70F,% lower case tau (maths italic)
 \glshex 1D6F5% upper case tau (maths italic)
}
```

thItalicUpsilon

```
\newcommand*{\glsxtrMathItalicUpsilon}{%
 \glshex 1D710,% lower case upsilon (maths italic)
 \glshex 1D6F6% upper case upsilon (maths italic)
}
```

trMathItalicPhi

```
\newcommand*{\glsxtrMathItalicPhi}{%
 \glshex 1D711% lower case phi (maths italic)
 \string=\glshex 1D719,% lower case phi variant (maths italic)
 \glshex 1D6F7% upper case phi (maths italic)
}
```

trMathItalicChi

```
\newcommand*{\glsxtrMathItalicChi}{%
 \glshex 1D712,% lower case chi (maths italic)
 \glshex 1D6F8% upper case chi (maths italic)
}
```

trMathItalicPsi

```
\newcommand*{\glsxtrMathItalicPsi}{%
 \glshex 1D713,% lower case psi (maths italic)
 \glshex 1D6F9% upper case psi (maths italic)
}
```

MathItalicOmega

```
\newcommand*{\glsxtrMathItalicOmega}{%
 \glshex 1D714,% lower case omega (maths italic)
 \glshex 1D6FA% upper case omega (maths italic)
}
```

thItalicPartial

```
\newcommand*{\glsxtrMathItalicPartial}{%
 \glshex 1D715% partial differential (maths italic)
}
```

MathItalicNabla

```
\newcommand*{\glsxtrMathItalicNabla}{%
 \glshex 1D6FB% nabla (maths italic)
}
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
\newcommand*{\glsxtrdigitrules}{%
 0\string=\glshex 2080\string=\glshex 2070
 \string<1\string=\glshex 2081\string=\glshex 00B9
 \string<2\string=\glshex 2082\string=\glshex 00B2
 \string<3\string=\glshex 2083\string=\glshex 00B3
 \string<4\string=\glshex 2084\string=\glshex 2074
 \string<5\string=\glshex 2085\string=\glshex 2075
 \string<6\string=\glshex 2086\string=\glshex 2076
 \string<7\string=\glshex 2087\string=\glshex 2077
 \string<8\string=\glshex 2088\string=\glshex 2078
 \string<9\string=\glshex 2089\string=\glshex 2079
}
```

BasicDigitrules Digits from the Basic Latin set.

```
\newcommand*{\glsxtrBasicDigitrules}{%
 0\string<1\string<2\string<3\string<4%
 \string<5\string<6\string<7\string<8\string<9%
}
```

criptDigitrules Subscript digits.

```
\newcommand*{\glsxtrSubScriptDigitrules}{%
 \glshex 2080% subscript 0
 \string<\glshex 2081% subscript 1
 \string<\glshex 2082% subscript 2
 \string<\glshex 2083% subscript 3
 \string<\glshex 2084% subscript 4
 \string<\glshex 2085% subscript 5
 \string<\glshex 2086% subscript 6
 \string<\glshex 2087% subscript 7
 \string<\glshex 2088% subscript 8
 \string<\glshex 2089% subscript 9
}
```

criptDigitrules Superscript digits.

```
\newcommand*{\glsxtrSuperScriptDigitrules}{%
 \glshex 2070% superscript 0
 \string<\glshex 00B9% superscript 1
 \string<\glshex 00B2% superscript 2
 \string<\glshex 00B3% superscript 3
 \string<\glshex 2074% superscript 4
 \string<\glshex 2075% superscript 5
 \string<\glshex 2076% superscript 6
 \string<\glshex 2077% superscript 7
 \string<\glshex 2078% superscript 8
 \string<\glshex 2079% superscript 9
}
```

trfractionrules Vulgar fractions.

```
\newcommand*{\glsxtrfractionrules}{%
 \glshex 215F% fraction numerator one (1/)
 \string<\glshex 2189% zero thirds (0/3 = 0)
 \string<\glshex 2152% one tenth (1/10 = 0.1)
 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
 \string<\glshex 215B% one eighth (1/8 = 0.125)
 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
 \string<\glshex 2155% one fifth (1/5 = 0.2)
 \string<\glshex 00BC% one quarter (1/4 = 0.25)
 \string<\glshex 2153% one third (1/3 ~ 0.333)
 \string<\glshex 215C% three eighths (3/8 = 0.375)
 \string<\glshex 2156% two fifths (2/5 = 0.4)
 \string<\glshex 00BD% one half (1/2 = 0.5)
 \string<\glshex 2157% three fifths (3/5 = 0.6)
 \string<\glshex 215D% five eighths (5/8 = 0.625)
 \string<\glshex 2154% two thirds (2/3 ~ 0.667)
 \string<\glshex 00BE% three quarters (3/4 = 0.75)
 \string<\glshex 2158% four fifths (4/5 = 0.8)
 \string<\glshex 215A% five sixths (5/6 ~ 0.833)
 \string<\glshex 215E% seven eighths (7/8 = 0.875)
}
```

sxtrdialecthook Check for scripts associated with the document dialects.

```
\renewcommand{\@glsxtrdialecthook}{%
  \ifundef\CurrentTrackedScript
  {%
    \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
    {%
     \edef\CurrentTrackedScript{%
       \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
    }%
    {}%
  }%
  {}%
  \ifdef\CurrentTrackedScript
  {%
    \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
    \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
    \let\CurrentTrackedTag\CurrentTrackedScript
    \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
    {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
    {}%
    \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
  }%
  {}%
}
```

If `\glsxtr@loaddialect` has been defined, then glossaries-extra-bib2gls has been loaded after glossaries-extra. (For example, through `\glossariesextrasetup`.) Not recommended, but if this has been done try to find the associated language resources.

```
\ifdef\glsxtr@loaddialect
{%
  \@ifpackageloaded{tracklang}
  {%
    \AnyTrackedLanguages
    {%
      \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
    }%
    {}%
  }
  {}
}
{}
```

2 Style Adjustments (glossaries-extra-stylemods.sty)

This package adjusts the predefined styles so that they include the post description hook. Also, some other minor adjustments may be made to make existing styles more flexible.

2.1 Package Initialisation

First identify package:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra-stylemods}[2018/12/01 1.38 (NLCT)]
```

Provide package options to automatically load required predefined styles. The simplest method is to just test for the existence of the file `glossary-`⟨*option*⟩`.sty`. Packages can't be loaded whilst the options are being processed, so save the list in `\@glsxtr@loadstyles`.

`sxtr@loadstyles`

```
\newcommand*{\@glsxtr@loadstyles}{}
```

`all` Provide all known styles.

```
\DeclareOption{all}{%
  \appto\@glsxtr@loadstyles{%
    \RequirePackage{glossary-inline}%
    \RequirePackage{glossary-list}%
    \RequirePackage{glossary-tree}%
    \RequirePackage{glossary-mcols}%
    \RequirePackage{glossary-long}%
    \RequirePackage{glossary-longragged}%
    \RequirePackage{glossary-longbooktabs}%
    \RequirePackage{glossary-super}%
    \RequirePackage{glossary-superragged}%
    \RequirePackage{glossary-bookindex}%
    \RequirePackage{glossary-longextra}%
  }
}
```

```
\DeclareOption*{%
  \IfFileExists{glossary-\CurrentOption.sty}
   {\eappto\@glsxtr@loadstyles{%
      \noexpand\RequirePackage{glossary-\CurrentOption}}%
   }%
   {%
```

```
      \PackageError{glossaries-extra-styles}%
      {Unknown option ‘\CurrentOption’}{}%
    }%
}
```

Process the package options:

```
\ProcessOptions
```

Load the required packages:

```
\@glsxtr@loadstyles
```

Adjust the styles so that they all have the post description hook. Also, instead of having a hard-coded `\space` before the location, use:

sxtrprelocation

This uses `\providecommand` as the same command is also provided by glossary-bookindex.

```
\providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

ewglossarystyle

```
\providecommand{\renewglossarystyle}[2]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
  }%
  {%
    \csdef{@glsstyle@#1}{#2}%
  }%
}
```

2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the listdotted style need modifying to add this.

```
\ifdef{\@glsstyle@listdotted}
{%
 \renewglossarystyle{listdotted}{%
   \setglossarystyle{list}%
   \renewcommand*{\glossentry}[2]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
      \glsentryitem{##1}%
      \glstarget{##1}{\glossentryname{##1}}%
      \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
      \glossentrydesc{##1}\glspostdescription}%
   \renewcommand*{\subglossentry}[3]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
    \glssubentryitem{##2}%
    \glstarget{##2}{\glossentryname{##2}}%
    \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
    \glossentrydesc{##2}\glspostdescription}%
```

```
  }
}
{%
```

Assume the style isn't required if it hasn't already been defined.

```
}
```

The sublistdotted style doesn't display the description for top-level entries. Sub-level entries use the listdottedstyle.

The other list styles would be easier to adapt if the space before the number list wasn't hard coded.

```
\ifdef{\@glsstyle@list}
{%
```

`listprelocation` Space before number list for top-level entries.

```
  \newcommand{\glslistprelocation}{\glsxtrprelocation}
```

`hildprelocation` Space before number list for child entries.

```
  \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

`ildpostlocation` Full stop after number list.

```
  \newcommand{\glslistchildpostlocation}{.}
```

`\glslistdesc`

```
  \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

Redefine list to use these commands.

```
  \renewglossarystyle{list}{%
    \renewenvironment{theglossary}%
      {\begin{description}}{\end{description}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand*{\glossentry}[2]{%
      \item[\glsentryitem{##1}%
            \glstarget{##1}{\glossentryname{##1}}]
        \glslistdesc{##1}\glslistprelocation ##2}%
    \renewcommand*{\subglossentry}[3]{%
      \glssubentryitem{##2}%
      \glstarget{##2}{\strut}\space
      \glslistdesc{##2}%
      \glslistchildprelocation ##3\glslistchildpostlocation}%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
  }
}
{}
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
\ifdef{\@glsstyle@altlist}
{%
```

```
  \renewglossarystyle{altlist}{%
    \setglossarystyle{list}%
    \renewcommand*{\glossentry}[2]{%
      \item[\glsentryitem{##1}%
        \glstarget{##1}{\glossentryname{##1}}]%
      \mbox{}\par\nobreak\@afterheading
      \glslistdesc{##1}\glslistprelocation ##2}%
    \renewcommand{\subglossentry}[3]{%
      \par
      \glssubentryitem{##2}%
      \glstarget{##2}{\strut}\glslistdesc{##2}%
      \glslistchildprelocation ##3}%
  }
}
{}
```

Redefine listgroup so that it discourages a break after group headings.

```
\ifdef{\@glsstyle@listgroup}
{%
  \renewglossarystyle{listgroup}{%
    \setglossarystyle{list}%
    \renewcommand*{\glsgroupheading}[1]{%
      \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
      \mbox{}\par\nobreak\@afterheading
    }%
  }
}
{}
```

Similarly for listhypergroup.

```
\ifdef{\@glsstyle@listhypergroup}
{%
  \renewglossarystyle{listhypergroup}{%
    \setglossarystyle{list}%
    \renewcommand*{\glossaryheader}{%
      \glslistnavigationitem{\glsnavigation}}%
    \renewcommand*{\glsgroupheading}[1]{%
      \item[\glslistgroupheaderfmt
            {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
      \mbox{}\par\nobreak\@afterheading
    }%
  }
}
{}
```

Similarly for altlistgroup.

```
\ifdef{\@glsstyle@altlistgroup}
{%
  \renewglossarystyle{altlistgroup}{%
    \setglossarystyle{altlist}%
    \renewcommand*{\glsgroupheading}[1]{%
```

```
      \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
      \mbox{}\par\nobreak\@afterheading
    }%
  }
}
{}
```

Similarly for altlisthypergroup.

```
\ifdef{\@glsstyle@altlisthypergroup}
{%
  \renewglossarystyle{altlisthypergroup}{%
    \setglossarystyle{altlist}%
    \renewcommand*{\glossaryheader}{%
      \glslistnavigationitem{\glsnavigation}}%
    \renewcommand*{\glsgroupheading}[1]{%
      \item[\glslistgroupheaderfmt
        {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
      \mbox{}\par\nobreak\@afterheading
    }%
  }
}
{}
```

2.3 Longtable Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glsxtrprelocation`.

```
\ifcsdef{@glsstyle@long}
{%
  \renewglossarystyle{long}{%
    \renewenvironment{theglossary}%
       {\begin{longtable}{lp{\glsdescwidth}}}%
       {\end{longtable}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription
      \glsxtrprelocation ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
       \glsxtrprelocation ##3\tabularnewline
    }%
    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
    \fi
  }
}
{}
```

Three column style:

```
\ifcsdef{@glsstyle@long3col}
{%
  \renewglossarystyle{long3col}{%
    \renewenvironment{theglossary}%
      {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
      {\end{longtable}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
       ##3\tabularnewline
    }%
```

Conditional needs to be outside of `\glsgroupskip` otherwise it can cause “Incomplete `\iftrue`” errors.

```
    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
    \fi
  }
}
{}
```

Four column style:

```
\ifcsdef{@glsstyle@long4col}
{%
  \renewglossarystyle{long4col}{%
    \renewenvironment{theglossary}%
      {\begin{longtable}{llll}}%
      {\end{longtable}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription &
      \glossentrysymbol{##1} &
      ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
       \glossentrysymbol{##2} & ##3\tabularnewline
    }%
    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{& & &\tabularnewline}%
    \fi
  }
}
{}
```

The styles in glossary-longbooktabs are all based on the styles in glossary-long, so no adjustments are needed for that package.

2.4 Long Ragged Styles

The three and four column styles require adjustment for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glsxtrprelocation`.

```
\ifcsdef{@glsstyle@longragged}
{%
  \renewglossarystyle{longragged}{%
    \renewenvironment{theglossary}%
       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
       {\end{longtable}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription\glsxtrprelocation ##2%
      \tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}%
       \glspostdescription\glsxtrprelocation ##3%
       \tabularnewline
    }%
    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
```

```
      \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
    \fi
  }
}
{}
```

Three and four column styles don't use `\glsxtrprelocation` since the number list is in its own column.

```
\ifcsdef{@glsstyle@longragged3col}
{%
  \renewglossarystyle{longragged3col}{%
    \renewenvironment{theglossary}%
      {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
         >{\raggedright}p{\glspagelistwidth}}}%
      {\end{longtable}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
       ##3\tabularnewline
    }%

    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
    \fi
  }
}
{}
```

Four column style:

```
\ifcsdef{@glsstyle@altlongragged4col}
{%
  \renewglossarystyle{altlongragged4col}{%
    \renewenvironment{theglossary}%
      {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
         >{\raggedright}p{\glspagelistwidth}}}%
      {\end{longtable}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
      ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
       \glossentrysymbol{##2} & ##3\tabularnewline
    }%

    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{& & &\tabularnewline}%
    \fi
  }
}
{}
```

2.5 Supertabular Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glsxtrprelocation`.

```
\ifcsdef{@glsstyle@super}
{%
  \renewglossarystyle{super}{%
    \renewenvironment{theglossary}%
      {\tablehead{}\tabletail{}%
       \begin{supertabular}{lp{\glsdescwidth}}}%
      {\end{supertabular}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription
      \glsxtrprelocation ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
       \glsxtrprelocation ##3\tabularnewline
    }%
    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{& \tabularnewline}%
    \fi
  }
}
{}
```

Three column style:

```
\ifcsdef{@glsstyle@super3col}
{%
  \renewglossarystyle{super3col}{%
    \renewenvironment{theglossary}%
      {\tablehead{}\tabletail{}%
       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
      {\end{supertabular}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
       ##3\tabularnewline
    }%

    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
    \fi
  }
}
{}
```

Four column styles:

```
\ifcsdef{@glsstyle@super4col}
{%
  \renewglossarystyle{super4col}{%
    \renewenvironment{theglossary}%
      {\tablehead{}\tabletail{}%
       \begin{supertabular}{llll}}{%
       \end{supertabular}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription &
      \glossentrysymbol{##1} & ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
       \glossentrysymbol{##2} & ##3\tabularnewline
    }%

    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{& & &\tabularnewline}%
    \fi
  }
}
{}
```

2.6 Super Ragged Styles

The three and four column styles require adjustment for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glsxtrprelocation`.

```
\ifcsdef{@glsstyle@superragged}
{%
  \renewglossarystyle{superragged}{%
    \renewenvironment{theglossary}%
      {\tablehead{}\tabletail{}%
       \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
      {\end{supertabular}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription\glsxtrprelocation ##2%
      \tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
       \glsxtrprelocation ##3%
       \tabularnewline
    }%
    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{& \tabularnewline}%
    \fi
  }
}
{}
```

Three column style:

```
\ifcsdef{@glsstyle@superragged3col}
{%
  \renewglossarystyle{superragged3col}{%
    \renewenvironment{theglossary}%
      {\tablehead{}\tabletail{}%
       \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
          >{\raggedright}p{\glspagelistwidth}}}%
      {\end{supertabular}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription &
      ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
       ##3\tabularnewline
    }%

    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
    \fi
  }
}
{}
```

Four columns:

```
\ifcsdef{@glsstyle@altsuperragged4col}
{%
  \renewglossarystyle{altsuperragged4col}{%
    \renewenvironment{theglossary}%
      {\tablehead{}\tabletail{}%
       \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
         >{\raggedright}p{\glspagelistwidth}}}%
      {\end{supertabular}}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand{\glossentry}[2]{%
      \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
      \glossentrydesc{##1}\glspostdescription &
      \glossentrysymbol{##1} & ##2\tabularnewline
    }%
    \renewcommand{\subglossentry}[3]{%
       &
       \glssubentryitem{##2}%
      \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
      \glossentrysymbol{##2} & ##3\tabularnewline
    }%
    \ifglsnogroupskip
      \renewcommand*{\glsgroupskip}{}%
    \else
      \renewcommand*{\glsgroupskip}{& & &\tabularnewline}%
    \fi
  }
}
{}
```

2.7 Inline Style

The inline style is dealt with slightly differently. The `\glspostdescription` hook is actually in `\glspostinline`, which is called at the end of the glossary. The original definition of `\glspostinline` also includes a space, which is unnecessary. Here, instead of redefining the inline style, just redefine `\glspostinline` and `\glsinlinedescformat`.

```
\ifdef{\@glsstyle@inline}
{%
  \renewcommand*{\glspostinline}{.\spacefactor\sfcode`\.}
```

Just use `\glsxtrpostdescription` instead of `\glspostdescription`.

```
  \renewcommand*{\glsinlinedescformat}[3]{%
    \space#1\glsxtrpostdescription}
  \renewcommand*{\glsinlinesubdescformat}[3]{%
    #1\glsxtrpostdescription}
```

The default settings don't show the location lists, so there's no adjustment for `\glsxtrprelocation`.

```
}
{}
```

2.8 Tree Styles

Redefine both `\glstreenamefmt` and `\glstreegroupheaderfmt` in terms of `\glstreedefaultnamefmt` to make it easier to change both at the same time or only change one without affecting the other.

```
\ifdef\glstreenamefmt
{
```

`edefaultnamefmt`

```
  \newcommand{\glstreedefaultnamefmt}[1]{\textbf{#1}}
```

`\glstreenamefmt`

```
  \renewcommand{\glstreenamefmt}[1]{\glstreedefaultnamefmt{#1}}
```

`egroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
\def\glstreegroupheaderfmt#1{\glstreedefaultnamefmt{#1}}
```

`eenavigationfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
\def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
```

```
}
{}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
\ifdef{\@glsstyle@index}
{
```

`treeprelocation` The space before the number list for top-level entries. This is shared by the other tree styles.

```
\newcommand*{\glstreeprelocation}{\glsxtrprelocation}
```

`hildprelocation` The space before the number list for child entries. This is shared by the other tree styles.

```
\newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
\renewglossarystyle{index}{%
  \renewenvironment{theglossary}%
    {\setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}%
     \let\item\glstreeitem
     \let\subitem\glstreesubitem
     \let\subsubitem\glstreesubsubitem
    }%
  {\par}%
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand*{\glossentry}[2]{%
     \item\glsentryitem{##1}%
     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
     \glstreesymbol{##1}%
     \glstreedesc{##1}%
     \glstreeprelocation ##2%
  }%
  \renewcommand{\subglossentry}[3]{%
    \ifcase##1\relax
      \item
    \or
      \subitem
      \glssubentryitem{##2}%
    \else
      \subsubitem
    \fi
    \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
    \glstreechildsymbol{##2}%
```

```
      \glstreechilddesc{##2}%
      \glstreechildprelocation ##3%
    }%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
  }
}
{}
```

The indexgroup style is redefined to discourage a page break after the heading.

```
\ifdef{\@glsstyle@indexgroup}
{%
  \renewglossarystyle{indexgroup}{%
    \setglossarystyle{index}%
    \renewcommand*{\glsgroupheading}[1]{%
      \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
      \nopagebreak\indexspace
      \nobreak\@afterheading
    }%
  }
}
{}
```

Similarly for indexhypergroup.

```
\ifdef{\@glsstyle@indexhypergroup}
{%
  \renewglossarystyle{indexhypergroup}{%
    \setglossarystyle{index}%
    \renewcommand*{\glossaryheader}{%
      \item\glstreenavigationfmt{\glsnavigation}%
       \nobreak\@afterheading\indexspace}%
    \renewcommand*{\glsgroupheading}[1]{%
      \item\glstreegroupheaderfmt
        {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace
      \nobreak\@afterheading}%
  }%
}
{}
```

Adjust tree style to remove hard coded space before number list.

```
\ifdef{\@glsstyle@tree}
{%
```

Provide a command for use with the tree styles that displays the pre-description separator, the description and post-description hook.

`\glstreedesc`

```
  \newcommand{\glstreedesc}[1]{%
    \glstreepredesc\glossentrydesc{#1}\glspostdescription
  }
```

Similarly for the symbol.

\glstreesymbol

```
\newcommand{\glstreesymbol}[1]{%
  \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
}%
```

And for the child entries:

lstreechilddesc

```
\newcommand{\glstreechilddesc}[1]{%
  \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
}%
```

treechildsymbol This just behaves in the same way as the top-level.

```
\newcommand{\glstreechildsymbol}[1]{%
  \glstreesymbol{#1}%
}%

\renewglossarystyle{tree}{%
  \renewenvironment{theglossary}%
    {\setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}}%
    {}%
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand{\glossentry}[2]{%
    \hangindent0pt\relax
    \parindent0pt\relax
    \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
    \glstreesymbol{##1}%
    \glstreedesc{##1}%
    \glstreeprelocation##2\par
  }%
  \renewcommand{\subglossentry}[3]{%
    \hangindent##1\glstreeindent\relax
    \parindent##1\glstreeindent\relax
    \ifnum##1=1\relax
      \glssubentryitem{##2}%
    \fi
    \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
    \glstreechildsymbol{##2}%
    \glstreechilddesc{##2}%
    \glstreechildprelocation ##3\par
  }%
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
}%
}
{}
```

The treegroup style is redefined to discourage a page break after the heading.

```
\ifdef{\@glsstyle@treegroup}
```

```
{%
  \renewglossarystyle{treegroup}{%
    \setglossarystyle{tree}%
    \renewcommand{\glsgroupheading}[1]{\par
      \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par
      \nopagebreak\indexspace\nobreak\@afterheading}%
  }
}
{}
```

Similarly for treehypergroup

```
\ifdef{\@glsstyle@treehypergroup}
{%
  \renewglossarystyle{treehypergroup}{%
    \setglossarystyle{tree}%
    \renewcommand*{\glossaryheader}{%
      \par\noindent\glstreenavigationfmt{\glsnavigation}\par
      \nobreak\@afterheading\indexspace}%
    \renewcommand*{\glsgroupheading}[1]{%
      \par\noindent
      \glstreegroupheaderfmt
        {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
      \nopagebreak\indexspace\nobreak\@afterheading}%
  }
}
{}
```

Adjust treenoname style to remove hard coded space before number list.

```
\ifdef{\@glsstyle@treenoname}
{%
```

Provide a command for use with the treenoname styles that displays the pre-description separator, the description and post-description hook.

streenonamedesc

```
  \newcommand{\glstreenonamedesc}[1]{%
    \glstreepredesc\glossentrydesc{#1}\glspostdescription
  }%
```

Similarly for the symbol.

reenonamesymbol

```
  \newcommand{\glstreenonamesymbol}[1]{%
    \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
  }%
```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```
  \newcommand{\glstreenonamechilddesc}[1]{%
    \glossentrydesc{#1}\glspostdescription
  }%
  \renewglossarystyle{treenoname}{%
    \renewenvironment{theglossary}%
      {\setlength{\parindent}{0pt}%
       \setlength{\parskip}{0pt plus 0.3pt}}%
      {}%
    \renewcommand*{\glossaryheader}{}%
    \renewcommand*{\glsgroupheading}[1]{}%
    \renewcommand{\glossentry}[2]{%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
      \glstreenonamesymbol{##1}%
      \glstreenonamedesc{##1}%
      \glstreeprelocation##2\par
    }%
    \renewcommand{\subglossentry}[3]{%
      \hangindent##1\glstreeindent\relax
      \parindent##1\glstreeindent\relax
      \ifnum##1=1\relax
        \glssubentryitem{##2}%
      \fi
      \glstarget{##2}{\strut}%
      \glstreenonamechilddesc{##2}%
      \glstreechildprelocation##3\par
    }%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
  }
}
{}
```

The treenonamegroup style is redefined to discourage a page break after the heading.

```
\ifdef{\@glsstyle@treenonamegroup}
{%
  \renewglossarystyle{treenonamegroup}{%
    \setglossarystyle{treenoname}%
    \renewcommand{\glsgroupheading}[1]{\par
      \noindent\glstreegroupheaderfmt
        {\glsgetgrouptitle{##1}}%
      \nopagebreak\indexspace\nobreak\@afterheading
    }%
  }
}
{}
```

Similarly for treenonamehypergroup

```
\ifdef{\@glsstyle@treenonamehypergroup}
{%
  \renewglossarystyle{treenonamehypergroup}{%
    \setglossarystyle{treenoname}%
    \renewcommand*{\glossaryheader}{%
```

```
   \par\noindent\glstreenavigationfmt{\glsnavigation}\par
   \nobreak\@afterheading\indexspace}%
  \renewcommand*{\glsgroupheading}[1]{%
   \par\noindent
   \glstreegroupheaderfmt
    {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
   \nopagebreak\indexspace\nobreak\@afterheading}%
 }
}
{}
```

The alttree style is redefined to make it easier to made minor adjustments.

```
\ifdef{\@glsstyle@alttree}
{%
```

Only redefine this style if it's already been defined.

mbolDescLocation `\glsxtralttreeSymbolDescLocation{⟨label⟩}{⟨location list⟩}`

Layout the symbol, description and location for top-level entries.

```
 \newcommand{\glsxtralttreeSymbolDescLocation}[2]{%
  {%
   \let\par\glsxtrAltTreePar
   \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
   \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
  }%
 }
```

trAltTreeIndent Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```
 \newlength\glsxtrAltTreeIndent
```

lsxtrAltTreePar Multi-paragraph descriptions need to keep the hanging indent.

```
 \newcommand{\glsxtrAltTreePar}{%
  \@@par
  \glsxtrAltTreeSetHangIndent
  \setlength{\parindent}{\dimexpr\hangindent+\glsxtrAltTreeIndent}%
 }
```

mbolDescLocation `\glsxtralttreeSubSymbolDescLocation{⟨level⟩}{⟨label⟩}{⟨location list⟩}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```
 \newcommand{\glsxtralttreeSubSymbolDescLocation}[3]{%
  \glsxtralttreeSymbolDescLocation{#2}{#3}%
 }
```

trtreetopindent The original style has to keep computing the width of the name at each entry. This register allows the style to compute it once for the top-level at the start of the glossary.

```
\newlength\glsxtrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
\newcommand*{\glsxtralttreeInit}{%
  \settowidth{\glsxtrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
  \glsxtrAltTreeIndent=\parindent
}
```

\gglssetwidest The original `\glssetwidest` only uses `\def`. This uses `\gdef`.

```
\newcommand*{\gglssetwidest}[2][0]{%
  \csgdef{@glswidestname\romannumeral#1}{#2}%
}
```

\eglssetwidest The original `\glssetwidest` only uses `\def`. This uses `\protected@csedef`.

```
\newcommand*{\eglssetwidest}[2][0]{%
  \protected@csedef{@glswidestname\romannumeral#1}{#2}%
}
```

\xglssetwidest Like the above but uses `\protected@csxdef`.

```
\newcommand*{\xglssetwidest}[2][0]{%
  \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
}
```

glsupdatewidest Only sets if new value is wider than old value.

```
\newcommand*{\glsupdatewidest}[2][0]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {\csdef{@glswidestname\romannumeral#1}{#2}}%
  {%
    \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
    \settowidth{\dimen@ii}{#2}%
    \ifdim\dimen@ii>\dimen@
     \csdef{@glswidestname\romannumeral#1}{#2}%
    \fi
  }%
}
```

glsupdatewidest As above but global definition.

```
\newcommand*{\gglsupdatewidest}[2][0]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {\csgdef{@glswidestname\romannumeral#1}{#2}}%
  {%
    \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
    \settowidth{\dimen@ii}{#2}%
    \ifdim\dimen@ii>\dimen@
     \csgdef{@glswidestname\romannumeral#1}{#2}%
    \fi
  }%
}
```

glsupdatewidest As `\glsupdatewidest` but expands value.

```
\newcommand*{\eglsupdatewidest}[2][0]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
  {%
    \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
    \settowidth{\dimen@ii}{#2}%
    \ifdim\dimen@ii>\dimen@
     \protected@csedef{@glswidestname\romannumeral#1}{#2}%
    \fi
  }%
}
```

glsupdatewidest As above but global.

```
\newcommand*{\xglsupdatewidest}[2][0]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
  {%
    \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
    \settowidth{\dimen@ii}{#2}%
    \ifdim\dimen@ii>\dimen@
     \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
    \fi
  }%
}
```

lsgetwidestname Provide a user-level macro to obtain the widest top-level name.

```
\newcommand*{\glsgetwidestname}{\@glswidestname}
```

etwidestsubname Provide a user-level macro to obtain the widest sub-entry name.

```
\newcommand*{\glsgetwidestsubname}[1]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {\@glswidestname}%
  {\csuse{@glswidestname\romannumeral#1}}%
}
```

estTopLevelName CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```
\let\glsFindWidestTopLevelName\glsfindwidesttoplevelname
```

sedTopLevelName Like `\glsfindwidesttoplevelname` but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```
\newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {%
        \ifglshasparent{\@glo@label}%
        {}%
        {%
          \settowidth{\dimen@}%
           {\glstreenamefmt{\glsentryname{\@glo@label}}}%
          \ifdim\dimen@>\gls@tmplen
            \gls@tmplen=\dimen@
            \eglssetwidest{\glsentryname{\@glo@label}}%
          \fi
        }%
      }%
      {}%
    }%
  }%
}
```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```
\newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
      }%
      {}%
    }%
  }%
}
```

ndWidestAnyName Like the above but doesn't check is the entry has been used.

```
\newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \settowidth{\dimen@}%
       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
      \ifdim\dimen@>\gls@tmplen
        \gls@tmplen=\dimen@
        \eglssetwidest{\glsentryname{\@glo@label}}%
      \fi
    }%
  }%
}
```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```
\newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \dimen@i=0pt\relax
  \dimen@ii=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {%
        \ifglshasparent{\@glo@label}%
        {%
          \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
          \ifglshasparent{\@glo@parent}%
          {%
            \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
            \ifglshasparent{\@glo@parent}%
            {}%
            {%
              \settowidth{\gls@tmplen}%
                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
              \ifdim\gls@tmplen>\dimen@ii
                \dimen@ii=\gls@tmplen
                \eglssetwidest[2]{\glsentryname{\@glo@label}}%
              \fi
            }%
          }%
          {%
            \settowidth{\gls@tmplen}%
               {\glstreenamefmt{\glsentryname{\@glo@label}}}%
            \ifdim\gls@tmplen>\dimen@i
              \dimen@i=\gls@tmplen
              \eglssetwidest[1]{\glsentryname{\@glo@label}}%
            \fi
          }%
        }%
        {%
          \settowidth{\gls@tmplen}%
             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
          \ifdim\gls@tmplen>\dimen@
            \dimen@=\gls@tmplen
            \eglssetwidest{\glsentryname{\@glo@label}}%
          \fi
        }%
      }%
      {}%
    }%
  }%
}
```

`dWidestLevelTwo` This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```
\newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \dimen@i=0pt\relax
  \dimen@ii=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglshasparent{\@glo@label}%
      {%
        \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
        \ifglshasparent{\@glo@parent}%
        {%
          \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
          \ifglshasparent{\@glo@parent}%
          {}%
          {%
            \settowidth{\gls@tmplen}%
               {\glstreenamefmt{\glsentryname{\@glo@label}}}%
            \ifdim\gls@tmplen>\dimen@ii
              \dimen@ii=\gls@tmplen
              \eglssetwidest[2]{\glsentryname{\@glo@label}}%
            \fi
          }%
        }%
        {%
          \settowidth{\gls@tmplen}%
             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
          \ifdim\gls@tmplen>\dimen@i
            \dimen@i=\gls@tmplen
            \eglssetwidest[1]{\glsentryname{\@glo@label}}%
          \fi
        }%
      }%
      {%
        \settowidth{\gls@tmplen}%
           {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\gls@tmplen>\dimen@
          \dimen@=\gls@tmplen
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
      }%
    }%
  }%
}
```

edAnyNameSymbol

Like the `\glsFindWidestUsedAnyName` but also measures the symbol. The length of the widest symbol is stored in the second argument should be a length register.

```
\newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
        \settowidth{\dimen@}%
         {\glsentrysymbol{\@glo@label}}%
        \ifdim\dimen@>#2\relax
          #2=\dimen@
        \fi
      }%
      {}%
    }%
  }%
}
```

stAnyNameSymbol

Like the above but doesn't check if the entry has been used.

```
\newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \settowidth{\dimen@}%
       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
      \ifdim\dimen@>\gls@tmplen
        \gls@tmplen=\dimen@
        \eglssetwidest{\glsentryname{\@glo@label}}%
      \fi
      \settowidth{\dimen@}%
       {\glsentrysymbol{\@glo@label}}%
      \ifdim\dimen@>#2\relax
        #2=\dimen@
      \fi
    }%
  }%
}
```

eSymbolLocation

Like the `\glsFindWidestUsedAnyNameSymbol` but also measures the location list. This requires `\glsentrynumberlist`. The length of the widest symbol is stored in the second argument should be a length register. The length of the widest location list is stored in the third argument, which should also be a length register.

```
\newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  #3=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
        \settowidth{\dimen@}%
         {\glsentrysymbol{\@glo@label}}%
        \ifdim\dimen@>#2\relax
          #2=\dimen@
        \fi
        \settowidth{\dimen@}%
         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
        \ifdim\dimen@>#3\relax
```

```
          #3=\dimen@
        \fi
      }%
      {}%
    }%
  }%
}
```

`eSymbolLocation` Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```
\newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  #3=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \settowidth{\dimen@}%
       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
      \ifdim\dimen@>\gls@tmplen
        \gls@tmplen=\dimen@
        \eglssetwidest{\glsentryname{\@glo@label}}%
      \fi
      \settowidth{\dimen@}%
       {\glsentrysymbol{\@glo@label}}%
      \ifdim\dimen@>#2\relax
        #2=\dimen@
      \fi
      \settowidth{\dimen@}%
        {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
      \ifdim\dimen@>#3\relax
        #3=\dimen@
      \fi
    }%
  }%
}
```

`AnyNameLocation` Like the `\glsFindWidestUsedAnyNameSymbolLocation` but doesn't measure the symbol. The length of the widest location list is stored in the second argument, which should be a length register.

```
\newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
```

```
      \ifglsused{\@glo@label}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
        \settowidth{\dimen@}%
         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
        \ifdim\dimen@>#2\relax
          #2=\dimen@
        \fi
      }%
      {}%
    }%
  }%
}
```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the first use flag.

```
\newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \settowidth{\dimen@}%
       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
      \ifdim\dimen@>\gls@tmplen
        \gls@tmplen=\dimen@
        \eglssetwidest{\glsentryname{\@glo@label}}%
      \fi
      \settowidth{\dimen@}%
       {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
      \ifdim\dimen@>#2\relax
        #2=\dimen@
      \fi
    }%
  }%
}
```

mputeTreeIndent Compute the value of `\glstreeindent`. Argument is the entry label. (Ignored in default definition, but this command may be redefined to take the particular entry into account.) Note that the sub-levels modify `\glstreeindent`.

```
\newcommand*{\glsxtrComputeTreeIndent}[1]{%
  \glstreeindent=\glsxtrtreetopindent\relax
}
```

uteTreeSubIndent

```
\glsxtrComputeTreeSubIndent{⟨level⟩}{⟨label⟩}{⟨register⟩}
```

Compute the indent for the sub-entries. The first argument is the level, the second argument is the entry label and the third argument is the length register used to store the computed indent.

```
\newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {%
    \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
  }%
  {%
    \settowidth{#3}{\glstreenamefmt{%
           \csname @glswidestname\romannumeral#1\endcsname\space}}%
  }%
}
```

eeSetHangIndent Set `\hangindent` for top-level entries:

```
\newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set `\hangindent` for sub-entries:

```
\newcommand*{\glsxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
\renewglossarystyle{alttree}{%
  \renewenvironment{theglossary}%
    {%
     \glsxtralttreeInit
     \def\@gls@prevlevel{-1}%
     \mbox{}\par}%
    {\par}%
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand{\glossentry}[2]{%
    \ifnum\@gls@prevlevel=0\relax
    \else
      \glsxtrComputeTreeIndent{##1}%
    \fi
    \parindent\glstreeindent
    \glsxtrAltTreeSetHangIndent
    \makebox[0pt][r]%
    {%
      \glstreenamebox{\glstreeindent}%
      {%
         \glsentryitem{##1}%
         \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
      }%
    }%
```

```
      \glsxtralttreeSymbolDescLocation{##1}{##2}%
      \def\@gls@prevlevel{0}%
    }
    \renewcommand{\subglossentry}[3]{%
      \ifnum##1=1\relax
        \glssubentryitem{##2}%
      \fi
      \ifnum\@gls@prevlevel=##1\relax
      \else
        \glsxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
        \ifnum\@gls@prevlevel<##1\relax
          \setlength\glstreeindent\gls@tmplen
          \addtolength\glstreeindent\parindent
          \parindent\glstreeindent
        \else
          \ifnum\@gls@prevlevel=0\relax
            \glsxtrComputeTreeIndent{##2}%
          \else
            \glsxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
          \fi
          \addtolength\parindent{-\glstreeindent}%
          \setlength\glstreeindent\parindent
        \fi
      \fi
      \glsxtrAltTreeSetSubHangIndent{##1}%
      \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
        \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
      \glsxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
      \def\@gls@prevlevel{##1}%
    }%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
  }
}%
{%
}
```

Redefine alttreegroup so that it discourages a break after group headings. Can't use `\@afterheading` here as it messes with the first item of the group.

```
\ifdef{\@glsstyle@alttreegroup}
{%
  \renewglossarystyle{alttreegroup}{%
    \setglossarystyle{alttree}%
    \renewcommand{\glsgroupheading}[1]{\par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
      \nopagebreak\indexspace\nopagebreak
    }%
  }%
}%
{%
}
```

Similarly for alttreehypergroup.

```
\ifdef{\@glsstyle@alttreehypergroup}
{%
  \renewglossarystyle{alttreehypergroup}{%
    \setglossarystyle{alttree}%
    \renewcommand*{\glossaryheader}{%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreenavigationfmt{\glsnavigation}\par\indexspace
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt
       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
      \nopagebreak\indexspace\nopagebreak
    }%
  }
}%
{%
}
```

2.9 Multicolumn Styles

Adjust mcolindexgroup to discourage page breaks after the group headings.

```
\ifdef{\@glsstyle@mcolindexgroup}
{%
  \renewglossarystyle{mcolindexgroup}{%
    \setglossarystyle{mcolindex}%
    \renewcommand*{\glsgroupheading}[1]{%
      \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
      \nopagebreak\indexspace\nobreak\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcolindexhypergroup.

```
\ifdef{\@glsstyle@mcolindexhypergroup}
{%
```

```
  \renewglossarystyle{mcolindexhypergroup}{%
    \setglossarystyle{mcolindex}%
    \renewcommand*{\glossaryheader}{%
      \item\glstreenavigationfmt{\glsnavigation}%
      \indexspace
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \item\glstreegroupheaderfmt
        {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace\nobreak\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcolindexspannav.

```
\ifdef{\@glsstyle@mcolindexspannav}
{%
  \renewglossarystyle{mcolindexspannav}{%
    \setglossarystyle{index}%
    \renewenvironment{theglossary}%
    {%
     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]%
     \setlength{\parindent}{0pt}%
     \setlength{\parskip}{0pt plus 0.3pt}%
     \let\item\glstreeitem}%
    {\end{multicols}}%
    \renewcommand*{\glsgroupheading}[1]{%
      \item\glstreegroupheaderfmt
        {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace\nobreak\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreegroup.

```
\ifdef{\@glsstyle@mcoltreegroup}
{%
  \renewglossarystyle{mcoltreegroup}{%
    \setglossarystyle{mcoltree}%
    \renewcommand{\glsgroupheading}[1]{\par
      \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
      \nopagebreak\indexspace\nobreak\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreehypergroup.

```
\ifdef{\@glsstyle@mcoltreehypergroup}
{%
  \renewglossarystyle{mcoltreehypergroup}{%
    \setglossarystyle{mcoltree}%
    \renewcommand*{\glossaryheader}{%
      \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \par\noindent
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace\nobreak\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreespannav.

```
\ifdef{\@glsstyle@mcoltreespannav}
{%
  \renewglossarystyle{mcoltreespannav}{%
    \setglossarystyle{tree}%
    \renewenvironment{theglossary}%
    {%
       \begin{multicols}{\glsmcols}%
         [\noindent\glstreenavigationfmt{\glsnavigation}]%
       \setlength{\parindent}{0pt}%
       \setlength{\parskip}{0pt plus 0.3pt}%
    }%
    {\end{multicols}}%
    \renewcommand*{\glsgroupheading}[1]{%
      \par\noindent
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace\nobreak\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreenonamegroup.

```
\ifdef{\@glsstyle@mcoltreenonamegroup}
{%
  \renewglossarystyle{mcoltreenonamegroup}{%
    \setglossarystyle{mcoltreenoname}%
    \renewcommand{\glsgroupheading}[1]{\par
      \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
      \nopagebreak\indexspace\nobreak\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreenonamehypergroup.

```
\ifdef{\@glsstyle@mcoltreenonamehypergroup}
{%
  \renewglossarystyle{mcoltreenonamehypergroup}{%
    \setglossarystyle{mcoltreenoname}%
    \renewcommand*{\glossaryheader}{%
      \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    \renewcommand*{\glsgroupheading}[1]{%
      \par\noindent
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace\nobreak\@afterheading}%
  }
}%
{%
}
```

Similarly for mcoltreenonamespannav.

```
\ifdef{\@glsstyle@mcoltreenonamespannav}
{%
  \renewglossarystyle{mcoltreenonamespannav}{%
    \setglossarystyle{treenoname}%
    \renewenvironment{theglossary}%
    {%
       \begin{multicols}{\glsmcols}%
         [\noindent\glstreenavigationfmt{\glsnavigation}]%
       \setlength{\parindent}{0pt}%
       \setlength{\parskip}{0pt plus 0.3pt}%
    }%
    {\end{multicols}}%
    \renewcommand*{\glsgroupheading}[1]{%
      \par\noindent
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace\nobreak\@afterheading}%
  }
}%
{%
}
```

mcolalttree needs adjusting so that it uses `\glsxtralttreeInit` This doesn't use `\mbox{}\par` which would unbalance the top of the columns.

```
\ifdef{\@glsstyle@mcolalttree}
{%
  \renewglossarystyle{mcolalttree}{%
    \setglossarystyle{alttree}%
    \renewenvironment{theglossary}%
    {%
       \glsxtralttreeInit
       \def\@gls@prevlevel{-1}%
       \begin{multicols}{\glsmcols}%
    }%
    {\par\end{multicols}}%
  }
}%
{%
}
```

Redefine mcolalttreegroup to discourage page breaks after the group headings.

```
\ifdef{\@glsstyle@mcolalttreegroup}
{%
  \renewglossarystyle{mcolalttreegroup}{%
    \setglossarystyle{mcolalttree}%
    \renewcommand{\glsgroupheading}[1]{\par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
      \nopagebreak\indexspace\nopagebreak
    }%
  }
}%
{%
}
```

Similarly for mcolalttreehypergroup.

```
\ifdef{\@glsstyle@mcolalttreehypergroup}
{%
  \renewglossarystyle{mcolalttreehypergroup}{%
    \setglossarystyle{mcolalttree}%
    \renewcommand*{\glossaryheader}{%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreenavigationfmt{\glsnavigation}%
      \par\indexspace
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace\nopagebreak
    }%
  }
}%
{%
}
```

Similarly for mcolalttreespannav.

```
\ifdef{\@glsstyle@mcolalttreespannav}
{%
  \renewglossarystyle{mcolalttreespannav}{%
    \setglossarystyle{alttree}%
    \renewenvironment{theglossary}%
    {%
       \glsxtralttreeInit
       \def\@gls@prevlevel{-1}%
       \begin{multicols}{\glsmcols}%
          [\noindent\glstreenavigationfmt{\glsnavigation}]%
    }%
    {\par\end{multicols}}%
    \renewcommand*{\glsgroupheading}[1]{%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
      \nopagebreak\indexspace\nopagebreak
    }%
  }
}%
{%
}
```

Reset the default style

```
\ifx\@glossary@default@style\relax
\else
  \setglossarystyle{\@glsxtr@current@style}
\fi
```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossary-bookindex}[2018/12/01 1.38 (NLCT)]
```

Load required packages.

```
\RequirePackage{multicol}
\RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
\newcommand{\glsxtrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
\newcommand*{\glsxtrbookindexname}[1]{\glossentryname{#1}}
```

`ookindexsubname` Format used for sub entries.

```
\newcommand*{\glsxtrbookindexsubname}[1]{\glsxtrbookindexname{#1}}
```

`sxtrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
\providecommand*{\glsxtrprelocation}{\space}
```

`ndexprelocation` Separator used before location list for top-level entries. Version 1.22 has removed the `\ifglsnopostdot` check since this style doesn't display the description.

```
\newcommand*{\glsxtrbookindexprelocation}[1]{%
  \glsxtrifhasfield{location}{#1}%
  {,\glsxtrprelocation}%
  {\glsxtrprelocation}%
}
```

`xsubprelocation` Separator used before location list for sub-entries.

```
\newcommand*{\glsxtrbookindexsubprelocation}[1]{%
 \glsxtrbookindexprelocation{#1}%
}
```

`xparentchildsep` Separator used between top-level parent and child entry.

```
\newcommand{\glsxtrbookindexparentchildsep}{\nopagebreak}
```

`rentsubchildsep` Separator used between sub-level parent and child entry.

```
\newcommand{\glsxtrbookindexparentsubchildsep}{\glsxtrbookindexparentchildsep}
```

`ookindexbetween` Between two top-level entries identified by the labels in the arguments.

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

`indexsubbetween` Between two level 1 entries identified by the labels in the arguments.

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

`exsubsubbetween` Between two level 2 entries identified by the labels in the arguments.

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

`indexatendgroup` At the end of a letter group. The argument is the index of the last top-level entry.

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

`exsubatendgroup` At the end of a letter group. The argument is the index of the last level 1 entry.

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

`ubsubatendgroup` At the end of a letter group. The argument is the index of the last level 2 entry.

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

`kindexgroupskip` Group separator.

```
\newcommand{\glsxtrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

Format group title.

`dexformatheader` Group separator.

```
\newcommand*{\glsxtrbookindexformatheader}[1]{%
  \par{\centering\glstreegroupheaderfmt{#1}\par}%
}
```

`okindexbookmark` Book mark group heading if supported.

```
\ifdef\pdfbookmark
{%
  \newcommand*{\glsxtrbookindexbookmark}[2]{%
    \ifdefstring{\@@glossarysec}{chapter}%
    {\pdfbookmark[1]{#1}{#2}}%
    {\pdfbookmark[2]{#1}{#2}}%
  }
}
{%
  \newcommand*{\glsxtrbookindexbookmark}[2]{}
}
```

`kindexcolspread`

```
\newcommand*{\glsxtrbookindexcolspread}{}
```

`dexmulticolsenv`

```
\newcommand*{\glsxtrbookindexmulticolsenv}{multicols}
```

Define the style.

```
\newglossarystyle{bookindex}{%
  \setglossarystyle{index}%
  \renewenvironment{theglossary}%
  {%
    \ifdefempty\glsxtrbookindexcolspread
    {%
      \expandafter\begin\expandafter{\glsxtrbookindexmulticolsenv}%
        {\glsxtrbookindexcols}%
    }%
    {%
      \expandafter\begin\expandafter{\glsxtrbookindexmulticolsenv}%
        {\glsxtrbookindexcols}[\glsxtrbookindexcolspread]%
    }%
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \let\@glsxtr@bookindex@sep\glsxtrbookindexparentchildsep
    \let\@glsxtr@bookindex@subsep\glsxtrbookindexparentsubchildsep
    \let\@glsxtr@bookindex@between\@gobble
    \let\@glsxtr@bookindex@subbetween\@gobble
    \let\@glsxtr@bookindex@subsubbetween\@gobble
    \let\@glsxtr@bookindex@atendgroup\relax
    \let\@glsxtr@bookindex@subatendgroup\relax
    \let\@glsxtr@bookindex@subsubatendgroup\relax
    \let\@glsxtr@bookindexgroupskip\relax
  }%
  {%
```

Do end group hooks.

```
     \@glsxtr@bookindex@subsubatendgroup
     \@glsxtr@bookindex@subatendgroup
     \@glsxtr@bookindex@atendgroup
```

End multicols environment.

```
      \expandafter\end\expandafter{\glsxtrbookindexmulticolsenv}%
  }%
```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```
  \renewcommand*{\glossaryheader}{\raggedright}%
```

Top level entry format.

```
  \renewcommand*{\glossentry}[2]{%
```

Do separator.

```
    \@glsxtr@bookindex@between{##1}%
```

Update separators.

```
    \let\@glsxtr@bookindex@sep\glsxtrbookindexparentchildsep
    \let\@glsxtr@bookindex@subsep\glsxtrbookindexparentsubchildsep
    \let\@glsxtr@bookindex@subbetween\@gobble
    \let\@glsxtr@bookindex@subsubbetween\@gobble
    \edef\@glsxtr@bookindex@between{%
```

```
      \noexpand\glsxtrbookindexbetween{##1}%
    }%
    \edef\@glsxtr@bookindex@atendgroup{%
      \noexpand\glsxtrbookindexatendgroup{##1}%
    }%
    \let\@glsxtr@bookindex@subatendgroup\relax
    \let\@glsxtr@bookindex@subsubatendgroup\relax
```

Format entry.

```
    \glstreeitem
      \glsentryitem{##1}%
      \glstarget{##1}{\glsxtrbookindexname{##1}}%
    \glsxtrbookindexprelocation{##1}##2%
  }%
  \renewcommand{\subglossentry}[3]{%
    \ifcase##1\relax
```

Level 0 (shouldn't happen as that's formatted with `\glossentry`).

```
      \glstreeitem
    \or
```

Level 1.

```
      \@glsxtr@bookindex@sep
      \@glsxtr@bookindex@subbetween{##2}%
      \let\@glsxtr@bookindex@sep\relax
```

Update separators.

```
      \let\@glsxtr@bookindex@subsubbetween\@gobble
      \let\@glsxtr@bookindex@subsep\glsxtrbookindexparentsubchildsep
      \edef\@glsxtr@bookindex@subbetween{%
         \noexpand\glsxtrbookindexsubbetween{##2}%
      }%
      \edef\@glsxtr@bookindex@atsubendgroup{%
        \noexpand\glsxtrbookindexatsubendgroup{##1}%
      }%
```

Start sub-item.

```
      \glstreesubitem
      \glssubentryitem{##2}%
    \else
```

All other levels.

```
      \@glsxtr@bookindex@subsep
      \@glsxtr@bookindex@subsubbetween{##2}%
```

Update separators.

```
      \let\@glsxtr@bookindex@subsep\relax
      \edef\@glsxtr@bookindex@subsubbetween{%
         \noexpand\glsxtrbookindexsubsubbetween{##2}%
      }%
      \edef\@glsxtr@bookindex@atsubsubendgroup{%
        \noexpand\glsxtrbookindexatsubsubendgroup{##1}%
      }%
```

Start sub-sub-item.

```
      \glstreesubsubitem
    \fi
```

Format entry.

```
    \glstarget{##2}{\glsxtrbookindexsubname{##2}}%
    \glsxtrbookindexsubprelocation{##2}##3%
  }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
  \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
  \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
    \@glsxtr@bookindex@subsubatendgroup
    \@glsxtr@bookindex@subatendgroup
    \@glsxtr@bookindex@atendgroup
    \@glsxtr@bookindexgroupskip
```

Update separators.

```
    \let\@glsxtr@bookindexgroupskip\glsxtrbookindexgroupskip
    \let\@glsxtr@bookindex@between\@gobble
    \let\@glsxtr@bookindex@atendgroup\relax
    \let\@glsxtr@bookindex@subatendgroup\relax
    \let\@glsxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
    \glsxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
    \glsxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
    \glsxtrbookindexformatheader{\thisgrptitle}%
    \nopagebreak\indexspace\nopagebreak\@afterheading
  }%
}
```

Some supplementary commands that may be useful. These store the entry label for the current page. Since the page number is needed in the control sequence, this uses `\glsxtrbookindexthepage` instead of `\thepage` in case the page numbering has been set to something that contains formatting commands.

ookindexthepage

The `\@printglossary` sets `\currentglossary` to the current glossary label. This is used as a prefix in case the page number is reset.

```
\newcommand{\glsxtrbookindexthepage}{%
 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
}
```

kindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
\newcommand*{\glsxtrbookindexmarkentry}[1]{%
  \protected@write\@auxout
  {\let\glsxtrbookindexthepage\relax}%
  {\string\glsxtr@setbookindexmark{\glsxtrbookindexthepage}{#1}}%
}
```

etbookindexmark

```
\newcommand*{\glsxtr@setbookindexmark}[2]{%
 \ifcsundef{glsxtr@idxfirstmark@#1}%
 {\csgdef{glsxtr@idxfirstmark@#1}{#2}}%
 {}%
 \csgdef{glsxtr@idxlastmark@#1}{#2}%
}
```

dexfirstmarkfmt

```
\newcommand*{\glsxtrbookindexfirstmarkfmt}[1]{%
  \glsentryname{#1}%
}
```

kindexfirstmark

```
\newcommand*{\glsxtrbookindexfirstmark}{%
  \letcs{\glsxtr@label}{glsxtr@idxfirstmark@\glsxtrbookindexthepage}%
  \ifdef\glsxtr@label
  {\glsxtrbookindexfirstmarkfmt{\glsxtr@label}}%
  {}%
}
```

ndexlastmarkfmt

```
\newcommand*{\glsxtrbookindexlastmarkfmt}[1]{%
  \glsentryname{#1}%
}
```

okindexlastmark

```
\newcommand*{\glsxtrbookindexlastmark}{%
  \letcs{\glsxtr@label}{glsxtr@idxlastmark@\glsxtrbookindexthepage}%
  \ifdef\glsxtr@label
  {\glsxtrbookindexlastmarkfmt{\glsxtr@label}}%
  {}%
}
```

4 longextra styles (glossary-longextra.sty)

4.1 Package Initialisation and Options

Provides additional long styles.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossary-longextra}[2018/12/01 1.38 (NLCT)]
```

Load required packages.

```
\RequirePackage{glossary-longbooktabs}
```

longextraNameFmt

```
\glslongextraNameFmt{⟨label⟩}
```

Governs the way the name is displayed.

```
\newcommand{\glslongextraNameFmt}[1]{%
 \glsentryitem{#1}\glstarget{#1}{\glossentryname{#1}}%
}
```

longextraDescFmt

```
\glslongextraDescFmt{⟨label⟩}
```

Governs the way the description is displayed.

```
\newcommand{\glslongextraDescFmt}[1]{%
  \glossentrydesc{#1}\glspostdescription
}
```

ngextraSymbolFmt

```
\glslongextraSymbolFmt{⟨label⟩}
```

Governs the way the symbol is displayed.

```
\newcommand{\glslongextraSymbolFmt}[1]{\glossentrysymbol{#1}}
```

extraLocationFmt

```
\glslongextraLocationFmt{⟨label⟩}{⟨location list⟩}
```

Governs the way the location is displayed.

```
\newcommand{\glslongextraLocationFmt}[2]{#2}
```

gextraSubNameFmt

```
\glslongextraSubNameFmt{⟨level⟩}{⟨label⟩}
```

Governs the way the child name is displayed. Just does the sub-entry counter, if enabled, and the target.

```
\newcommand{\glslongextraSubNameFmt}[2]{%
 \glssubentryitem{#2}\glstarget{#2}{\strut}%
}
```

gextraSubDescFmt

```
\glslongextraSubDescFmt{⟨level⟩}{⟨label⟩}
```

Governs the way the child description is displayed.

```
\newcommand{\glslongextraSubDescFmt}[2]{%
  \glslongextraDescFmt{#2}%
}
```

xtraSubSymbolFmt

```
\glslongextraSubSymbolFmt{⟨level⟩}{⟨label⟩}
```

Governs the way the child symbol is displayed.

```
\newcommand{\glslongextraSubSymbolFmt}[2]{%
  \glslongextraSymbolFmt{#2}%
}
```

raSubLocationFmt

```
\glslongextraSubLocationFmt{⟨level⟩}{⟨label⟩}{⟨location list⟩}
```

Governs the way the child location list is displayed.

```
\newcommand{\glslongextraSubLocationFmt}[3]{#3}
```

gextraNameAlign Alignment for the name column.

```
\newcommand{\glslongextraNameAlign}{l}
```

gextraDescAlign Alignment for the description column.

```
\newcommand{\glslongextraDescAlign}{>{\raggedright}p{\glsdescwidth}}
```

xtraSymbolAlign Alignment for the symbol column.

```
\newcommand{\glslongextraSymbolAlign}{c}
```

`raLocationAlign` Alignment for the location column.

```
\newcommand{\glslongextraLocationAlign}{>{\raggedright}p{\glspagelistwidth}}
```

`traGroupHeading` Used to format the letter group headings. The first argument is the number of columns in the table. The second is the group *label* (not the title).

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

`traHeaderFormat` Format for the column headers.

```
\newcommand{\glslongextraHeaderFmt}[1]{\textbf{#1}}
```

`aNameDescHeader`

```
\newcommand{\glslongextraNameDescHeader}{%
 \glslongextraNameDescTabularHeader\endhead
 \glslongextraNameDescTabularFooter\endfoot
}
```

`scTabularHeader`

```
\newcommand{\glslongextraNameDescTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\entryname &
 \glslongextraHeaderFmt\descriptionname\tabularnewline
 \midrule
}
```

`scTabularFooter`

```
\newcommand{\glslongextraNameDescTabularFooter}{%
 \bottomrule
}
```

Unlike the alttree style, there aren't different widths for the hierarchical levels.

`gextraSetWidest` Provide in case the tree styles haven't been loaded.

```
\newcommand*{\glslongextraSetWidest}[1]{%
 \def\@glslongextrawidestname{#1}%
}
```

`extrawidestname` Pick up the widest name from the alttree style if it has been set. (Will expand to nothing otherwise.)

```
\newcommand*{\@glslongextrawidestname}{\csuse{@glswidestname}}
```

`traUpdateWidest`

```
\newcommand*{\glslongextraUpdateWidest}[1]{%
  \ifundef\@glslongextrawidestname
  {\def\@glslongextrawidestname{#1}}%
  {%
    \settowidth{\dimen@}{\@glslongextrawidestname}%
    \settowidth{\dimen@ii}{#1}%
    \ifdim\dimen@ii>\dimen@
    \def\@glslongextrawidestname{#1}%
   \fi
  }%
}
```

pdateWidestChild `\glslongextraUpdateWidestChild{⟨level⟩}{⟨text⟩}`

Used by `\glsxtrSetWidest` in glossaries-extra-bib2gls. Does nothing by default, since the default action in these styles is to omit the child name. If the child name should be displayed, then this needs to be redefined to use `\glslongextraUpdateWidest`.

```
\newcommand*{\glslongextraUpdateWidestChild}[2]{}
```

traSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name and description columns.

```
\newcommand{\glslongextraSetDescWidth}{%
  \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\entryname}%
```

Has the widest name been set.

```
  \settowidth{\dimen@}{\glsnamefont{\@glslongextrawidestname}}%
  \ifdim\dimen@>\gls@tmplen
   \gls@tmplen=\dimen@
  \fi
```

Description width is `\linewidth` less 4`\tabcolsep` less the width of the name column.

```
  \setlength{\glsdescwidth}{\dimexpr\linewidth-4\tabcolsep-\gls@tmplen}%
}
```

SymSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name, symbol and description columns.

```
\newcommand{\glslongextraSymSetDescWidth}{%
```

Work out the size for just the name and description style.

```
  \glslongextraSetDescWidth
```

Now work out the symbol column width. This is assuming that the column title will be the widest text in the column.

```
  \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\symbolname}%
```

Subtract 2`\tabcolsep` and the symbol header width.

```
  \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\gls@tmplen}%
}
```

LocSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name, location and description columns.

```
\newcommand{\glslongextraLocSetDescWidth}{%
```

Work out the size for just the name and description style.

```
  \glslongextraSetDescWidth
```

Subtract 2`\tabcolsep` and the location list column width.

```
  \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
}
```

`LocSetDescWidth` Computes the value of `\glsdescwidth` for the styles that have name, symbol, location and description columns.

```
\newcommand{\glslongextraSymLocSetDescWidth}{%
```

Work out the size for just the name, symbol and description style.

```
  \glslongextraSymSetDescWidth
```

Subtract 2`\tabcolsep` and the location list column width.

```
  \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
}
```

`ExtraUseTabular` If true use tabular instead of longtable. Obviously only intended for short glossaries that can fit into a single page.

```
\newif\ifGlsLongExtraUseTabular
\GlsLongExtraUseTabularfalse
```

`raTabularVAlign` Only used with the tabular setting.

```
\newcommand*{\glslongextraTabularVAlign}{c}
```

`long-name-desc` Two column style with multi-lined descriptions and header. This is similar to the longragged-booktabs style.

```
\newglossarystyle{long-name-desc}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
          \expandonce\glslongextraNameAlign
          \expandonce\glslongextraDescAlign}}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraNameDescTabularFooter
      \end{tabular}%
    }%
   \renewcommand*{\glossaryheader}{\glslongextraNameDescTabularHeader}%
  \else
   \renewenvironment{theglossary}%
    {%
      \glspatchLToutput
      \glslongextraSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
```

```
          \expandonce\glslongextraNameAlign
          \expandonce\glslongextraDescAlign}}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraNameDescHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraNameFmt{##1} &
    \glslongextraDescFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubNameFmt{##1}{##2}
     &
     \glslongextraSubDescFmt{##1}{##2}%
     \tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

cLocationHeader

```
\newcommand{\glslongextraNameDescLocationHeader}{%
 \glslongextraNameDescLocationTabularHeader\endhead
 \glslongextraNameDescLocationTabularFooter\endfoot
}
```

onTabularHeader

```
\newcommand{\glslongextraNameDescLocationTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\entryname &
 \glslongextraHeaderFmt\descriptionname &
 \glslongextraHeaderFmt\pagelistname\tabularnewline
 \midrule
}
```

onTabularFooter

```
\newcommand{\glslongextraNameDescLocationTabularFooter}{%
 \bottomrule
}
```

g-name-desc-loc Three columns: name, description and location list.

```
\newglossarystyle{long-name-desc-loc}%
{%
  \ifGlsLongExtraUseTabular
```

```
 \renewenvironment{theglossary}%
  {%
    \glslongextraLocSetDescWidth
    \edef\@glslongextra@begintab{%
      \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
        \expandonce\glslongextraNameAlign
        \expandonce\glslongextraDescAlign
        \expandonce\glslongextraLocationAlign
    }}%
    \@glslongextra@begintab
  }%
  {%
    \glslongextraNameDescLocationTabularFooter
    \end{tabular}%
  }%
 \renewcommand*{\glossaryheader}{\glslongextraNameDescLocationTabularHeader}%
\else
 \renewenvironment{theglossary}%
  {%
    \glspatchLToutput
    \glslongextraLocSetDescWidth
    \edef\@glslongextra@begintab{%
      \noexpand\begin{longtable}{%
        \expandonce\glslongextraNameAlign
        \expandonce\glslongextraDescAlign
        \expandonce\glslongextraLocationAlign
    }}%
    \@glslongextra@begintab
  }%
  {\end{longtable}}%
 \renewcommand*{\glossaryheader}{\glslongextraNameDescLocationHeader}%
 \fi
\renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
\renewcommand{\glossentry}[2]{%
  \glslongextraNameFmt{##1} &
  \glslongextraDescFmt{##1} &
  \glslongextraLocationFmt{##1}{##2}\tabularnewline
}%
\renewcommand{\subglossentry}[3]{%
   \glslongextraSubNameFmt{##1}{##2}&
   \glslongextraSubDescFmt{##1}{##2}&
   \glslongextraSubLocationFmt{##1}{##2}{##3}%
   \tabularnewline
}%
\ifglsnogroupskip
  \renewcommand*{\glsgroupskip}{}%
\else
  \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
\fi
```

```
}
```

aDescNameHeader

```
\newcommand{\glslongextraDescNameHeader}{%
 \glslongextraDescNameTabularHeader\endhead
 \glslongextraDescNameTabularFooter\endfoot
}
```

meTabularHeader

```
\newcommand{\glslongextraDescNameTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\descriptionname&
 \glslongextraHeaderFmt\entryname \tabularnewline
 \midrule
}
```

meTabularFooter

```
\newcommand{\glslongextraDescNameTabularFooter}{%
 \bottomrule
}
```

long-desc-name Like name-desc but swaps the columns.

```
\newglossarystyle{long-desc-name}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraNameAlign}}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraDescNameTabularFooter
      \end{tabular}%
    }%
      \renewcommand*{\glossaryheader}{\glslongextraDescNameTabularHeader}%
  \else
   \renewenvironment{theglossary}%
    {%
      \glspatchLToutput
      \glslongextraSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraNameAlign}}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
    \renewcommand*{\glossaryheader}{\glslongextraDescNameHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraDescFmt{##1} &
    \glslongextraNameFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubDescFmt{##1}{##2} &
     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

nDescNameHeader

```
\newcommand{\glslongextraLocationDescNameHeader}{%
 \glslongextraLocationDescNameTabularHeader\endhead
 \glslongextraLocationDescNameTabularFooter\endfoot
}
```

meTabularHeader

```
\newcommand{\glslongextraLocationDescNameTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\pagelistname&
 \glslongextraHeaderFmt\descriptionname&
 \glslongextraHeaderFmt\entryname \tabularnewline
 \midrule
}
```

meTabularFooter

```
\newcommand{\glslongextraLocationDescNameTabularFooter}{%
 \bottomrule
}
```

g-loc-desc-name Three columns: location, description and name.

```
\newglossarystyle{long-loc-desc-name}%
{%
  \ifGlsLongExtraUseTabular
   {%
     \glslongextraLocSetDescWidth
     \edef\@glslongextra@begintab{%
       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
         \expandonce\glslongextraLocationAlign
         \expandonce\glslongextraDescAlign
         \expandonce\glslongextraNameAlign}}%
     \@glslongextra@begintab
   }%
   {%
     \glslongextraLocationDescNameTabularFooter
     \end{tabular}%
   }%
   \renewcommand*{\glossaryheader}{\glslongextraLocationDescNameTabularHeader}%
  \else
  \renewenvironment{theglossary}%
   {%
     \glspatchLToutput
     \glslongextraLocSetDescWidth
     \edef\@glslongextra@begintab{%
       \noexpand\begin{longtable}{%
         \expandonce\glslongextraLocationAlign
         \expandonce\glslongextraDescAlign
         \expandonce\glslongextraNameAlign}}%
     \@glslongextra@begintab
   }%
   {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraLocationDescNameHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraLocationFmt{##1}{##2} &
    \glslongextraDescFmt{##1} &
    \glslongextraNameFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubLocationFmt{##1}{##2}{##3} &
     \glslongextraSubDescFmt{##1}{##2} &
     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

meDescSymHeader

```
\newcommand{\glslongextraNameDescSymHeader}{%
 \glslongextraNameDescSymTabularHeader\endhead
 \glslongextraNameDescSymTabularFooter\endfoot
}
```

ymTabularHeader

```
\newcommand{\glslongextraNameDescSymTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\entryname &
 \glslongextraHeaderFmt\descriptionname &
 \glslongextraHeaderFmt\symbolname\tabularnewline
 \midrule
}
```

ymTabularFooter

```
\newcommand{\glslongextraNameDescSymTabularFooter}{%
 \bottomrule
}
```

g-name-desc-sym Three column style with symbol in the third column.

```
\newglossarystyle{long-name-desc-sym}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSymSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
          \expandonce\glslongextraNameAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraSymbolAlign
        }}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraNameDescSymTabularFooter
      \end{tabular}%
    }%
   \renewcommand*{\glossaryheader}{\glslongextraNameDescSymTabularHeader}%
  \else
   \renewenvironment{theglossary}%
    {%
      \glspatchLToutput
      \glslongextraSymSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
          \expandonce\glslongextraNameAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraSymbolAlign
        }}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraNameDescSymHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraNameFmt{##1} &
    \glslongextraDescFmt{##1} &
    \glslongextraSymbolFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubNameFmt{##1}{##2} &
     \glslongextraSubDescFmt{##1}{##2} &
     \glslongextraSubSymbolFmt{##1}{##2}%
     \tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

mLocationHeader

```
\newcommand{\glslongextraNameDescSymLocationHeader}{%
 \glslongextraNameDescSymLocationTabularHeader\endhead
 \glslongextraNameDescSymLocationTabularFooter\endfoot
}
```

onTabularHeader

```
\newcommand{\glslongextraNameDescSymLocationTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\entryname &
 \glslongextraHeaderFmt\descriptionname &
 \glslongextraHeaderFmt\symbolname &
 \glslongextraHeaderFmt\pagelistname\tabularnewline
 \midrule
}
```

onTabularFooter

```
\newcommand{\glslongextraNameDescSymLocationTabularFooter}{%
 \bottomrule
}
```

me-desc-sym-loc Four columns: name, description and location

```
\newglossarystyle{long-name-desc-sym-loc}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSymLocSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
```

```
          \expandonce\glslongextraNameAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraLocationAlign
        }}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraNameDescSymLocationTabularFooter
      \end{tabular}%
    }%
   \renewcommand*{\glossaryheader}{\glslongextraNameDescSymLocationTabularHeader}%
  \else
   \renewenvironment{theglossary}%
    {%
      \glspatchLToutput
      \glslongextraSymLocSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
          \expandonce\glslongextraNameAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraLocationAlign
        }}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraNameDescSymLocationHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraNameFmt{##1} &
    \glslongextraDescFmt{##1} &
    \glslongextraSymbolFmt{##1}&
    \glslongextraLocationFmt{##1}{##2}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubNameFmt{##1}{##2} &
     \glslongextraSubDescFmt{##1}{##2} &
     \glslongextraSubSymbolFmt{##1}{##2}&
     \glslongextraSubLocationFmt{##1}{##2}{##3}%
     \tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

meSymDescHeader

```
\newcommand{\glslongextraNameSymDescHeader}{%
 \glslongextraNameSymDescTabularHeader\endhead
 \glslongextraNameSymDescTabularFooter\endfoot
}
```

scTabularHeader

```
\newcommand{\glslongextraNameSymDescTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\entryname &
 \glslongextraHeaderFmt\symbolname &
 \glslongextraHeaderFmt\descriptionname\tabularnewline
 \midrule
}
```

scTabularFooter

```
\newcommand{\glslongextraNameSymDescTabularFooter}{%
 \bottomrule
}
```

g-name-sym-desc Three column style with symbol in the second column.

```
\newglossarystyle{long-name-sym-desc}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSymSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
          \expandonce\glslongextraNameAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraDescAlign
        }}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraNameSymDescTabularFooter
      \end{tabular}%
    }%
   \renewcommand*{\glossaryheader}{\glslongextraNameSymDescTabularHeader}%
  \else
   \renewenvironment{theglossary}%
    {%
      \glspatchLToutput
      \glslongextraSymSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
          \expandonce\glslongextraNameAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraDescAlign
        }}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraNameSymDescHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraNameFmt{##1} &
    \glslongextraSymbolFmt{##1} &
    \glslongextraDescFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubNameFmt{##1}{##2} &
     \glslongextraSubSymbolFmt{##1}{##2} &
     \glslongextraSubDescFmt{##1}{##2}\tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

cLocationHeader

```
\newcommand{\glslongextraNameSymDescLocationHeader}{%
 \glslongextraNameSymDescLocationTabularHeader\endhead
 \glslongextraNameSymDescLocationTabularFooter\endfoot
}
```

onTabularHeader

```
\newcommand{\glslongextraNameSymDescLocationTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\entryname &
 \glslongextraHeaderFmt\symbolname &
 \glslongextraHeaderFmt\descriptionname &
 \glslongextraHeaderFmt\pagelistname\tabularnewline
 \midrule
}
```

onTabularFooter

```
\newcommand{\glslongextraNameSymDescLocationTabularFooter}{%
 \bottomrule
}
```

me-sym-desc-loc Four column style with symbol in the second column.

```
\newglossarystyle{long-name-sym-desc-loc}%
{%
```

```
\ifGlsLongExtraUseTabular
  \renewenvironment{theglossary}%
   {%
     \glslongextraSymLocSetDescWidth
     \edef\@glslongextra@begintab{%
       \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
         \expandonce\glslongextraNameAlign
         \expandonce\glslongextraSymbolAlign
         \expandonce\glslongextraDescAlign
         \expandonce\glslongextraLocationAlign
       }}%
     \@glslongextra@begintab
   }%
   {%
     \glslongextraNameSymDescLocationTabularFooter
     \end{tabular}%
   }%
  \renewcommand*{\glossaryheader}{\glslongextraNameSymDescLocationTabularHeader}%
\else
  \renewenvironment{theglossary}%
   {%
     \glspatchLToutput
     \glslongextraSymLocSetDescWidth
     \edef\@glslongextra@begintab{%
       \noexpand\begin{longtable}{%
         \expandonce\glslongextraNameAlign
         \expandonce\glslongextraSymbolAlign
         \expandonce\glslongextraDescAlign
         \expandonce\glslongextraLocationAlign
       }}%
     \@glslongextra@begintab
   }%
   {\end{longtable}}%
  \renewcommand*{\glossaryheader}{\glslongextraNameSymDescLocationHeader}%
\fi
\renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
\renewcommand{\glossentry}[2]{%
  \glslongextraNameFmt{##1} &
  \glslongextraSymbolFmt{##1} &
  \glslongextraDescFmt{##1} &
  \glslongextraLocationFmt{##1}{##2}\tabularnewline
}%
\renewcommand{\subglossentry}[3]{%
   \glslongextraSubNameFmt{##1}{##2} &
   \glslongextraSubSymbolFmt{##1}{##2} &
   \glslongextraSubDescFmt{##1}{##2} &
   \glslongextraSubLocationFmt{##1}{##2}{##3}\tabularnewline
}%
\ifglsnogroupskip
```

```
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

mDescNameHeader

```
\newcommand{\glslongextraSymDescNameHeader}{%
 \glslongextraSymDescNameTabularHeader\endhead
 \glslongextraSymDescNameTabularFooter\endfoot
}
```

meTabularHeader

```
\newcommand{\glslongextraSymDescNameTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\symbolname &
 \glslongextraHeaderFmt\descriptionname &
 \glslongextraHeaderFmt\entryname\tabularnewline
 \midrule
}
```

meTabularFooter

```
\newcommand{\glslongextraSymDescNameTabularFooter}{%
 \bottomrule
}
```

g-sym-desc-name Three column style with symbol in the first column, description in the second and name in the third.

```
\newglossarystyle{long-sym-desc-name}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSymSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraNameAlign
        }}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraSymDescNameTabularFooter
      \end{tabular}%
    }%
   \renewcommand*{\glossaryheader}{\glslongextraSymDescNameTabularHeader}%
  \else
   \renewenvironment{theglossary}%
```

```
    {%
      \glspatchLToutput
      \glslongextraSymSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraNameAlign
        }}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraSymDescNameHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraSymbolFmt{##1} &
    \glslongextraDescFmt{##1} &
    \glslongextraNameFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubSymbolFmt{##1}{##2} &
     \glslongextraSubDescFmt{##1}{##2} &
     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

mDescNameHeader

```
\newcommand{\glslongextraLocationSymDescNameHeader}{%
 \glslongextraLocationSymDescNameTabularHeader\endhead
 \glslongextraLocationSymDescNameTabularFooter\endfoot
}
```

meTabularHeader

```
\newcommand{\glslongextraLocationSymDescNameTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\pagelistname &
 \glslongextraHeaderFmt\symbolname &
 \glslongextraHeaderFmt\descriptionname &
 \glslongextraHeaderFmt\entryname\tabularnewline
 \midrule
}
```

meTabularFooter

```
\newcommand{\glslongextraLocationSymDescNameTabularFooter}{%
 \bottomrule
}
```

c-sym-desc-name Four column style with location list, symbol, description and name.

```
\newglossarystyle{long-loc-sym-desc-name}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSymLocSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
          \expandonce\glslongextraLocationAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraNameAlign
        }}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraLocationSymDescNameTabularFooter
      \end{tabular}%
    }%
   \renewcommand*{\glossaryheader}{\glslongextraLocationSymDescNameTabularHeader}%
  \else
   \renewenvironment{theglossary}%
    {%
      \glspatchLToutput
      \glslongextraSymLocSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
          \expandonce\glslongextraLocationAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraNameAlign
        }}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraLocationSymDescNameHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraLocationFmt{##1}{##2} &
    \glslongextraSymbolFmt{##1} &
    \glslongextraDescFmt{##1} &
    \glslongextraNameFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubLocationFmt{##1}{##2}{##3} &
     \glslongextraSubSymbolFmt{##1}{##2} &
     \glslongextraSubDescFmt{##1}{##2} &
     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

scSymNameHeader

```
\newcommand{\glslongextraDescSymNameHeader}{%
 \glslongextraDescSymNameTabularHeader\endhead
 \glslongextraDescSymNameTabularFooter\endfoot
}
```

meTabularHeader

```
\newcommand{\glslongextraDescSymNameTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\descriptionname &
 \glslongextraHeaderFmt\symbolname &
 \glslongextraHeaderFmt\entryname\tabularnewline
 \midrule
}
```

meTabularFooter

```
\newcommand{\glslongextraDescSymNameTabularFooter}{%
 \bottomrule
}
```

g-desc-sym-name Three column style with description in the first column, symbol in the second and name in the third.

```
\newglossarystyle{long-desc-sym-name}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSymSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraNameAlign
        }}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraDescSymNameTabularFooter
      \end{tabular}%
    }%
   \renewcommand*{\glossaryheader}{\glslongextraDescSymNameTabularHeader}%
  \else
   \renewenvironment{theglossary}%
    {%
      \glspatchLToutput
      \glslongextraSymSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraNameAlign
        }}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraDescSymNameHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraDescFmt{##1} &
    \glslongextraSymbolFmt{##1} &
    \glslongextraNameFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubDescFmt{##1}{##2} &
     \glslongextraSubSymbolFmt{##1}{##2} &
     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

scSymNameHeader

```
\newcommand{\glslongextraLocationDescSymNameHeader}{%
 \glslongextraLocationDescSymNameTabularHeader\endhead
 \glslongextraLocationDescSymNameTabularFooter\endfoot
}
```

meTabularHeader

```
\newcommand{\glslongextraLocationDescSymNameTabularHeader}{%
 \toprule
 \glslongextraHeaderFmt\pagelistname &
 \glslongextraHeaderFmt\descriptionname &
 \glslongextraHeaderFmt\symbolname &
 \glslongextraHeaderFmt\entryname\tabularnewline
 \midrule
}
```

meTabularFooter

```
\newcommand{\glslongextraLocationDescSymNameTabularFooter}{%
 \bottomrule
}
```

c-desc-sym-name Four column style with location list, description, symbol and name.

```
\newglossarystyle{long-loc-desc-sym-name}%
{%
  \ifGlsLongExtraUseTabular
   \renewenvironment{theglossary}%
    {%
      \glslongextraSymLocSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
          \expandonce\glslongextraLocationAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraNameAlign
        }}%
      \@glslongextra@begintab
    }%
    {%
      \glslongextraLocationDescSymNameTabularFooter
      \end{tabular}%
    }%
   \renewcommand*{\glossaryheader}{\glslongextraLocationDescSymNameTabularHeader}%
  \else
   \renewenvironment{theglossary}%
    {%
      \glspatchLToutput
      \glslongextraSymLocSetDescWidth
      \edef\@glslongextra@begintab{%
        \noexpand\begin{longtable}{%
          \expandonce\glslongextraLocationAlign
          \expandonce\glslongextraDescAlign
          \expandonce\glslongextraSymbolAlign
          \expandonce\glslongextraNameAlign
        }}%
      \@glslongextra@begintab
    }%
    {\end{longtable}}%
   \renewcommand*{\glossaryheader}{\glslongextraLocationDescSymNameHeader}%
  \fi
  \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
  \renewcommand{\glossentry}[2]{%
    \glslongextraLocationFmt{##1}{##2} &
    \glslongextraDescFmt{##1} &
    \glslongextraSymbolFmt{##1} &
    \glslongextraNameFmt{##1}\tabularnewline
  }%
  \renewcommand{\subglossentry}[3]{%
     \glslongextraSubLocationFmt{##1}{##2}{##3} &
     \glslongextraSubDescFmt{##1}{##2} &
     \glslongextraSubSymbolFmt{##1}{##2} &
     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
  }%
  \ifglsnogroupskip
    \renewcommand*{\glsgroupskip}{}%
  \else
    \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
  \fi
}
```

Glossary

`bib2gls` A command line Java application that selects entries from a .bib file and converts them to glossary definitions (like `bibtex` but also performs hierarchical sorting and collation, thus omitting the need for `xindy` or `makeindex`). Further details at: `http://www.dickimaw-books.com/software/bib2gls/`..

First use The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* First use flag & First use text

First use flag A conditional that determines whether or not the entry has been used according to the rules of first use.

First use text The text that is displayed on first use, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

`makeindex` An indexing application.

`xindy` An flexible indexing application with multilingual support written in Perl.

Change History

- 0.1 (2015-11-22)
  - General: Initial experimental release 5
- 0.2 (2015-11-30)
  - `\Glsfmtshort`: new 332
  - `\glsfmtshort`: new 331
  - `\Glsfmtshortpl`: new 332
  - `\glsfmtshortpl`: new 331
  - `short`: switched inline full form to short (long) 234
- 0.3 (2015-12-02)
  - `\@ACRlong`: added redefinition 86
  - `\@ACRlongpl`: added redefinition 87
  - `\@ACRshort`: added redefinition 84
  - `\@ACRshortpl`: added redefinition 85
  - `\@Acrlong`: added redefinition 85
  - `\@Acrlongpl`: added redefinition 86
  - `\@Acrshort`: added redefinition 83
  - `\@Acrshortpl`: added redefinition 84
  - `\@GLSdesc@`: added redefinition 79
  - `\@GLSdescplural@`: added redefinition 80
  - `\@GLSfirst@`: added redefinition 77
  - `\@GLSfirstplural@`: added redefinition 78
  - `\@GLSname@`: added redefinition 79
  - `\@GLSplural@`: added redefinition 77
  - `\@GLSsymbol@`: added redefinition 80
  - `\@GLSsymbolplural@`: added redefinition 81
  - `\@GLStext@`: added redefinition 76
  - `\@GLSuseri@`: added redefinition 81
  - `\@GLSuserii@`: added redefinition 82
  - `\@GLSuseriii@`: added redefinition 82
  - `\@GLSuseriv@`: added redefinition 82
  - `\@GLSuserv@`: added redefinition 82
  - `\@GLSuservi@`: added redefinition 83
  - `\@Glsdesc@`: added redefinition 79
  - `\@Glsdescplural@`: added redefinition 80
  - `\@Glsfirst@`: added redefinition 77
  - `\@Glsfirstplural@`: added redefinition 78
  - `\@Glsname@`: added redefinition 79
  - `\@Glsplural@`: added redefinition 77
  - `\@Glssymbol@`: added redefinition 80
  - `\@Glssymbolplural@`: added redefinition 81
  - `\@Glstext@`: added redefinition 76
  - `\@Glsuseri@`: added redefinition 81
  - `\@Glsuserii@`: added redefinition 81
  - `\@Glsuseriii@`: added redefinition 82
  - `\@Glsuseriv@`: added redefinition 82
  - `\@Glsuserv@`: added redefinition 82
  - `\@Glsuservi@`: added redefinition 83
  - `\@acrlong`: added redefinition 85
  - `\@acrlongpl`: added redefinition 86
  - `\@acrshort`: added redefinition 83
  - `\@acrshortpl`: added redefinition 84
  - `\@gls@field@link`: added optional argument 67
  - `\@glsdescplural@`: added redefinition 79
  - `\@glsfirst@`: added redefinition 76
  - `\@glsfirstplural@`: added redefinition 78
  - `\@glsplural@`: added redefinition 77
  - `\@glssymbolplural@`: added redefinition 80
  - `\@glsxtr@defaultnoglossarywarning`: new 139
  - `\@glsxtr@field@linkdefs`: new 75
  - `\@glsxtr@insertdots`: new 202
  - `\@print@glossary`: added redefinition 136
  - `\glsabbrvdefaultfont`: renamed from `\abbrvdefaultfont` 208
  - `\glsaccessdesc`: new 166
  - `\glsaccessdescplural`: new 166
  - `\glsaccessfirst`: new 163
  - `\glsaccessfirstplural`: new 164
  - `\Glsaccesslong`: new 168
  - `\glsaccesslong`: new 168
  - `\glsaccessname`: new 162
  - `\glsaccessplural`: new 163
  - `\Glsaccessshort`: new 167
  - `\glsaccessshort`: new 167
  - `\Glsaccessshortpl`: new 168

\glsaccessshortpl: new 168
\glsaccesssymbol: new 165
\glsaccesssymbolplural: new 165
\glsaccesstext: new 162
\glsentryfmt: added check for short 67
\glslongpltok: new 202
\glsshortpltok: new 202
\glsxtr@newabbreviation: fixed family name in \setkeys 204
\glsxtrdiscardperiod: added check for plural 199
\GLSxtrlongpl: new 218
\Glsxtrlongpl: new 218
\glsxtrlongpl: new 217
\glsxtrNoGlossaryWarning: new 23
\glsxtrpostlinkAddDescOnFirstUse: new 198
\glsxtrpostlinkAddSymbolOnFirstUse: new 199
\glsxtrpostlinkendsentence: new 198
\GLSxtrshortpl: new 217
\Glsxtrshortpl: new 216
\glsxtrshortpl: new 215
short-long-desc: fixed name to use \glslabeltok 229
long-short-desc: fixed name to use \glslabeltok 227

0.4 (2015-12-03)
  \@glsxtr@doabbreviationsdef: added redefinition of \acronymtype 20
  \Glsfmtshort: changed to use \Glsxtrshort 332
  \glsfmtshort: changed to use \glsxtrshort 331
  \Glsfmtshortpl: changed to use \glsxtrshortpl 332
  \glsfmtshortpl: changed to use \glsxtrshortpl 331
  \glsxtrifemptyglossary: new 31
  \glsxtrnewnumber: added extra argument 180
  \glsxtrnewsymbol: added extra argument 180
  \MakeAcronymsAbbreviations: set the default type to \acronymtype 121
  \newterm: fixed name argument 179

0.5 (2015-12-07)
  \@cGLS: new 112
  \@cGLS@: new 112
  \@cGLSpl: new 113
  \@cGLSpl@: new 113
  \@glsxtr@setentrycountunsetattr: new 108
  \cGLS: new 112
  \cGLSformat: new 112
  \cGLSpl: new 113
  \cGLSplformat: new 113
  \GlossariesExtraWarningNoLine: new 18
  \glsenableentrycount: new 108
  \glsfirstabbrvdefaultfont: new 208
  \glsfirstlongdefaultfont: new 208
  \Glsfmtfirst: new 334
  \glsfmtfirst: new 334
  \Glsfmtfirstpl: new 335
  \glsfmtfirstpl: new 334
  \Glsfmtplural: new 334
  \glsfmtplural: new 333
  \Glsfmtshort: changed to use \Glsxtrtitleshort 332
    renamed from \Glsentryfmtshort 332
  \glsfmtshort: changed to use \glsxtrtitleshort 331
    renamed from \glsentryfmtshort 331
  \Glsfmtshortpl: changed to use \Glsxtrtitleshortpl 332
    renamed from \Glsentryfmtshortpl 332
  \glsfmtshortpl: changed to use \glsxtrtitleshortpl 331
    renamed from \glsentryfmtshortpl 331
  \Glsfmttext: new 333
  \glsfmttext: new 333
  \glshasattribute: new 177
  \glshascategoryattribute: new 176
  \glsxtremsuffix: new 270
  \GlsXtrEnableEntryCounting: new 107
  \glsxtrifcounttrigger: new 110
  \glsxtrscfont: new 242
  \glsxtrscsuffix: new 242
  \glsxtrsmfont: new 256
  \glsxtrsmsuffix: new 257
  short-em: new 278
  short-em-desc: new 279
  short-em-footnote: new 288
  short-em-long: new 274
  short-em-long-desc: new 275

- `short-em-postfootnote`: new 290
- `short-sc-footnote`: new 253
- `short-sc-postfootnote`: new 254
- `short-sm`: new 260
- `short-sm-desc`: new 261
- `short-sm-footnote`: new 267
- `short-sm-long`: new 258
- `short-sm-long-desc`: new 259
- `short-sm-postfootnote`: new 268
- `long-noshort-em`: new 281
- `long-noshort-em-desc`: new 285
- `long-noshort-sm`: new 263
- `long-noshort-sm-desc`: new 265
- `long-short-em`: new 271
- `long-short-em-desc`: new 272
- `long-short-sm`: new 257
- `long-short-sm-desc`: new 258

0.5.1 (2015-12-02)
- `\Glsaccesstext`: new 163

0.5.1 (2015-12-07)
- `\@glsxtr@doaccsupp`: new 23
- General: removed `\ifglsxtruseuchead` 322
- `\Glsaccessdesc`: new 166
- `\Glsaccessdescplural`: new 167
- `\Glsaccessfirst`: new 164
- `\Glsaccessfirstplural`: new 164
- `\Glsaccessname`: new 162
- `\Glsaccessplural`: new 163
- `\Glsaccesssymbol`: new 165
- `\Glsaccesssymbolplural`: new 165
- `\Glsxtrheadfirst`: now uses headuc attribute 326
- `\glsxtrheadfirst`: now uses headuc attribute 326
- `\Glsxtrheadfirstplural`: now uses headuc attribute 327
- `\glsxtrheadfirstplural`: now uses headuc attribute 327
- `\Glsxtrheadplural`: now uses headuc attribute 326
- `\glsxtrheadplural`: now uses headuc attribute 325
- `\Glsxtrheadshort`: now uses headuc attribute 323
- `\glsxtrheadshort`: now uses headuc attribute 322
- `\Glsxtrheadshortpl`: now uses headuc attribute 323
- `\glsxtrheadshortpl`: now uses headuc attribute 322
- `\Glsxtrheadtext`: now uses headuc attribute 325
- `\glsxtrheadtext`: now uses headuc attribute 324
- `short-em-footnote`: switch off regular attribute if set 289
- `short-long`: switch off regular attribute if set 228
- `short-long-desc`: switch off regular attribute if set 229
- `short-sc-footnote`: switch off regular attribute if set 253
- `short-sm-footnote`: switch off regular attribute if set 267
- `long-short`: switch off regular attribute if set 226
- `long-short-desc`: switch off regular attribute if set 227
- `long-short-sc-desc`: switch off regular attribute if set 244
- `footnote`: switch off regular attribute if set 230
- `postfootnote`: switch off regular attribute if set 232

0.5.2 (2015-12-08)
- `\@GLSdesc@`: added accessibility support 79
- `\@GLSdescplural@`: added accessibility support 80
- `\@GLSfirst@`: added accessibility support 77
- `\@GLSfirstplural@`: added accessibility support 78
- `\@GLSname@`: added accessibility support 79
- `\@GLSplural@`: added accessibility support 77
- `\@GLSsymbol@`: added accessibility support 80
- `\@GLSsymbolplural@`: added accessibility support 81
- `\@GLStext@`: added accessibility support 76
- `\@Glsdesc@`: added accessibility support 79
- `\@Glsdescplural@`: added accessibility support 80
- `\@Glsfirst@`: added accessibility support 77
- `\@Glsfirstplural@`: added accessibility support 78

\@Glsname@: add accessibility support . . 79
\@Glsplural@: added accessibility support 77
\@Glssymbol@: added accessibility support 80
\@Glssymbolplural@: added accessibility support 81
\@Glstext@: added accessibility support 76
\@glsdesc@: added accessibility support 79
\@glsdescplural@: added accessibility support 79
\@glsfirst@: added accessibility support 76
\@glsfirstplural@: added accessibility support 78
\@glsname@: added accessibility support 79
\@glsplural@: added accessibility support 77
\@glssymbol@: added accessibility support 80
\@glssymbolplural@: added accessibility support 80
\@glstext@: added accessibility support 76
\@glsxtr@activate@initialtagging: new 196
\@glsxtr@do@titlecaps@warn: new . 196
\@glsxtr@tag: new 196
General: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded 162
  removed \glsxtrabbrvfmt 219
\glossaryentrynumbers: added 64
\Glossentrydesc: added 194
\Glossentryname: added 185
\Glossentrysymbol: added 194
\glossentrysymbol: added 194
\GLSaccessdesc: new 166, 174
\GLSaccessdescplural: new . . . 167, 174
\GLSaccessfirst: new 164, 173
\GLSaccessfirstplural: new . . 164, 173
\GLSaccesslong: new 168, 175
\GLSaccesslongpl: new 169, 175
\Glsaccesslongpl: new 169
\glsaccesslongpl: new 169
\GLSaccessname: new 162, 172
\GLSaccessplural: new 163, 173
\GLSaccessshort: new 167, 174
\GLSaccessshortpl: new 168, 175
\GLSaccesssymbol: new 165, 173
\GLSaccesssymbolplural: new . 166, 174
\GLSaccesstext: new 163, 172
\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here 67
\GlsXtrEnableInitialTagging: new 194
\glsxtrfieldtitlecase: new 181
\GlsXtrFormatLocationList: new . . . 64
\glsxtrnewabbrevpresetkeyhook: new 206
\glsxtrtagfont: new 196
\KV@printgloss@nonumberlist: added 66
\mfu@checkword@do: added 195
\setabbreviationstyle: added check for post-definition style switch 222
0.5.3 (2015-12-09)
  \@glsxtr@autoindex@at: new 192
  \@glsxtr@autoindex@encap: new . . . 192
  \@glsxtr@autoindex@esc: new 193
  \@glsxtr@autoindex@level: new . . . 192
  \@glsxtr@autoindex@setname: new . 190
  \@glsxtr@doabbreviationsdef: new . 20
  General: removed \GlsXtrNoGlsWarningNoAutoMakeMain 138
  \glsdescwidth: added 63
  \glspagelistwidth: added 64
  \glsxtrdoautoindexname: new 190
  \glsxtrpostnamehook: new 187
  \if@glsxtr@format@override: new . 189
  \ProvidesGlossariesExtraLang: new 338
  \RequireGlossariesExtraLang: new 337
0.5.4 (2015-12-15)
  \@@newglossaryentry@defunitcounters: new 114
  \@GLSxtr@p@acrlong@: new 99
  \@GLSxtr@p@acrlongpl@: new 99
  \@GLSxtr@p@acrshort@: new 98
  \@GLSxtr@p@acrshortpl@: new 99
  \@GLSxtr@p@long@: new 98
  \@GLSxtr@p@longpl@: new 98
  \@GLSxtr@p@plural@: new 97
  \@GLSxtr@p@short@: new 97
  \@GLSxtr@p@shortpl@: new 98
  \@GLSxtr@p@text@: new 97
  \@GlsXtrEnableOnTheFly: new 60
  \@Glsxtr: new 60
  \@Glsxtr@p@acrlong@: new 99
  \@Glsxtr@p@acrlongpl@: new 99

\@Glsxtr@p@acrshort@: new 98
\@Glsxtr@p@acrshortpl@: new 99
\@Glsxtr@p@long@: new 98
\@Glsxtr@p@longpl@: new 98
\@Glsxtr@p@plural@: new 97
\@Glsxtr@p@short@: new 97
\@Glsxtr@p@shortpl@: new 98
\@Glsxtr@p@text@: new 97
\@Glsxtrpl: new 61
\@alt@gls@hyp@opt: new 93
\@gls@alt@hyp@opt: new 92
\@gls@alt@hyp@opt@char: new 93
\@gls@alt@hyp@opt@keys: new 93
\@gls@increment@currunitcount: new 114
\@gls@local@increment@currunitcount: new 115
\@gls@setdefault@glslink@opts: new 90
\@glsxtr: new 60
\@glsxtr@addunitcounter: new 114
\@glsxtr@currunitcount: new 115
\@glsxtr@ifunitcounter: new 114
\@glsxtr@p@acrlong@: new 99
\@glsxtr@p@acrlongpl@: new 99
\@glsxtr@p@acrshort@: new 98
\@glsxtr@p@acrshortpl@: new 99
\@glsxtr@p@long@: new 98
\@glsxtr@p@longpl@: new 98
\@glsxtr@p@plural@: new 97
\@glsxtr@p@short@: new 97
\@glsxtr@p@shortpl@: new 97
\@glsxtr@p@text@: new 97
\@glsxtr@prevunitcount: new 115
\@glsxtr@setentryunitcountunsetattr: new 119
\@glsxtr@unitcountlist: new 114
\@glsxtrpl: new 61
\@newglossaryentryposthook: added empty see value if not set and added ‘see’ to field key map 49
\@sGlsXtrEnableOnTheFly: new 59
\cGlsformat: added 113
\cglsformat: added 113
\cGlsplformat: added 113
\cglsplformat: added 113
\glsdisablehyper: added 95
\glsdohyperlink: added 94
\glsdonohyperlink: added 96
\glsenableentryunitcount: new 116
\glshasattribute: added check for entry’s existence 177
\glsifattribute: added check for entry’s existence 177
\glspostlinkhook: added existence check 197
\Glsxtr: new 60
\glsxtr: new 60
\glsxtrcat: new 60
\glsxtrdowrglossaryhook: new 92
\GlsXtrEnableEntryUnitCounting: new 119
\GlsXtrEnableOnTheFly: new 59
\Glsxtrpl: new 61
\glsxtrpl: new 61
\glsxtrpostlocalreset: new 107
\glsxtrpostlocalunset: new 106
\glsxtrpostreset: new 106
\glsxtrpostunset: new 105
\glsxtrprotectlinks: new 96
\GlsXtrSetAltModifier: new 93
\GlsXtrSetDefaultGlsOpts: new 91
\glsxtrstarflywarn: new 60
\GlsXtrWarning: new 61
\MakeAcronymsAbbreviations: now disables \setacronymstyle 121
1.0 (2016-01-24)
\@glsxtr@autoindexcrossrefs: new 17
\@glsxtr@idx@displaynumberlist: new 130
\@glsxtr@idx@entrynumberlist: new 131
\@glsxtr@noidx@displaynumberlist: new 130
\@glsxtr@noidx@entrynumberlist: new 131
\@glsxtr@noidx@numberlistloop: new 130
\@glsxtr@reg@glosslist: new 122
\makeglossaries: new 122
1.01 (2016-02-02)
\glsxtrdiscardperiod: added check for first use 199
short-desc: fixed typo in \glsxtrinlinefullformat and added missing second argument 236
1.02 (2016-04-25)
\@glsxtr@current@style: new 62
\Glsfmtfull: new 336

- \glsfmtfull: new 336
- \Glsfmtfullpl: new 337
- \glsfmtfullpl: new 337
- \Glsfmtlong: new 335
- \glsfmtlong: new 335
- \Glsfmtlongpl: new 336
- \glsfmtlongpl: new 336
- \Glsxtrheadfull: new 330
- \glsxtrheadfull: new 329
- \Glsxtrheadfullpl: new 331
- \glsxtrheadfullpl: new 330
- \Glsxtrheadlong: new 328
- \glsxtrheadlong: new 328
- \Glsxtrheadlongpl: new 329
- \glsxtrheadlongpl: new 328
- \Glsxtrtitlefull: new 330
- \glsxtrtitlefull: new 330
- \Glsxtrtitlefullpl: new 331
- \glsxtrtitlefullpl: new 330
- \Glsxtrtitlelong: new 329
- \glsxtrtitlelong: new 328
- \Glsxtrtitlelongpl: new 329
- \glsxtrtitlelongpl: new 328
- \ifglsxtrinsertinside: new 225
- postfootnote: added redef of \glsxtrsetupfulldefs 233
- stylemods: new 24

1.03 (2016-04-27)
- \@GLSfirstplural@: bug fix: misspelt cs name 78
- \@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural 77
- \@Glsfirstplural@: bug fix: misspelt cs name 78
- \@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural 77
- \@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural 77
- \glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl 328
- \glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl 323

1.04 (2015-04-30)
- short-em-footnote: renamed from “footnote-em” 288

1.04 (2016-05-02)
- \@@glsxtrpostloctag: new 66
- \@GLSdesc@: set abbreviation and regular format 79
- \@GLSdescplural@: set abbreviation and regular format 80
- \@GLSfirst@: set abbreviation format 77
- \@GLSfirstplural@: set abbreviation and regular format 78
- \@GLSname@: set abbreviation and regular format 79
- \@GLSplural@: set abbreviation and regular format 77
- \@GLSsymbol@: set regular format 80
- \@GLSsymbolplural@: set regular format 81
- \@GLStext@: set abbreviation and regular format 76
- \@GLSuseri@: set regular format 81
- \@GLSuserii@: set regular format 82
- \@GLSuseriii@: set regular format 82
- \@GLSuseriv@: set regular format 82
- \@GLSuserv@: set regular format 82
- \@GLSuservi@: set regular format 83
- \@Glsdesc@: set abbreviation and regular format 79
- \@Glsdescplural@: set abbreviation and regular format 80
- \@Glsfirst@: set abbreviation and regular format 77
- \@Glsfirstplural@: set abbreviation and regular format 78
- \@Glsname@: set abbreviation and regular format 79
- \@Glsplural@: set abbreviation and regular format 77
- \@Glssymbol@: set regular format 80
- \@Glssymbolplural@: set regular format 81
- \@Glstext@: set abbreviation and regular format 76
- \@Glsuseri@: set regular format 81
- \@Glsuserii@: set regular format 81
- \@Glsuseriii@: set regular format 82
- \@Glsuseriv@: set regular format 82
- \@Glsuserv@: set regular format 82
- \@Glsuservi@: set regular format 83
- \@gls@preglossaryhook: added check for entry’s existence 196
- \@glsdesc@: set abbreviation and regular format 79
- \@glsdescplural@: set abbreviation and regular format 79
- \@glsfirst@: set abbreviation and regular format 76

\@glsfirstplural@: set abbreviation and regular format 78
\@glsname@: set abbreviation and regular format 79
\@glsplural@: set abbreviation and regular format 77
\@glssymbol@: set regular format 80
\@glssymbolplural@: set regular format 80
\@glstext@: set abbreviation and regular format 76
\@glsxtr@deprecated@abbrstyle: new 224
\@glsxtr@do@style: new 25
\@glsxtr@doloctag: new 66
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand 131
\@glsxtr@pagestag: new 66
\@glsxtr@pagetag: new 66
\@glsxtr@preloctag: new 66
\@glsxtrpostloctag: new 66
\@glsxtrpreloctag: new 65
\glossentrydesc: added glossdescfont attribute check 181
\Glossentryname: added glossnamefont attribute check 185
\glossentryname: added glossnamefont attribute check 183
  moved post name hook inside condition 185
\glsabbrvemfont: new 270
\glsabbrvuserfont: new 292
\glsfirstabbrvemfont: new 270
\glsfirstabbrvuserfont: new 292
\glsfirstlongemfont: new 271
\glsfirstlonguserfont: new 293
\glsifnotregularcategory: new 178
\glslongdefaultfont: new 208
\glslongemfont: new 271
\glslongfont: new 208
\glslonguserfont: new 293
\glsxtrassignfieldfont: new 75
\GlsXtrEnablePreLocationTag: new 65
\glsxtrfirstscfont: new 242
\glsxtrfirstsmfont: new 256
\glsxtrlongshortdescsort: new 227
\glsxtrpostnamehook: added category check 187
\glsxtrregularfont: new 67
\glsxtruserfield: new 292
\glsxtruserparen: new 292
\glsxtrusersuffix: new 293
\GlsXtrWarnDeprecatedAbbrStyle: new 224
short-em-long-em: new 276
short-em-long-em-desc: new 277
short-em-nolong: new 279
short-em-nolong-desc: new 280
short-em-postfootnote: renamed from “postfootnote-em” 290
short-footnote: new 232
short-long-user: new 299
short-long-user-desc: new 301
short-nolong: new 235
short-nolong-desc: new 237
short-postfootnote: new 234
short-sc-footnote: renamed from “footnote-sc” 253
short-sc-nolong: new 247
short-sc-nolong-desc: new 249
short-sc-postfootnote: renamed from “postfootnote-sc” 254
short-sm-footnote: renamed from “footnote-sm” 267
short-sm-nolong: new 261
short-sm-nolong-desc: new 263
short-sm-postfootnote: renamed from “postfootnote-sm” 268
\letabbreviationstyle: new 224
\newabbreviationstyle: bug fix: corrected test for existence 223
long-em-noshort-em: new 283
long-em-noshort-em-desc: new 286
long-em-short-em: new 272
long-em-short-em-desc: new 273
long-noshort: new 241
long-noshort-desc: new 240
long-noshort-em: renamed from “long-em” 281
long-noshort-em-desc: renamed from “long-desc-em” 285
long-noshort-sc: renamed from “long-sc” 249
long-noshort-sc-desc: renamed from “long-desc-sc” 251
long-noshort-sm: renamed from “long-sm” 263
long-noshort-sm-desc: renamed from \long-desc-sm 265

  `long-short-user`: new . . . 293
  `long-short-user-desc`: new . . . 299
  `\renewabbreviationstyle`: new . . . 224
  `style`: new . . . 25
1.05 (2016-06-10)
  `\eglssetwidest`: new . . . 402
  `\glsFindWidestAnyName`: new . . . 404
  `\glsFindWidestAnyNameLocation`: new . . . 410
  `\glsFindWidestAnyNameSymbol`: new . . . 407
  `\glsFindWidestAnyNameSymbolLocation`: new . . . 409
  `\glsFindWidestLevelTwo`: new . . . 406
  `\glsFindWidestUsedAnyName`: new . . . 404
  `\glsFindWidestUsedAnyNameLocation`: new . . . 409
  `\glsFindWidestUsedAnyNameSymbol`: new . . . 407
  `\glsFindWidestUsedAnyNameSymbolLocation`: new . . . 408
  `\glsFindWidestUsedLevelTwo`: new . . . 405
  `\glsFindWidestUsedTopLevelName`: new . . . 403
  `\glsfirstlongfootnotefont`: new . . . 230
  `\glsgetwidestname`: new . . . 403
  `\glsgetwidestsubname`: new . . . 403
  `\glslongfootnotefont`: new . . . 230
  `\glsxtrAltTreeIndent`: new . . . 401
  `\glsxtralttreeInit`: new . . . 402
  `\glsxtrAltTreePar`: new . . . 401
  `\glsxtrAltTreeSetHangIndent`: new . . . 411
  `\glsxtrAltTreeSetSubHangIndent`: new . . . 411
  `\glsxtralttreeSubSymbolDescLocation`: new . . . 401
  `\glsxtralttreeSymbolDescLocation`: new . . . 401
  `\glsxtrComputeTreeIndent`: new . . . 410
  `\glsxtrComputeTreeSubIndent`: new . . . 411
  `\glsxtrtreetopindent`: new . . . 402
  `short-em-long`: fixed incorrect font used by long form . . . 275
  `\xglssetwidest`: new . . . 402
1.06 (2016-06-18)
  `\@glsdoifexistsorwarn`: new . . . 17
  `\@glsxtr@docdefval`: new . . . 16
  `\@glsxtr@usesee`: new . . . 49
  General: disabled docdef key at the start of the document . . . 30
    docdef option changed to choice . . . 16
  `\glsxtr@usesee`: new . . . 49
  `\glsxtrusesee`: new . . . 49
  `\glsxtruseseeformat`: new . . . 49
  `\if@glsxtrdocdefrestricted`: new . . . 17
1.07 (2016-08-15)
  `\@@glsxtrp`: new . . . 99
  `\@GLSfirst@`: added check for nohyperfirst attribute . . . 77
  `\@GLSfirstplural@`: added check for nohyperfirst attribute . . . 78
  `\@GLSxtrp`: new . . . 100
  `\@Glsfirst@`: added check for nohyperfirst attribute . . . 77
  `\@Glsfirstplural@`: added check for nohyperfirst attribute . . . 78
  `\@Glsxtrp`: new . . . 100
  `\@gls@preglossaryhook`: added `\glossxtrsetpopts` . . . 197
  `\@glsfirst@`: added check for nohyperfirst attribute . . . 76
  `\@glsfirstplural@`: added check for nohyperfirst attribute . . . 78
  `\@glsxtrinmark`: new . . . 320
  `\@glsxtrnotinmark`: new . . . 320
  `\@glsxtrp`: new . . . 100
  `\@glsxtrp@opt`: new . . . 99
  `\glossxtrsetpopts`: new . . . 99
  `\glsps`: new . . . 102
  `\glspt`: new . . . 102
  `\glsxtr@entry@p`: new . . . 101
  `\glsxtrabbrvfootnote`: new . . . 230
  `\glsxtrchecknohyperfirst`: new . . . 76
  `\glsxtrfieldtitlecasecs`: new . . . 181
  `\glsxtrifinmark`: new . . . 319
  `\GLSxtrp`: new . . . 103
  `\Glsxtrp`: new . . . 102
  `\glsxtrp`: new . . . 101
  `\glsxtrsetpopts`: new . . . 99
  `short-long-desc`: added text key . . . 229
    fixed misspelling of `\glsabbrvfont` in plural key . . . 229
  `long-short-desc`: added missing text key . . . 227
    fixed misspelling of `\glsabbrvfont` . . . 227
  `footnote`: changed first forms to use `\glsfirstlongfootnotefont` . . . 230
  `postfootnote`: removed `\footnote` from first keys . . . 232

switched from `\glsfirstlongfont` to `\glsfirstlongfootnotefont` 233
`\RestoreAcronyms`: modified `\@gls@link@checkfirsthyper` to set `\glsxtrifwasfirstuse` 122
1.08 (2016-12-13)
`\@@glsxtr@record`: new 8
`\@GLS@`: added `\@glsxtr@record` 68
`\@GLSpl@`: added `\@glsxtr@record` 68
`\@Gls@`: added `\@glsxtr@record` 68
`\@Glspl@`: added `\@glsxtr@record` 68
`\@gls@`: added `\@glsxtr@record` 68
`\@gls@@link@`: added `\@glsxtr@record` 69
`\@gls@field@link`: added `\@glsxtr@record` 67
`\@gls@saveentrycounter`: new 29
`\@glsdisp`: added `\@glsxtr@record` 69
`\@glspl@`: added `\@glsxtr@record` 68
`\@glsxtr@dorecord`: new 10
`\@glsxtr@err@undefaction`: new 6
`\@glsxtr@record`: new 7
`\@glsxtr@warn@onexistsordo`: new 6
`\@glsxtr@warn@undefaction`: new 6
`\@print@unsrt@glossary`: new 147
`record`: added record package option 15
`\glsadd`: added `\@glsxtr@record` 74
`\glsdoifexists`: now defines `\glslabel` 47
`\glsxtr@@do@wrglossary`: new 29
`\glsxtr@addloclistfield`: new 13
`\glsxtr@indexonly@saveentrycounter`: new 13
`\glsxtr@record`: new 144
`\glsxtr@resource`: new 142
`\glsxtr@saveentrycounter`: new 29
`\glsxtr@setup@record`: new 13
`\glsxtrassignfieldfont`: added check for existence 75
`\glsxtrresourcefile`: new 141
`\printunsrtglossaries`: new 147
`\printunsrtglossary`: new 147
1.09 (2016-12-16)
`\@glsxtr@gettype`: new 129
`\@glsxtr@mixed@assign@sortkey`: new 129
`\@printglossary`: redefined to save options 128
`\glsxtr@makeglossaries`: new 129
1.10 (2016-12-17)
`\@GLSpl@`: fixed bug caused by typo in command name 68
1.11 (2017-01-19)
`\@glsxtr@do@redef@forglsentries`: new 6
`\@glsxtr@noidx@do`: new 151
`\@glsxtr@redef@forglsentries`: new 6
`\@glsxtr@shortcutsval`: new 22
`\@glsxtr@unsrt@getgrouptitle`: new 150
`\@print@noidx@glossary`: added redefinition 133
`\glsxtr@addloclistfield`: added group key 14
added location key 14
`\glsxtr@fields`: new 142
`\glsxtr@linkprefix`: new 142
`\glsxtr@org@newignoredglossary`: new 44
`\glsxtr@s@newignoredglossary`: new 45
`\glsxtr@shortcutsval`: new 142
`\glsxtr@texencoding`: new 142
`\glsxtr@writefields`: new 142
`\GlsXtrLoadResources`: new 142
`\glsxtrpageref`: new 41
`\glsxtrresourcefile`: changed extension to .glstex 141
`\newignoredglossary`: added starred version 44
1.12 (2017-02-03)
`\@@glsxtr@recordcounter`: new 12
`\@gls@preglossaryhook`: check for definition 196
`\@glsxtr@counterrecordhook`: new 144
`\@glsxtr@display@loc`: new 134
`\@glsxtr@docounterrecord`: new 145
`\@glsxtr@longnewglossaryentry`: new 43
`\@glsxtr@noop@recordcounter`: new 12
`\@glsxtr@op@recordcounter`: new 13
`\@glsxtr@provide@storagekey`: new 31
`\@glsxtr@s@longnewglossaryentry`: new 43
`\@glsxtrentryfmt`: new 34
`\@glsxtrindexaliased`: new 91
`\@glsxtrsetaliasnoindex`: new 90
`\@newglossaryentryposthook`: added check for alias key 55
`\@no@glsxtrindexaliased`: new 91

\@printunsrtglossary: new 147
General: added target key to printgloss family 128
\apptoglossarypreamble: new 42
\csGlsXtrLetField: new 38
\eGlsXtrSetField: new 38
\gGlsXtrSetField: new 38
\glsdohyperlink: added check for alias field 94
\glsnoidxdisplayloc: added redefinition 133
\glssettoctitle: added patch 45
\glsxtr@counterrecord: new 144
\glsxtr@langtag: new 142
\glsxtr@newabbreviation: new 204
\glsxtr@org@newignoredglossary: Added check for existence 44
\glsxtr@pluralsuffixes: new 142
\glsxtr@provideignoredglossary: new 46
\glsxtr@s@newignoredglossary: Added check for existence 45
\glsxtr@s@provideignoredglossary: new 46
\glsxtrabbrvpluralsuffix: new 208
\glsxtralias: new 55
\glsxtrcopytoglossary: new 47
\glsxtrdeffield: new 37
\glsxtrdisplayendloc: new 134
\glsxtrdisplayendlochook: new 135
\glsxtrdisplaysingleloc: new 134
\glsxtrdisplaystartloc: new 134
\glsxtredeffield: new 37
\glsxtrentryfmt: new 34
\glsxtrfielddolistloop: new 35
\glsxtrfieldforlistloop: new 35
\glsxtrfieldifinlist: new 35
\glsxtrfieldlistadd: new 34
\glsxtrfieldlisteadd: new 34
\glsxtrfieldlistgadd: new 34
\glsxtrfieldlistxadd: new 34
\glsxtrfieldxifinlist: new 35
\glsxtrfmt: new 32
\GlsXtrFmtDefaultOptions: new 32
\GlsXtrFmtField: new 32
\glsxtrifkeydefined: new 31
\glsxtrindexaliased: new 91
\GlsXtrLetField: new 38
\GlsXtrLetFieldToField: new 38
\GlsXtrLoadResources: removed restriction on only one per document 142
\glsxtrlocrangefmt: new 135
\glsxtrpostlongdescription: new 44
\glsxtrprovidestoragekey: new 31
\GlsXtrRecordCounter: new 144
\glsxtrresourcecount: new 142
\glsxtrresourcefile: added catcode change for @ 141
\glsxtrsetaliasnoindex: new 90
\GlsXtrSetField: new 38
\glsxtrsetfieldifexists: new 37
\glsxtrunsrtdo: new 150
\Glsxtrusefield: new 37
\glsxtrusefield: new 37
short-postlong-user: new 296
short-postlong-user-desc: new 298
\longnewglossaryentry: added starred version 43
long-postshort-user: new 294
long-postshort-user-desc: new 296
postdot: new 18
\pretoglossarypreamble: new 42
\print@noop@unsrtglossaryunit: new 150
\print@op@unsrtglossaryunit: new 149
\printunsrtglossary: added starred form 147
\printunsrtglossaryhandler: new 149
\printunsrtglossaryunit: new 13
\printunsrtglossaryunitsetup: new 149
\provideignoredglossary: new 46
\s@glsxtr@provide@storagekey: new 32
\s@printunsrtglossary: new 147
\xGlsXtrSetField: new 38

1.13 (2017-02-07)
  \@glsdisp: removed \@glsxtr@org@glsdisp 69
  \glsxtrsetaliasnoindex: switched to \providecommand 90

1.14 (2017-04-18)
  \@gls@link: added redefinition 72
  \@gls@noidx@getgrouptitle: new 131
  \@gls@removespaces: new 135
  \@glsxtr@do@automake@err: new 144
  \@glsxtr@org@gloautosee: new 28
  \@glsxtr@record: added third arg 7
  \@glsxtr@recordsee: new 13

General: added \glsadd option theHvalue 74
  added \glsadd option thevalue 74
\glsdisablehyper: added redefinition 95
\glsenableentrycount: fixed assignment of \@cGls@ 109
\glsenableentryunitcount: fixed assignment of \@cGls@ 117
\glsnavigation: new 132
\glsxtr@org@getgrouptitle: new 132
\glsxtr@recordsee: new 7
\glsxtr@writefields: added check for automake 144
\glsxtrdisplayendloc: added check for empty format 134
\glsxtrgetgrouptitle: new 132
\glsxtrinitwrgloss: new 69
\glsxtrlocationhyperlink: new 135
\glsxtrsetgrouptitle: new 132
\glsxtrsupphypernumber: new 136
\ifglsxtrwrglossbefore: new 70
1.15 (2017-05-10)
\@glsxtr@dorecord: corrected premature expansion of \@glslocref 10
short-em-long-em: fixed spelling of \glsabbrvfont 276
short-long: fixed spelling of \glsabbrvfont 228
short-long-user: fixed spelling of \glsabbrvfont 300
short-postlong-user: fixed spelling of \glsabbrvfont 297
short-postlong-user-desc: fixed spelling of \glsabbrvfont 298
long-em-short-em: fixed spelling of \glsabbrvfont 273
long-postshort-user: fixed spelling of \glsabbrvfont 294
long-postshort-user-desc: fixed spelling of \glsabbrvfont 296
long-short: fixed spelling of \glsabbrvfont 226
long-short-user: fixed spelling of \glsabbrvfont 293
footnote: fixed spelling of \glsabbrvfont 230
postfootnote: fixed spelling of \glsabbrvfont 232
1.16 (2017-06-15)
\@glo@autosee: added redefinition 28
\@gls@noidx@getgrouptitle: fixed bug 131
\@glsxtr@addunusedxrefs: added check for seealso field 56
\@glsxtr@checkgroup: use \csuse instead of \csname 151
\@glsxtr@dorecordnodefer: new 11
\@glsxtr@record@only@setup: added check for \@gls@setupsort@none 15
\@print@unsrt@glossary: corrected misspelt command 148
\@printunsrt@glossary@handler: new 149
\gls@checkseeallowed: added redefinition 28
\glsxtr@writefields: added \providecommand lines 143
\glsxtrautoindex: new 190
\glsxtrautoindexassignsort: new 191
\glsxtrautoindexentry: new 190
\glsxtrindexseealso: new 52
\glsxtrseealsolabels: new 55
\glsxtrseelist: new 52
\glsxtruseseealso: new 51
\glsxtruseseealsoformat: new 52
\seealsoname: new 52
autoseeindex: new 17
1.17 (2017-08-09)
\@glsxtr@mark@wordseps: new 203
\@glsxtr@markwordseps: new 203
\@glsxtr@noidx@displaynumberlist: replace hard-coded ?? with \glsxtrundeftag 130
\@glsxtr@noidx@entrynumberlist: replace hard-coded ?? with \glsxtrundeftag 131
\@glsxtr@noidx@numberlistloop: replace hard-coded ?? with \glsxtrundeftag 131
\@glsxtrifhyphenstart: new 302
General: removed some inconsistencies in the abbreviation styles 225
\glsabbrvhyphenfont: new 302
\glsabbrvonlyfont: new 315
\glsabbrvscfont: new 242
\glsabbrvsmfont: new 256

  - \glsabbrvuserfont: initialised to default font 292
  - \glsfirstabbrvhyphenfont: new 302
  - \glsfirstabbrvonlyfont: new 315
  - \glsfirstabbrvscfont: new 242
  - \glsfirstabbrvsmfont: new 256
  - \glsfirstlonghyphenfont: new 302
  - \glsfirstlongonlyfont: new 316
  - \glslonghyphenfont: new 302
  - \glslongonlyfont: new 315
  - \glslonguserfont: initialised to default font 293
  - \glsxtr@newabbreviation: added \glsxtrorgshort and \glsxtrorglong 204
  - \GlsXtrDefineAcShortcuts: new 21
  - \glsxtrgenabbrvfmt: added check for \ifglsxtrinsertinside 219
  - \glsxtrhyphensuffix: new 302
  - \glsxtrifhyphenstart: new 301
  - \glsxtrlonghyphen: new 307
  - \glsxtrlonghyphennoshort: new 304
  - \glsxtrlonghyphenshort: new 302
  - \glsxtrlongshortdescname: new 227
  - \glsxtronlydescname: new 317
  - \glsxtronlydescsort: new 317
  - \glsxtronlysuffix: new 316
  - \glsxtrparen: new 206
  - \glsxtrposthyphenlong: new 312
  - \glsxtrposthyphenshort: new 307
  - \glsxtrposthyphensubsequent: new 307
  - \glsxtrshortdescname: new 236
  - \glsxtrshorthyphen: new 312
  - \glsxtrshorthyphenlong: new 310
  - \glsxtrshortlongdescname: new 229
  - \glsxtrshortlongdescsort: new 229
  - \Glsxtrsubsequentfmt: new 222
  - \glsxtrsubsequentfmt: new 221
  - \Glsxtrsubsequentplfmt: new 222
  - \glsxtrsubsequentplfmt: new 222
  - \glsxtrword: new 203
  - \glsxtrwordsep: new 203
  - short-hyphen-long-hyphen: new 311
  - short-hyphen-long-hyphen-desc: new 311
  - short-hyphen-postlong-hyphen: new 313
  - short-hyphen-postlong-hyphen-desc: new 315
  - short-long-user-desc: corrected first forms 301
  - short-nolong-desc-noreg: new 237
  - short-nolong-noreg: new 235
  - long-em-noshort-em-desc-noreg: new 288
  - long-em-noshort-em-noreg: new 284
  - long-hyphen-noshort-desc-noreg: new 304
  - long-hyphen-postshort-hyphen: new 308
  - long-hyphen-postshort-hyphen-desc: new 309
  - long-hyphen-short-hyphen: new 303
  - long-hyphen-short-hyphen-desc: new 303
  - long-noshort-desc-noreg: new 240
  - long-noshort-noreg: new 241
  - long-only-short-only: new 316
  - long-only-short-only-desc: new 317
  - long-short-user-desc: corrected first forms 299
- 1.18 (2017-08-10)
  - stylemods: changed default value to “default” 24
- 1.19 (2017-09-09)
  - \@glsxtr@defaultnumberformat: new 7
  - \@glsxtr@dorecord: Use \@glsrecordlocref instead of \@glslocref 10
  - \@glsxtr@dorecordnodefer: Use \theglsentrycounter for the location rather than \@glslocref 11
  - \@glsxtr@record@setting: new 14
  - \@glsxtr@record@setting@alsoindex: new 14
  - \@glsxtrifhasfield: new 35
  - General: added \glslink option theHvalue 70
    - added \glslink option thevalue 70
  - \glsxtr@writefields: removed double-quotes around \jobname 144
  - \glsxtrdoautoindexname: changed format test 190
  - \glsxtrhyperlink: new 95
  - \glsxtrifhasfield: new 35
  - \GlsXtrSetDefaultNumberFormat: new 7
  - \s@glsxtrifhasfield: new 36

1.20 (2017-09-11)
- `\@glsxtrhypernameprefix`: new 129
- `\glsdohypertarget`: added redefinition 129
- `\printunsrtglossaryunitsetup`: switched from redefining `\glolinkprefix` to `\@glsxtrhypernameprefix` 150

1.21 (2017-11-03)
- `\@@glsxtr@record`: added check for default options 9
- `\@@glsxtrwrglossmark`: new 26
- `\@glslink`: changed `\let` to `\def` 96
- `\@glsxtr@checkgroup`: new 150
- `\@glsxtr@defpostpunc`: new 18
- `\@glsxtr@do@record@wrglossary`: new 8
- `\@glsxtr@dosee@alsoindex@glossary`: new 28
- `\@glsxtr@doseeglossary`: new 27
- `\@glsxtr@noidx@do`: removed code dealing with the group 152
- `\@glsxtr@record@setting@off`: new 15
- `\@glsxtr@record@setting@only`: new 14
- `\@glsxtr@rglstrigger@record`: new 156
- `\@glsxtrglossentry`: new 145
- `\@glsxtrnewgls`: new 153
- `\@glsxtrsetaliasnoindex`: changed to use `\glsxtrifhasfield` instead of `\ifglshasfield` 90
- `\@glsxtrwrglossmark`: new 26
- `\@rGLS`: new 158
- `\@rGLS@`: new 158
- `\@rGLSpl`: new 159
- `\@rGLSpl@`: new 159
- `\@rGls`: new 158
- `\@rGls@`: new 158
- `\@rGlspl`: new 158
- `\@rGlspl@`: new 158
- `\@rgls`: new 157
- `\@rgls@`: new 157
- `\@rglspl`: new 157
- `\@rglspl@`: new 157
- General: adjusted mcolalttree 416
  - modified index to remove hard coded `\space` 396
  - modified list to remove hard coded `\space` 385
  - moved conditional outside of `\glsgroupskip` 388–395
  - new 419
  - redefined `altlistgroup` to discourage breaks after group headings 386
  - redefined `altlisthypergroup` to discourage breaks after group headings 387
  - redefined `alttreegroup` to discourage breaks after group headings 412
  - redefined `alttreehypergroup` to discourage breaks after group headings 413
  - redefined `indexgroup` to discourage breaks after group headings 397
  - redefined `indexhypergroup` to discourage breaks after group headings 397
  - redefined `listgroup` to discourage breaks after group headings 386
  - redefined `listhypergroup` to discourage breaks after group headings 386
  - redefined `mcolalttreegroup` to discourage breaks after group headings 417
  - redefined `mcolalttreehypergroup` to discourage breaks after group headings 417
  - redefined `mcolalttreespannav` to discourage breaks after group headings 418
  - redefined `mcolindexgroup` to discourage breaks after group headings 413
  - redefined `mcolindexhypergroup` to discourage breaks after group headings 413
  - redefined `mcolindexspannav` to discourage breaks after group headings 414
  - redefined `mcoltreegroup` to discourage breaks after group headings 414
  - redefined `mcoltreehypergroup` to discourage breaks after group headings 415
  - redefined `mcoltreenonamegroup` to discourage breaks after group headings 415
  - redefined

  mcoltreenonamehypergroup to discourage breaks after group headings 416
  redefined mcoltreenonamespannav to discourage breaks after group headings 416
  redefined mcoltreespannav to discourage breaks after group headings 415
  redefined treegroup to discourage breaks after group headings 398
  redefined treehypergroup to discourage breaks after group headings 399
  redefined treenonamegroup to discourage breaks after group headings 400
  redefined treenonamehypergroup to discourage breaks after group headings 400
debug: new 26
\gglssetwidest: new 402
\glsdisablehyper: added check for existence 95
  changed to use \def rather than \let 95
\glsenablehyper: changed to use \def rather than \let 95
\Glsfmtname: new 332
\glsfmtname: new 332
\glshex: new 339
\glslistchildpostlocation: new 385
\glslistchildprelocation: new 385
\glslistprelocation: new 385
\glsnavhyperlink: patched 93
\glsseeitemformat: new 49
\glsshowtarget: new 27
\glstreechildprelocation: new 396
\glstreeprelocation: new 396
\glstriggerrecordformat: new 157
\glsuseabbrvfont: new 219
\glsuselongfont: new 219
\glsxtr@do@alsoindex@wrglossary: new 8
\glsxtr@org@@do@wrglossary: new 29
\glsxtr@org@dohyperlink: new 93
\glsxtr@setbookindexmark: new 424
\glsxtrbookindexatendgroup: new 420
\glsxtrbookindexbetween: new 420
\glsxtrbookindexbookmark: new 420
\glsxtrbookindexcols: new 419
\glsxtrbookindexcolspread: new 420
\glsxtrbookindexfirstmark: new 424
\glsxtrbookindexfirstmarkfmt: new 424
\glsxtrbookindexformatheader: new 420
\glsxtrbookindexgroupskip: new 420
\glsxtrbookindexlastmark: new 424
\glsxtrbookindexlastmarkfmt: new 424
\glsxtrbookindexmarkentry: new 424
\glsxtrbookindexname: new 419
\glsxtrbookindexparentchildsep: new 419
\glsxtrbookindexparentsubchildsep: new 419
\glsxtrbookindexprelocation: new 419
\glsxtrbookindexsubatendgroup: new 420
\glsxtrbookindexsubbetween: new 420
\glsxtrbookindexsubname: new 419
\glsxtrbookindexsubprelocation: new 419
\glsxtrbookindexsubsubatendgroup: new 420
\glsxtrbookindexsubsubbetween: new 420
\glsxtrbookindexthepage: new 423
\glsxtrdetoklocation: new 155
\glsxtrenablerecordcount: new 155
\glsxtrglossentry: new 145
\glsxtrgroupfield: new 150
\Glsxtrheadname: new 324
\glsxtrheadname: new 323
\GlsXtrIfFieldEqStr: new 38
\glsxtriflabelinlist: new 149
\glsxtrifrecordtrigger: new 156
\glsxtrindexseealso: added check that the entry exists 52
\glsxtrinithyperoutside: new 70
\GlsXtrLocationRecordCount: new 155
\glsxtrnewgls: new 152, 154
\glsxtrnewGLSlike: new 154
\glsxtrnewglslike: new 154
\glsxtrnewrgls: new 154
\glsxtrnewrGLSlike: new 154
\glsxtrnewrglslike: new 154
\glsxtrprelocation: new 384, 419
\GlsXtrRecordCount: new 155
\glsxtrrecordtriggervalue: new 155

\glsxtrresourcefile: now disables record key 141
\glsxtrresourceinit: new 142
\GlsXtrSetRecordCountAttribute: new 155
\glsxtrtitlename: new 324
\glsxtrtitleorpdforheading: new 320
\GlsXtrTotalRecordCount: new 154
\glsxtrwrglossmark: new 26
short-em: new 278
short-sc: corrected first letter uppercasing 247
short-sm: corrected first letter uppercasing 261
shortcuts: ac 23
\ifglsxtr@hyperoutside: new 70
all: new 383
nolong-short: new 238
nolong-short-em: new 280
nolong-short-noreg: new 238
nolong-short-sc: new 249
nolong-short-sm: new 263
nopostdot: new 19
postpunc: new 19
\printunsrtglossaryentryprocesshook: new 148, 149
\printunsrtglossarypredoglossary: new 149
\printunsrtglossaryskipentry: new 148
\rGLS: new 158
\rGls: new 157
\rgls: new 157
\rGLSformat: new 160
\rGlsformat: new 159
\rglsformat: new 159
\rGLSpl: new 159
\rGlspl: new 158
\rglspl: new 157
\rGLSplformat: new 160
\rGlsplformat: new 159
\rglsplformat: new 159
\s@glsxtrifhasfield: switched from \ifdef to \ifundef 36

1.22 (2017-11-08)
\@glsxtr@nopostpunc: new 128
\@glsxtr@orgprintglossary: changed explicit \let for \nopostdesc to \glsxtractivatenopost 127
\@glsxtrglossentryother: new 146
\glossentrynameother: new 188
\glsseeitemformat: switched check from regular to short 49
\glsxtr@setaccessdisplay: new 187
\glsxtr@writefields: provide \glsxtr@record in aux file 143
\glsxtractivatenopost: new 127
\glsxtrbookindexprelocation: removed check for no post dot 419
\glsxtrglossentryother: new 146
\glsxtrnopostpunc: new 127

1.23 (2017-11-12)
\@@glsxtrfmt: added check for indexing 33
added grouping 33
new 33
\@glsxtr@nopostpunc@postdesc: new 128
\@glsxtr@restore@postpunc: new 128
\@glsxtrentryfmt: fixed missing label argument 34
\@glsxtrfmt: new 33
\eglsupdatewidest: new 403
\gglsupdatewidest: new 402
\glsupdatewidest: new 402
\GlsXtrDefineAbbreviationShortcuts: changed \newabbr definition to use \providecommand 21
\GlsXtrDefineAcShortcuts: changed \newabbr definition to use \providecommand 21
\glsxtrfmtdisplay: new 33
\glsxtrifcustomdiscardperiod: new 198
\GlsXtrIfFieldUndef: new 36
\glsxtrrestorepostpunc: new 128
\s@@glsxtrfmt: new 33
\s@glsxtrfmt: new 33
\xglsupdatewidest: new 403

1.24 (2017-11-14)
\glsadd: added \@gls@setsort 75
\glsxtrforcsvfield: new 35
\glsxtrlocalsetgrouptitle: new 132

1.25 (2017-11-14)
\glsxtrbookindexmulticolsenv: new 420

1.25 (2017-11-24)
\glsextrapostnamehook: new 187
\glsxtrfootnotename: new 230
\glsxtrlongnoshortdescname: new 239
\glsxtrlongnoshortname: new 241
\glsxtrlongshortname: new 225
\glsxtrlongshortuserdescname: new 296

- \glsxtronlyname: new 316
- \glsxtrpostlinkAddDescOnFirstUse: changed to use \glsxtrparen 198
- \glsxtrpostlinkAddSymbolOnFirstUse: changed to use \glsxtrparen 199
- \glsxtrshortlongname: new 228
- \glsxtrshortlonguserdescname: new 298
- \glsxtrshortnolongname: new 234

1.26 (2018-01-05)
- \@glsxtr@do@inc@linkcount: new 160
- \glslinkpresetkeys: new 71
- \glsxtr@inc@linkcount: new 71
- \GlsXtrEnableLinkCounting: new 161
- \GlsXtrIfLinkCounterDef: new 161
- \glsxtrinclinkcounter: new 161
- \GlsXtrLinkCounterName: new 161
- \GlsXtrLinkCounterValue: new 161
- \GlsXtrTheLinkCounter: new 161

1.27 (2018-02-26)
- \@glsxtrdialecthook: new 30
- General: added glossaries-extra-bib2gls.sty 338
- \Alpha: new 350
- \Beta: new 350
- \Chi: new 351
- \Digamma: new 351
- \Epsilon: new 350
- \Eta: new 350
- \glsxtr@loaddialect: new 338
- \glsxtrBasicDigitrules: new 380
- \glsxtrcombiningdiacriticIIIrules: new 354
- \glsxtrcombiningdiacriticIIrules: new 354
- \glsxtrcombiningdiacriticIrules: new 354
- \glsxtrcombiningdiacriticIVrules: new 355
- \glsxtrcombiningdiacriticrules: new 353
- \glsxtrcontrolrules: new 352
- \glsxtrcurrencyrules: new 357
- \glsxtrdigitrules: new 380
- \glsxtrfractionrules: new 381
- \glsxtrGeneralLatinIIIrules: new 359
- \glsxtrGeneralLatinIIrules: new 358
- \glsxtrGeneralLatinIrules: new 358
- \glsxtrGeneralLatinIVrules: new 360
- \glsxtrGeneralLatinVIIIrules: new 363
- \glsxtrGeneralLatinVIIrules: new 362
- \glsxtrGeneralLatinVIrules: new 361
- \glsxtrGeneralLatinVrules: new 361
- \glsxtrgeneralpuncIIrules: new 357
- \glsxtrgeneralpuncIrules: new 356
- \glsxtrgeneralpuncrules: new 356
- \glsxtrhyphenrules: new 356
- \glsxtrLatinA: new 363
- \glsxtrLatinAA: new 366
- \glsxtrLatinAELigature: new 365
- \glsxtrLatinE: new 363
- \glsxtrLatinEszettSs: new 365
- \glsxtrLatinEszettSz: new 365
- \glsxtrLatinEth: new 365
- \glsxtrLatinH: new 364
- \glsxtrLatinI: new 364
- \glsxtrLatinInsularG: new 366
- \glsxtrLatinK: new 364
- \glsxtrLatinL: new 364
- \glsxtrLatinLslash: new 366
- \glsxtrLatinM: new 364
- \glsxtrLatinN: new 364
- \glsxtrLatinO: new 364
- \glsxtrLatinOELigature: new 365
- \glsxtrLatinOslash: new 366
- \glsxtrLatinP: new 364
- \glsxtrLatinS: new 364
- \glsxtrLatinSchwa: new 365
- \glsxtrLatinT: new 365
- \glsxtrLatinThorn: new 365
- \glsxtrLatinWynn: new 366
- \glsxtrLatinX: new 365
- \glsxtrMathGreekIIrules: new 372
- \glsxtrMathGreekIrules: new 371
- \glsxtrMathItalicAlpha: new 377
- \glsxtrMathItalicBeta: new 377
- \glsxtrMathItalicChi: new 379
- \glsxtrMathItalicDelta: new 377
- \glsxtrMathItalicEpsilon: new 377
- \glsxtrMathItalicEta: new 377
- \glsxtrMathItalicGamma: new 377
- \glsxtrMathItalicGreekIIrules: new 368
- \glsxtrMathItalicGreekIrules: new 367
- \glsxtrMathItalicIota: new 378
- \glsxtrMathItalicKappa: new 378
- \glsxtrMathItalicLambda: new 378
- \glsxtrMathItalicLowerGreekIIrules: new 370

\glsxtrMathItalicLowerGreekIrules: new 370
\glsxtrMathItalicMu: new 378
\glsxtrMathItalicNabla: new 380
\glsxtrMathItalicNu: new 378
\glsxtrMathItalicOmega: new 380
\glsxtrMathItalicOmicron: new . . . 378
\glsxtrMathItalicPartial: new . . . 380
\glsxtrMathItalicPhi: new 379
\glsxtrMathItalicPi: new 378
\glsxtrMathItalicPsi: new 379
\glsxtrMathItalicRho: new 379
\glsxtrMathItalicSigma: new 379
\glsxtrMathItalicTau: new 379
\glsxtrMathItalicTheta: new 377
\glsxtrMathItalicUpperGreekIIrules: new 369
\glsxtrMathItalicUpperGreekIrules: new 368
\glsxtrMathItalicUpsilon: new . . . 379
\glsxtrMathItalicXi: new 378
\glsxtrMathItalicZeta: new 377
\glsxtrMathUpGreekIIrules: new . . 367
\glsxtrMathUpGreekIrules: new . . . 366
\glsxtrnonprintablerules: new . . . 353
\glsxtrprovidecommand: new 339
\glsxtrspacerules: new 353
\glsxtrSubScriptDigitrules: new . 380
\glsxtrSuperScriptDigitrules: new 381
\glsxtrUpAlpha: new 373
\glsxtrUpBeta: new 373
\glsxtrUpChi: new 376
\glsxtrUpDelta: new 374
\glsxtrUpDigamma: new 374
\glsxtrUpEpsilon: new 374
\glsxtrUpEta: new 374
\glsxtrUpGamma: new 373
\glsxtrUpIota: new 374
\glsxtrUpKappa: new 375
\glsxtrUpLambda: new 375
\glsxtrUpMu: new 375
\glsxtrUpNu: new 375
\glsxtrUpOmega: new 376
\glsxtrUpOmicron: new 375
\glsxtrUpPhi: new 376
\glsxtrUpPi: new 375
\glsxtrUpPsi: new 376
\glsxtrUpRho: new 375
\glsxtrUpSigma: new 376
\glsxtrUpTau: new 376
\glsxtrUpTheta: new 374
\glsxtrUpUpsilon: new 376
\glsxtrUpXi: new 375
\glsxtrUpZeta: new 374
\Iota: new 350
\Kappa: new 350
\Mu: new 350
\Nu: new 350
\Omicron: new 350
\omicron: new 351
\Rho: new 350
\Tau: new 351
\Upalpha: new 351
\Upbeta: new 351
\Upchi: new 352
\Upepsilon: new 351
\Upeta: new 351
\Upiota: new 351
\Upkappa: new 351
\Upmu: new 351
\Upnu: new 351
\Upomicron: new 351
\upomicron: new 352
\Uprho: new 352
\Uptau: new 352
\Upzeta: new 351
\Zeta: new 350

1.28 (2018-03-06)
  \@glsxtr@docdefval: changed from count register to macro 16
  \@glsxtrdialecthook: save and restore \TrackLangRequireDialectPrefix 381
  \glsxtredeffield: changed \csedef to \protected@csedef 37
  \glsxtrlocalsetgrouptitle: changed \csedef \protected@csedef 132
  \glsxtrsetgrouptitle: changed \csxdef \protected@csxdef 132

1.29 (2018-04-09)
  \@gls@removespaces: added expansion 135
  \@glsxtr@dorecord: don’t suppress expansion of \@glsrecordlocref if counter isn’t page 11
  \@glsxtr@wrglossary@locationhyperlink: new 25
  \glsxtr@inc@wrglossaryctr: new . . . 25
  \glsxtr@wrglossarylocation: new . 339

\GlsXtrBibTeXEntryAliases: new 340
\glsxtrfieldforlistloop: corrected argument order in \forlistcsloop 35
\GlsXtrIndexCounterLink: new 339
\GlsXtrInternalLocationHyperlink: new 25
\GlsXtrProvideBibTeXFields: new 341
indexcounter: new 25
\setentrycounter: new 135
1.30 (2018-04-25)
\@@glsxtr@record: added check for post-key hook 9
added check for pre-key hook 9
\@GLSxtr@fullpl: added \@glsxtr@record 211
\@GlsXtrStopUnsetBuffering: new 106
\@Glsxtr@fullpl: added \@glsxtr@record 211
\@glsxtr@dorecord: don't suppress expansion of \@glsrecordlocref 11
\@glsxtr@full: added \@glsxtr@record 209
\@glsxtr@fullpl: added \@glsxtr@record 210
\@glsxtr@glossadd@postkeys: new 10
\@glsxtr@glossadd@prekeys: new 10
\@glsxtr@glslink@postkeys: new 10
\@glsxtr@glslink@prekeys: new 10
\@glsxtr@local@textformat: new 70
\@glsxtr@unset: new 105
\@glsxtrbuffer@unset: new 105
\glsadd: added \glsaddpostsetkeys 75
added \glsaddpresetkeys 74
\glsaddpostsetkeys: new 74
\glsaddpresetkeys: new 74
\glsuserdescription: new 293
\glsxtrabbreviationfont: new 67
\GlsXtrDualBackLink: new 340
\GlsXtrDualField: new 340
\GlsXtrExpandedFmt: new 71
\GLSxtrlong: added \@glsxtr@record 215
\Glsxtrlong: added \@glsxtr@record 214
\glsxtrlong: added \@glsxtr@record 214
\GLSxtrlongpl: added \@glsxtr@record 219
\Glsxtrlongpl: added \@glsxtr@record 218
\glsxtrlongpl: added \@glsxtr@record 218
\GLSxtrshort: added \@glsxtr@record 213
\Glsxtrshort: added \@glsxtr@record 213
\glsxtrshort: added \@glsxtr@record 212
\GLSxtrshortpl: added \@glsxtr@record 217
\Glsxtrshortpl: added \@glsxtr@record 216
\glsxtrshortpl: added \@glsxtr@record 216
\GlsXtrStartUnsetBuffering: new 105
\GlsXtrStopUnsetBuffering: new 106
indexcounter: added check for wrglossary counter 26
\s@GlsXtrStopUnsetBuffering: new 106
1.31 (2018-05-09)
\@GlsXtrStartUnsetBuffering: new 105
\@gls@ifaccessattribute@set: new 170
\@gls@initaccesskeys: new 169, 175
\@gls@setup@default@short@access: new 170, 175
\@glsxtr@record@noglossarywarning: new 140
\@glsxtrbuffer@nodup@unset: new 105
General: added prefix key for glslink 70
added prefix key for printgloss 129
changed \let to \def 129
\glsaddeach: new 75
\glscapturedgroup: new 339
\glsdefpostdesc: new 197
\glsdefpostlink: new 198
\glsdefpostname: new 187
\glsdohypertarget: bug fix: ensure that new version is picked up 129
\glslistdesc: new 385
\glslocalreseteach: new 107
\glslocalunseteach: new 107
\glstreechilddesc: new 398
\glstreechildsymbol: new 398
\glstreedefaultnamefmt: new 395
\glstreedesc: new 397
\glstreegroupheaderfmt: added redefinition 396
\glstreenamefmt: added redefinition 395
\glstreenavigationfmt: added redefinition 396
\glstreenonamechilddesc: new 399

  \glstreenonamedesc: new 399
  \glstreenonamesymbol: new 399
  \glstreesymbol: new 398
  \glsxtr@newabbreviation: added \ExtraCustomAbbreviationFields 204
  \GlsXtrForUnsetBufferedList: new 106
  \GlsXtrIfFieldCmpNum: new 36
  \GlsXtrIfFieldEqNum: new 36
  \GlsXtrIfFieldEqXpStr: new 39
  \GlsXtrIfFieldNonZero: new 36
  \GlsXtrIfHasNonZeroChildCount: new 339
  \GlsXtrIfXpFieldEqXpStr: new 39
  \glsxtrpostlinkAddSymbolDescOnFirstUse: new 199
  \GlsXtrRecordWarning: new 139
  \glsxtrRevertTocMarks: new 319
  \GlsXtrStandaloneGlossaryType: new 145
  \GlsXtrStandaloneSubEntryItem: new 146
  \s@GlsXtrStartUnsetBuffering: new 105
1.32 (2018-05-24)
  \GlsXtrForeignText: new 39
  \GlsXtrForeignTextField: new 41
  \GlsXtrUnknownDialectWarning: new 41
1.33 (2018-07-26)
  \ifglsused: added redefinition 43
1.34 (2018-07-29)
  \gls@begindocdefs: atom 57
  \GlsXtrIfUnusedOrUndefined: new . . 30
  \glsxtrNoGlossaryWarning: added package warning 23
  \if@glsxtrdocdefrestricted: changed to allow for atom as well . . . 17
  docdef: atom 17
1.35 (2018-08-13)
  \@gls@@link@: initialise post-link hook commands 69
1.36 (2018-08-18)
  \glsxtrautoindexesc: new 190
  \glsxtrdisplaysupploc: new 341
  \glsxtrmultisupplocation: new . . . 341
1.37 (2018-11-30)
  \@@glsxtr@record: added check for auto-add 9
  \@dGLS: new 349
  \@dGLSpl: new 349
  \@dGls: new 349
  \@dGlspl: new 349
  \@dgls: new 348
  \@dglspl: new 349
  \@gls@getcounterprefix: new 29
  \@glslongextrawidestname: new . . . 427
  \@glsxtr@bibgls@removespaces: new 343
  \@glsxtr@check@bibgls@nameref: new 141
  \@glsxtr@do@nameref@record: new . . 12
  \@glsxtr@get@prefixedlabel: new . 348
  \@glsxtr@if@record@only: new 14
  \@glsxtr@ifnum@mmode: new 12
  \@glsxtr@labelprefixes: new 347
  \@glsxtr@prefixlabellist: new . . . 348
  \@glsxtr@providenewgls: new 152
  \@glsxtr@record@only@setup: new . . 15
  \@glsxtr@record@setting@nameref: new 14
  \@glsxtr@use@equation@counter@or: new 71
  General: new 425
  page: nameref 11
  \dGLS: new 349
  \dgls: new 348
  \dglsdisp: new 350
  \dglslink: new 349
  \dGLSpl: new 349
  \glsadd: added grouping 74
    ensure that \glsadd performs indexing 75
  \glslongextraDescAlign: new 426
  \glslongextraDescFmt: new 425
  \glslongextraDescNameHeader: new 432
  \glslongextraDescNameTabularFooter: new 432
  \glslongextraDescNameTabularHeader: new 432
  \glslongextraDescSymNameHeader: new 444
  \glslongextraDescSymNameTabularFooter: new 444
  \glslongextraDescSymNameTabularHeader: new 444
  \glslongextraGroupHeading: new . . 427
  \glslongextraHeaderFormat: new . . 427
  \glslongextraLocationAlign: new . 427
  \glslongextraLocationDescNameHeader: new 433

\glslongextraLocationDescNameTabularFooter: new 433
\glslongextraLocationDescNameTabularHeader: new 433
\glslongextraLocationDescSymNameHeader: new 445
\glslongextraLocationDescSymNameTabularFooter: new 446
\glslongextraLocationDescSymNameTabularHeader: new 445
\glslongextraLocationFmt: new 425
\glslongextraLocationSymDescNameHeader: new 442
\glslongextraLocationSymDescNameTabularFooter: new 442
\glslongextraLocationSymDescNameTabularHeader: new 442
\glslongextraLocSetDescWidth: new 428
\glslongextraNameAlign: new 426
\glslongextraNameDescHeader: new 427
\glslongextraNameDescLocationHeader: new 430
\glslongextraNameDescLocationTabularFooter: new 430
\glslongextraNameDescLocationTabularHeader: new 430
\glslongextraNameDescSymHeader: new 434
\glslongextraNameDescSymLocationHeader: new 436
\glslongextraNameDescSymLocationTabularFooter: new 436
\glslongextraNameDescSymLocationTabularHeader: new 436
\glslongextraNameDescSymTabularFooter: new 435
\glslongextraNameDescSymTabularHeader: new 434
\glslongextraNameDescTabularFooter: new 427
\glslongextraNameDescTabularHeader: new 427
\glslongextraNameFmt: new 425
\glslongextraNameSymDescHeader: new 437
\glslongextraNameSymDescLocationHeader: new 439
\glslongextraNameSymDescLocationTabularFooter: new 439
\glslongextraNameSymDescLocationTabularHeader: new 439
\glslongextraNameSymDescTabularFooter: new 438
\glslongextraNameSymDescTabularHeader: new 438
\glslongextraSetDescWidth: new 428
\glslongextraSetWidest: new 427
\glslongextraSubDescFmt: new 426
\glslongextraSubLocationFmt: new 426
\glslongextraSubNameFmt: new 426
\glslongextraSubSymbolFmt: new 426
\glslongextraSymbolAlign: new 426
\glslongextraSymbolFmt: new 425
\glslongextraSymDescNameHeader: new 441
\glslongextraSymDescNameTabularFooter: new 441
\glslongextraSymDescNameTabularHeader: new 441
\glslongextraSymLocSetDescWidth: new 429
\glslongextraSymSetDescWidth: new 428
\glslongextraTabularVAlign: new 429
\glslongextraUpdateWidest: new 427
\glslongextraUpdateWidestChild: new 428
\glsrenewcommand: new 339
\glsseeitemformat: removed reference to \glslabel 49
\glsxtr@dblfloat: new 18
\glsxtr@do@autoadd: new 71
\glsxtr@float: new 18
\glsxtr@record@nameref: new 144
\glsxtr@renewcommand: new 339
\glsxtr@writefields: provide \glsxtr@record@nameref in aux file 143
\glsxtraddlabelprefix: new 347
\GlsXtrAutoAddOnFormat: new 71
\glsxtrclearlabelprefixes: new 347
\glsxtrdisplaylocnameref: new 341
\glsxtrfmtexternalnameref: new 344
\glsxtrfmtinternalnameref: new 344
\GLSXTRhiername: new 51
\GLSxtrhiername: new 51
\GlsXtrhiername: new 50
\Glsxtrhiername: new 50
\glsxtrhiername: new 50

\glsxtrhiernamesep: new 51
\glsxtridentifyglslike: new 152
\glsxtrifinlabelprefixlist: new . 348
\GlsXtrLocationField: new 151
\glsxtrnameloclink: new 343
\glsxtrnamereflink: new 342
\glsxtrprependlabelprefix: new .. 347
\GlsXtrSetAltModifier: write modifier to aux 93
\glsxtrSetWidest: new 344
\glsxtrSetWidestFallback: new ... 346
\GlsXtrStandaloneEntryName: new . 145
\GlsXtrStandaloneEntryOther: new 147
\GLSxtrusefield: new 37
\Glsxtrusefield: fixed internal command and added check for \texorpdfstring 37
\ifGlsLongExtraUseTabular: new .. 429
floats: new 18
long-desc-name: new 432
long-desc-sym-name: new 444
long-loc-desc-name: new 433
long-loc-desc-sym-name: new 446
long-loc-sym-desc-name: new 443
long-name-desc: new 429
long-name-desc-loc: new 430
long-name-desc-sym: new 435
long-name-desc-sym-loc: new 436
long-name-sym-desc: new 438
long-name-sym-desc-loc: new 439
long-sym-desc-name: new 441
equations: new 18

1.38 (2018-12-01)
\glslongextraNameFmt: bug fix: removed double param 425

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

\$ 339
\, 51
\. 18, 19, 198, 395
\@ 57, 141
\@@cGLS@ 109, 117
\@@cGLSpl@ 109, 117
\@@cGls@ 109, 117
\@@cGlspl@ 109, 117
\@@cgls@ 109, 117
\@@cglspl@ 109, 117
\@@do@@wrglossary 8, 10, 124
\@@do@wrglossary 13, 15, 16, 29, 75, 91
\@@glo@assign@sortkey 129
\@@glo@list 6
\@@glo@type 147
\@@glossarysec 420
\@@gls@expand@field 32
\@@glslocalreset 107
\@@glslocalunset 106
\@@glsreset 106
\@@glsunset 105
\@@glsxtr@autoindex@escspch 192, 193
\@@glsxtr@checkspch 191–194
\@@glsxtr@disabledflycommand 62
\@@glsxtr@org@postdescription 128
\@@glsxtr@record 15, 16
\@@glsxtr@recordcounter 15, 16, 144
\@@glsxtrfmt 33
\@@glsxtrp 100, 101
\@@glsxtrpostloctag 65
\@@glsxtrpreloctag 65
\@@glsxtrwrglossmark 8, 9, 13, 28, 29, 53, 58, 124
\@@newglossaryentry@defcounters 108
\@@newglossaryentry@defunitcounters 116
\@@par 401
\@ACRlong 96
\@ACRlongpl 96
\@ACRshort 96
\@ACRshortpl 96
\@Acrlong 96
\@Acrlongpl 96
\@Acrshort 96
\@Acrshortpl 96
\@GLS@ 96, 112, 159, 349
\@GLSdesc@ 80
\@GLSpl@ 96, 112, 113, 159, 349
\@GLSplural@ 97
\@GLSsymbol@ 81
\@GLStext@ 97
\@GLSxtr@full 210
\@GLSxtr@fullpl 211
\@GLSxtr@p@acrlong@ 96
\@GLSxtr@p@acrlongpl@ 96
\@GLSxtr@p@acrshort@ 96
\@GLSxtr@p@acrshortpl@ 96
\@GLSxtr@p@long@ 96
\@GLSxtr@p@longpl@ 96
\@GLSxtr@p@plural@ 96
\@GLSxtr@p@short@ 96
\@GLSxtr@p@shortpl@ 96
\@GLSxtr@p@text@ 96
\@GLSxtrlong 96, 215
\@GLSxtrlongpl 96, 219
\@GLSxtrp 104
\@GLSxtrshort 96, 213
\@GLSxtrshortpl 96, 217
\@Gls@ 96, 111, 112, 158, 349
\@Gls@acrentryname 120
\@Gls@entry@field 37, 87, 102, 103, 188
\@Gls@entryname 120
\@GlsXtrEnableOnTheFly 59
\@GlsXtrStartUnsetBuffering 105

\@GlsXtrStopUnsetBuffering 106
\@Glspl@ 96, 111, 112, 158, 349
\@Glsplural@ 97
\@Glstext@ 97
\@Glsxtr 60, 62
\@Glsxtr@full 209
\@Glsxtr@fullpl 211
\@Glsxtr@p@acrlong@ 96
\@Glsxtr@p@acrlongpl@ 96
\@Glsxtr@p@acrshort@ 96
\@Glsxtr@p@acrshortpl@ 96
\@Glsxtr@p@long@ 96
\@Glsxtr@p@longpl@ 96
\@Glsxtr@p@plural@ 96
\@Glsxtr@p@short@ 96
\@Glsxtr@p@shortpl@ 96
\@Glsxtr@p@text@ 96
\@Glsxtrlong 96, 214
\@Glsxtrlongpl 96, 218
\@Glsxtrp 103
\@Glsxtrpl 61, 62
\@Glsxtrshort 96, 212
\@Glsxtrshortpl 96, 216
\@acrlong 96
\@acrlongpl 96
\@acrshort 96
\@acrshortpl 96
\@addtoreset 160
\@afterheading 386, 387, 397, 399–401, 413–416, 423
\@alt@gls@hyp@opt 93
\@auxout 11–13, 58, 66, 93, 109, 110, 118, 123, 124, 136, 137, 141, 143, 145, 152, 153, 348, 424
\@bibgls@restoreat 141
\@cGLS 112
\@cGLS@ 109, 112, 117
\@cGLSpl 113
\@cGLSpl@ 109, 113, 117
\@cGls@ 109, 117
\@cGlspl@ 109, 117
\@cgls@ 109, 117
\@cglspl@ 109, 117
\@dGLS 349
\@dGLSpl 349
\@dGls 349
\@dGlspl 349
\@dblfloat 18
\@dgls 348
\@dglspl 348
\@disable@onlypremakeg 123
\@do@auxoutstuff 136, 137
\@do@gls@getcounterprefix 11
\@do@glssee 54, 55
\@do@newglossaryentry 120, 206
\@do@seeglossary 15, 16, 27, 58, 124
\@do@wrglossary 73, 74, 156
\@empty 30, 75, 83–87, 128, 135, 191, 192, 209–219
\@end@glsxtr@addunused 56
\@end@glsxtr@gettype 126, 129
\@end@glsxtr@usesee 49
\@end@glsxtrifhyphenstart 301, 302
\@endfortrue 35, 187, 222, 348
\@firstofone 76, 148, 182, 189, 195, 342, 343
\@firstofthree 69, 75, 83–86, 92, 93, 209, 210, 212, 214, 216, 218
\@firstoftwo 76–78, 80, 81, 84–87, 89, 92, 93, 122, 187, 188, 200, 201, 209–211, 216–219, 320, 321
\@float 18
\@for 6, 24, 35, 56, 75, 107, 108, 119, 123, 126, 133, 148, 155, 161, 180, 187, 195, 348
\@glo@alias 53, 54
\@glo@assign@sortkey 126
\@glo@autosee 28
\@glo@autoseehook 54
\@glo@category 114
\@glo@check@sortallowed 126
\@glo@counterprefix 10, 11, 29, 30, 135, 343
\@glo@countunit 114
\@glo@default@sorttype 126
\@glo@desc 43, 44
\@glo@descplural 43, 44
\@glo@group 14
\@glo@label 13, 14, 32, 49, 53–55, 87, 95, 404–410
\@glo@location 14
\@glo@loclist 13
\@glo@name 190
\@glo@no@assign@sortkey 129
\@glo@parent 405, 406
\@glo@see 49, 52, 54–56
\@glo@seealso 53, 54
\@glo@sort 190
\@glo@sorttype 126, 133
\@glo@text 69
\@glo@thislettergrp 151

\@glo@thisvalue 292
\@glo@tmp 30–32, 52, 87
\@glo@type 55, 93, 120, 123, 126, 127, 129, 132, 133, 136, 137, 139, 140, 147, 148
\@glo@types 178, 179, 403–410
\@glossary@default@style 62, 63, 126, 418
\@glossarystyle 126, 127
\@gls@ 96, 111, 112, 157, 348
\@gls@@link 69
\@gls@ReturnAfterFi 135, 343
\@gls@actualchar 191
\@gls@adjustmode 74
\@gls@alt@hyp@opt 93
\@gls@alt@hyp@opt@char 93
\@gls@alt@hyp@opt@keys 93
\@gls@automake 126
\@gls@between 132, 133
\@gls@checkedmkidx 191–194
\@gls@checkmkidxchars 53, 190
\@gls@codepage 137
\@gls@counter 9, 11, 26, 30, 71, 72, 74, 91, 156
\@gls@currentlettergroup 133, 148, 151
\@gls@declareoption 5
\@gls@default@longpl 204, 205
\@gls@deffile 57
\@gls@doautomake 126, 144
\@gls@doautomake@err 144
\@gls@enablesavenonumberlist 57
\@gls@encapchar 191
\@gls@entry@count 109, 110
\@gls@entry@field 32, 37, 54, 87, 101, 103, 104, 108, 146, 147
\@gls@entry@unitcount 117, 118
\@gls@field@font 75–83
\@gls@field@link 76–83, 88, 89
\@gls@firstaccess 169, 172
\@gls@get@counterprefix 30
\@gls@getcounterprefix 11
\@gls@getgrouptitle 132, 148
\@gls@grptitle 93, 133
\@gls@hyp@opt 88, 89, 93, 112, 113, 153, 157–159, 208–219, 348, 349
\@gls@hyp@opt@cs 92, 93
\@gls@ifaccessattribute@set 170
\@gls@ifinlist 149
\@gls@increment@currcount 109
\@gls@increment@currunitcount 117
\@gls@initaccesskeys 204
\@gls@keymap 13, 14, 31, 49, 53, 87, 143, 187
\@gls@label 8, 9, 11, 58, 92, 124, 145, 222
\@gls@levelchar 191
\@gls@link 33, 68, 69, 83–87, 209–219
\@gls@link@checkfirsthyper 69, 122
\@gls@link@label 72, 156
\@gls@link@nocheckfirsthyper 67, 83–87, 209–219
\@gls@link@opts 72
\@gls@list 133
\@gls@local@increment@currcount 109
\@gls@local@increment@currunitcount 117
\@gls@location 151, 152
\@gls@loclist 130, 131, 151, 152
\@gls@long 204
\@gls@longpl 202, 204, 205
\@gls@map 187
\@gls@nameaccess 169, 171
\@gls@nohyperlist 44–46
\@gls@noidx@do 133
\@gls@noidx@getgrouptitle 148
\@gls@noidx@nosanitizesort 125
\@gls@noidx@sanitizesort 125
\@gls@noidxloclist@finalsep 130
\@gls@noidxloclist@prev 130
\@gls@noidxloclist@sep 130
\@gls@noref@warn 124, 133
\@gls@org@glsnoidxdisplayloc 130, 131
\@gls@org@glsseeformat 130, 131
\@gls@preglossaryhook 127, 195
\@gls@prevlevel 411–413, 417, 418
\@gls@quotechar 191
\@gls@reference 58, 123, 124
\@gls@restoreat 57
\@gls@saveentrycounter 15, 16, 29, 73, 75, 156
\@gls@see@noindex 28, 141
\@gls@setdefault@glslink@opts 9, 33, 72, 91
\@gls@setsort 73, 75
\@gls@setup@default@short@access 205
\@gls@setupsort@none 15
\@gls@short 204, 205
\@gls@shortaccess 169–172
\@gls@shortaccesspl 169–171
\@gls@shortpl 202, 205
\@gls@sort 151
\@gls@textaccess 169, 171
\@gls@thisHloc 29
\@gls@thislabel 75, 107, 348–350
\@gls@thisloc 29

\@gls@thisval 187
\@gls@tmp 39, 133
\@gls@tmpb 193, 194
\@gls@type 124, 126, 222, 404–410
\@gls@write@entrycounts 109
\@gls@write@entryunitcounts 117
\@gls@write@entryunitcounts@do 118
\@gls@writedef 57
\@gls@xref 13, 53
\@glsabbrv@current@abbreviation 204, 219
\@glsacronymlists 120
\@glsdoifexistsorwarn 17, 183–186, 188
\@glsentry 57, 110, 118
\@glslink 73, 94, 95
\@glslocalreset 107
\@glslocalunset 106, 107
\@glslongextra@begintab 429–446
\@glslongextrawidestname 427, 428
\@glsnextpages 127
\@glsnonextpages 127
\@glsnumberformat 9, 11, 71, 72, 74, 91, 156, 187, 190, 343
\@glsorder 123
\@glspl@ 96, 111, 112, 157, 349
\@glsplural@ 97
\@glspunc@token 200, 201
\@glsrecordlocref 10, 11
\@glsshowtarget 95
\@glsstyle@altlist 385
\@glsstyle@altlistgroup 386
\@glsstyle@altlisthypergroup 387
\@glsstyle@alttree 401
\@glsstyle@alttreegroup 412
\@glsstyle@alttreehypergroup 413
\@glsstyle@index 396
\@glsstyle@indexgroup 397
\@glsstyle@indexhypergroup 397
\@glsstyle@inline 395
\@glsstyle@list 385
\@glsstyle@listdotted 384
\@glsstyle@listgroup 386
\@glsstyle@listhypergroup 386
\@glsstyle@mcolalttree 416
\@glsstyle@mcolalttreegroup 417
\@glsstyle@mcolalttreehypergroup 417
\@glsstyle@mcolalttreespannav 418
\@glsstyle@mcolindexgroup 413
\@glsstyle@mcolindexhypergroup 413
\@glsstyle@mcolindexspannav 414
\@glsstyle@mcoltreegroup 414
\@glsstyle@mcoltreehypergroup 415
\@glsstyle@mcoltreenonamegroup 415
\@glsstyle@mcoltreenonamehypergroup 416
\@glsstyle@mcoltreenonamespannav 416
\@glsstyle@mcoltreespannav 415
\@glsstyle@tree 397
\@glsstyle@treegroup 398
\@glsstyle@treehypergroup 399
\@glsstyle@treenoname 399
\@glsstyle@treenonamegroup 400
\@glsstyle@treenonamehypergroup 400
\@glstarget 95, 129
\@glstext@ 97
\@glsunset 105, 106
\@glswidestname 403, 411
\@glsxtr 60, 62
\@glsxtr@@do@@wrglossary 124
\@glsxtr@abbreviationsdef 20, 28, 29
\@glsxtr@accessdisplay 187–189
\@glsxtr@activate@initialtagging 195, 196
\@glsxtr@addunitcounter 114
\@glsxtr@addunused 56
\@glsxtr@addunusedxrefs 55, 56
\@glsxtr@altmodifier 93
\@glsxtr@attrval 73, 181–186, 188, 190
\@glsxtr@autoindex@at 190–192
\@glsxtr@autoindex@doextra@esc 190
\@glsxtr@autoindex@encap 190–192
\@glsxtr@autoindex@esc 191, 193, 194
\@glsxtr@autoindex@escat 191, 192
\@glsxtr@autoindex@escencap 191, 192
\@glsxtr@autoindex@esclevel 191, 192
\@glsxtr@autoindex@escquote 191, 193
\@glsxtr@autoindex@level 191, 192
\@glsxtr@autoindex@setname 190
\@glsxtr@autoindexcrossrefs 15, 17, 49, 54
\@glsxtr@autoseeindexfalse 15
\@glsxtr@autoseeindextrue 17
\@glsxtr@bibgls@removespaces 343
\@glsxtr@bookindex@atendgroup 421–423
\@glsxtr@bookindex@atsubendgroup 422
\@glsxtr@bookindex@atsubsubendgroup 422
\@glsxtr@bookindex@between 421, 423
\@glsxtr@bookindex@sep 421, 422
\@glsxtr@bookindex@subatendgroup 421–423
\@glsxtr@bookindex@subbetween 421, 422

\@glsxtr@bookindex@subsep 421, 422
\@glsxtr@bookindex@subsubatendgroup 421–423
\@glsxtr@bookindex@subsubbetween 421, 422
\@glsxtr@bookindexgroupskip . . . 421, 423
\@glsxtr@cat 108, 119, 155, 195
\@glsxtr@check@bibgls@nameref 141
\@glsxtr@checkgroup 148
\@glsxtr@counterrecordhook 11–13
\@glsxtr@csname 115, 117
\@glsxtr@current@style 63, 418
\@glsxtr@currentunitcount 115, 117
\@glsxtr@currunitcount 116, 118
\@glsxtr@debugnr 26
\@glsxtr@debugval 26
\@glsxtr@declareoption . . . 5, 18, 20, 23, 25
\@glsxtr@defaultnoglossarywarning 23, 24
\@glsxtr@defaultnumberformat 7, 9, 72, 74, 91, 187, 190
\@glsxtr@defpostpunc 18, 19, 27
\@glsxtr@deprecated@abbrstyle 251, 253, 254, 256, 265, 267, 268, 270, 283, 286, 290, 292
\@glsxtr@dialect 39–41
\@glsxtr@disabledflycommand 62
\@glsxtr@display@loc 134
\@glsxtr@do@@wrindex 92
\@glsxtr@do@autoadd 71
\@glsxtr@do@glsdisablehyperinlist . . 90
\@glsxtr@do@inc@linkcount 161
\@glsxtr@do@nameref@record 11
\@glsxtr@do@record@wrglossary 8, 15
\@glsxtr@do@redef@forglsentries 7
\@glsxtr@do@style 25, 338
\@glsxtr@do@titlecaps@warn 182–185, 188, 195, 196
\@glsxtr@doabbreviationsdef 20
\@glsxtr@doaccsupp 23, 27
\@glsxtr@docdefsetting 17, 57
\@glsxtr@docdefval 16, 17, 57, 58
\@glsxtr@docounterrecord 13
\@glsxtr@doglossary 148
\@glsxtr@doiflabelinlist 149
\@glsxtr@doloctag 65
\@glsxtr@dorecord 8, 10
\@glsxtr@dorecordnodefer 8, 10
\@glsxtr@dosee@alsoindex@glossary . . 16
\@glsxtr@doseeglossary 15, 28
\@glsxtr@dostylewarn 222
\@glsxtr@enabletagging 194
\@glsxtr@end@ 59
\@glsxtr@endescspch 191–194
\@glsxtr@entrycount@org@localreset 109
\@glsxtr@entrycount@org@localunset 109
\@glsxtr@entrycount@org@reset 109
\@glsxtr@entrycount@org@unset 109
\@glsxtr@entryunitcount@org@localreset 117
\@glsxtr@entryunitcount@org@localunset 117
\@glsxtr@entryunitcount@org@reset . 117
\@glsxtr@entryunitcount@org@unset . 117
\@glsxtr@equationsfalse 18
\@glsxtr@err@undefaction 7, 16
\@glsxtr@field@linkdefs 67, 69
\@glsxtr@floatsfalse 18
\@glsxtr@format@overridefalse 189
\@glsxtr@format@overridetrue 189
\@glsxtr@foundinlist 201
\@glsxtr@full 208
\@glsxtr@fullpl 210
\@glsxtr@get@prefixedlabel 348–350
\@glsxtr@gettype 126
\@glsxtr@glossdescfont 181–183
\@glsxtr@glossnamefont . 183–186, 188, 189
\@glsxtr@gobbleto@endescspch 193
\@glsxtr@groupheading 148, 150, 151
\@glsxtr@idx@displaynumberlist 125
\@glsxtr@idx@entrynumberlist 125
\@glsxtr@if@record@only 122
\@glsxtr@ifcsstart 59
\@glsxtr@ifnum@mmode 71
\@glsxtr@ifpunctoken 201
\@glsxtr@ifunitcounter 114
\@glsxtr@insert@dots 202
\@glsxtr@insert@dots@next 203
\@glsxtr@insertdots 170, 204
\@glsxtr@label 35, 56, 161, 180, 181
\@glsxtr@labelprefixes 347, 348
\@glsxtr@loadstyles 383, 384
\@glsxtr@local@textformat 72, 73
\@glsxtr@locale 40, 41
\@glsxtr@longnewglossaryentry 43
\@glsxtr@mark@wordseps 203
\@glsxtr@mark@wordseps@next 203
\@glsxtr@markwordseps 204, 205
\@glsxtr@mixed@assign@sortkey 126

\@glsxtr@newglslike 152, 153
\@glsxtr@noidx@displaynumberlist 125
\@glsxtr@noidx@do 150
\@glsxtr@noidx@entrynumberlist 125
\@glsxtr@noidx@numberlistloop 125
\@glsxtr@nomissingglstextnr 23, 24
\@glsxtr@nomissingglstextval 23
\@glsxtr@noop@recordcounter 12, 16
\@glsxtr@nopostpunc 127
\@glsxtr@nopostpunc@postdesc 128
\@glsxtr@notfoundinlist 201
\@glsxtr@op@recordcounter 15, 16
\@glsxtr@optlist 62
\@glsxtr@org@@starttoc 319
\@glsxtr@org@GLS@ 68
\@glsxtr@org@GLSpl@ 68
\@glsxtr@org@Gls@ 68
\@glsxtr@org@Glspl@ 68
\@glsxtr@org@Glsxtrtitlefirst 320, 321
\@glsxtr@org@Glsxtrtitlefirstplural 320, 322
\@glsxtr@org@Glsxtrtitlefull 320, 322
\@glsxtr@org@Glsxtrtitlefullpl 320, 322
\@glsxtr@org@Glsxtrtitlelong 320, 322
\@glsxtr@org@Glsxtrtitlelongpl 320, 322
\@glsxtr@org@Glsxtrtitlename 320, 321
\@glsxtr@org@Glsxtrtitleplural 320, 321
\@glsxtr@org@Glsxtrtitleshort 320, 321
\@glsxtr@org@Glsxtrtitleshortpl 320, 321
\@glsxtr@org@Glsxtrtitletext 320, 321
\@glsxtr@org@MakeUppercase 320, 321
\@glsxtr@org@checkfirsthyper 89, 122
\@glsxtr@org@currentfieldvalue 39, 40
\@glsxtr@org@delimN 65, 66
\@glsxtr@org@delimR 65, 66
\@glsxtr@org@doseeglossary 28, 124
\@glsxtr@org@gloautosee 28
\@glsxtr@org@glolinkprefix 72, 74
\@glsxtr@org@gls@ 68
\@glsxtr@org@glsdohypertarget 129
\@glsxtr@org@glsignore 65, 66
\@glsxtr@org@glspl@ 68
\@glsxtr@org@glsxtrtitlefirst 320, 321
\@glsxtr@org@glsxtrtitlefirstplural 320, 321
\@glsxtr@org@glsxtrtitlefull 320, 322
\@glsxtr@org@glsxtrtitlefullpl 320, 322
\@glsxtr@org@glsxtrtitlelong 320, 322
\@glsxtr@org@glsxtrtitlelongpl 320, 322
\@glsxtr@org@glsxtrtitlename 320, 321
\@glsxtr@org@glsxtrtitleorpdforheading 320, 321
\@glsxtr@org@glsxtrtitleplural 320, 321
\@glsxtr@org@glsxtrtitleshort 320, 321
\@glsxtr@org@glsxtrtitleshortpl 320, 321
\@glsxtr@org@glsxtrtitletext 320, 321
\@glsxtr@org@makeglossaries 122, 123
\@glsxtr@org@markboth 319
\@glsxtr@org@markright 318, 319
\@glsxtr@org@newacronymstyle 121, 122
\@glsxtr@org@postdescription 128, 196
\@glsxtr@org@see@noindex 141
\@glsxtr@org@setacronymstyle 121, 122
\@glsxtr@org@theHvalue 8, 9
\@glsxtr@org@unset@buffer 105, 106
\@glsxtr@orgprefix 10, 11
\@glsxtr@orgprintglossary 62, 128
\@glsxtr@orgwarndep 202
\@glsxtr@p@acrlong@ 96
\@glsxtr@p@acrlongpl@ 96
\@glsxtr@p@acrshort@ 96
\@glsxtr@p@acrshortpl@ 96
\@glsxtr@p@long@ 96
\@glsxtr@p@longpl@ 96
\@glsxtr@p@plural@ 96
\@glsxtr@p@short@ 96
\@glsxtr@p@shortpl@ 96
\@glsxtr@p@text@ 96
\@glsxtr@pagestag 65
\@glsxtr@pagetag 65
\@glsxtr@prefix 348
\@glsxtr@prevunitcount 116
\@glsxtr@printglossnr 128
\@glsxtr@printglossopts 62, 126, 128
\@glsxtr@printglossval 128
\@glsxtr@printunsrtglossaryskipentry 148, 149
\@glsxtr@provide@addstoragekey 32
\@glsxtr@provide@storagekey 31
\@glsxtr@providenewgls 153
\@glsxtr@record 15, 16, 67–69, 74, 209–219
\@glsxtr@record@noglossarywarning 15
\@glsxtr@record@only@setup 16
\@glsxtr@record@setting 8, 10, 11, 14, 15, 30, 52, 57, 58, 93, 123, 141, 143, 152
\@glsxtr@record@setting@alsoindex 8, 10, 52, 123

\@glsxtr@record@setting@nameref 11, 14, 30, 141, 143
\@glsxtr@record@setting@off 57, 93, 152
\@glsxtr@record@setting@only 14, 30
\@glsxtr@recordsee 15, 28, 52
\@glsxtr@redef@forglsentries 7, 29
\@glsxtr@redefstyles 24, 338
\@glsxtr@reg@glosslist 123–126, 129
\@glsxtr@restore@postpunc 128
\@glsxtr@rglstrigger@record 157–159
\@glsxtr@s@longnewglossaryentry 43
\@glsxtr@savepreloctag 65, 66
\@glsxtr@setentrycountunsetattr 108
\@glsxtr@setentryunitcountunsetattr 119
\@glsxtr@setupshortcuts 22, 23, 28, 29
\@glsxtr@shortcutsnr 22
\@glsxtr@shortcutsval 22, 143
\@glsxtr@swaptwo 201
\@glsxtr@tag 195
\@glsxtr@taggingcs 195
\@glsxtr@textformat 73, 74
\@glsxtr@theHvalue 8–10, 70, 72–75, 156
\@glsxtr@thevalue 8–10, 70, 72–75, 156
\@glsxtr@thisloctag 65, 66
\@glsxtr@titlelabel 131, 132, 150
\@glsxtr@tmp 24, 71, 134
\@glsxtr@type 181
\@glsxtr@unitcountlist 114
\@glsxtr@unset 105, 106
\@glsxtr@unset@buffer 105, 106
\@glsxtr@unsrt@getgrouptitle 148
\@glsxtr@use@equation@counter 9, 71, 72
\@glsxtr@usesee 49
\@glsxtr@warn@onexistsordo 7, 15, 16
\@glsxtr@warn@undefaction 7, 15, 16
\@glsxtr@wrglossary@locationhyperlink 26
\@glsxtr@wrglossnr 70
\@glsxtr@wrglossval 70
\@glsxtrbuffer@nodup@unset 105
\@glsxtrbuffer@unset 105
\@glsxtrdialecthook 338
\@glsxtrdocdeffalse 58
\@glsxtrentryfmt 34
\@glsxtrfmt 32
\@glsxtrglossentry 145
\@glsxtrglossentryother 146
\@glsxtrhypernameprefix 129, 150
\@glsxtrifhasfield 35
\@glsxtrifhyphenstart 301
\@glsxtrindexaliased 91
\@glsxtrindexcrossrefsfalse 17
\@glsxtrindexcrossrefstrue 17
\@glsxtrinmark 319
\@glsxtrlong 96, 214
\@glsxtrlongpl 96, 217
\@glsxtrnewgls 154
\@glsxtrnewgls@inner 152, 153
\@glsxtrnewgls@innercsname 153
\@glsxtrnotinmark 319
\@glsxtrp 101, 102
\@glsxtrp@opt 99, 100
\@glsxtrpl 61, 62
\@glsxtrpostloctag 64–66
\@glsxtrpreloctag 64–66
\@glsxtrsetaliasnoindex 90, 91
\@glsxtrshort 96, 212
\@glsxtrshortpl 96, 216
\@glsxtrundeftag 6, 30
\@glsxtrwrglossmark 26, 27
\@gobble 7, 16, 18, 76, 149, 150, 203, 421–423
\@gobbletwo 202
\@gtempa 339
\@ifdefinable 339
\@ifnextchar 92, 93
\@ifpackageloaded 5, 20, 143, 162, 181, 183, 185, 187, 189, 338, 351, 382
\@ifstar 31, 32, 35, 43, 44, 46, 59, 92, 105, 106, 147, 194
\@ifundefined 338, 339
\@ignored@glossaries 44–46
\@input 141
\@input@ 136
\@istfilename 123
\@makeglossary 123
\@mfu@domakefirstuc 195, 196
\@mfu@nocaplist 196
\@ne 110, 119, 153
\@newglossaryentry@defcounters 108, 116
\@newglossaryentryposthook 13, 14, 31, 53, 87
\@newglossaryentryprehook 13, 14, 31, 43, 53, 87
\@nil 135, 151, 343
\@nnil 191, 193, 194, 201–203
\@no@glsxtrindexaliased 91
\@no@makeglossaries 140
\@nocounterr 161

\@nopostdesc 127
\@onelevel@sanitize 13, 53, 62, 132, 150
\@onlypreamble 62, 65, 118, 141, 144, 161, 190, 192–194
\@org@glossaryentrynumbers 127
\@org@newglossaryentryprehook 43, 44
\@print@unsrt@glossary 147
\@printgloss@setsort 126, 127
\@printglossary 62, 147
\@printunsrt@glossary@handler 148
\@printunsrtglossary 147
\@rGLS 158
\@rGLS@ 158
\@rGLSpl 159
\@rGLSpl@ 159
\@rGls 157
\@rGls@ 158
\@rGlspl 158
\@rGlspl@ 158
\@rc@ifdefinable 339
\@rgls 157
\@rgls@ 157
\@rglspl 157
\@rglspl@ 157
\@sGlsXtrEnableOnTheFly 59
\@secondofthree 76–78, 83, 85–88, 209, 211, 213, 215, 216, 218
\@secondoftwo 69, 75, 79–87, 89, 95, 122, 129, 187, 201, 209, 210, 212–219, 233, 255, 269, 291, 320, 321
\@sglsxtr@provide@storagekey 31
\@star@or@long 339
\@starttoc 319
\@thirdofthree 76–78, 84–87, 89, 210, 211, 213, 215, 217, 219, 321
\@thirdoftwo 79–83
\@this@key 187
\@tracklang@lang 40, 41
\@warn@nomakeglossaries 137
\@xdy@main@language 136
\@xdycrossrefhook 52
\@xdylanguage 136
\@xdylocationclassorder 52
[package 342
\\ 135, 343

\␣ 58, 138, 139

A

\AA 366
\aa 366
\AB 20
\Ab 20
\ab 20
abbreviation styles:
  long-hyphen-postshort-hyphen 307, 309
  long-hyphen-short-hyphen 303, 308
  long-postshort-user 296
  long-short-user 294
  nolong-short 238
  short 236
  short-hyphen-long-hyphen 311, 313
  short-hyphen-postlong-hyphen 312, 315
  short-long-user 296
  short-nolong 235, 238
  short-nolong-desc 237
  short-postlong-user 298
\abbreviationsname 20
\abbrvpluralsuffix 143, 170, 171, 205, 226, 228, 231, 233, 234, 236, 239, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 308, 311, 313, 316
\ABP 20
\Abp 20
\abp 20
\AC 21
\Ac 21
\ac 21
\ACF 21
\Acf 21
\acf 21
\ACFP 21
\Acfp 21
\acfp 21
\ACL 21
\Acl 21
\acl 21
\ACLP 21
\Aclp 21
\aclp 21
\ACP 21
\Acp 21
\acp 21
\ACRfullfmt 120

\Acrfullfmt 120
\acrfullfmt 120
\ACRfullplfmt 121
\Acrfullplfmt 121
\acrfullplfmt 121
\acronymentry 120
\acronymfont 83–87, 98, 99, 121, 122
\acronymname 20
\acronymsort 120
\acronymtype 20, 120, 121
\acrpluralsuffix 120, 143
\ACS 21
\Acs 21
\acs 21
\ACSP 21
\Acsp 21
\acsp 21
\actualchar 193
\addtolength 412
\advance 110, 119, 142, 153
\AF 21
\Af 20
\af 20
\AFP 21
\Afp 20
\afp 20
\AL 21
\Al 20
\al 20
\ALP 21
\Alp 20
\alp 20
amsmath package 12, 25
\AnyTrackedLanguages 338, 382
\appto 13, 14, 24, 31, 49, 52–55, 87, 92, 108, 116, 148, 150, 189, 200, 203, 347, 383
\apptoglossarypreamble 344–346
\arabic 26, 423
\AS 21
\As 20
\as 20
\ASP 21
\Asp 20
\asp 20
\AtBeginDocument 26, 30, 63, 64, 143, 348
\AtEndDocument 55, 109, 117, 136, 137

B

babel package 190, 192, 200
\begin 133, 138, 148, 385, 387–394, 414–418, 421, 429, 431–438, 440–446
\begingroup 8, 9, 33, 71, 74, 91, 145–147, 160, 339, 342, 343, 348
\bgroup 43, 127
bib2gls 25, 33, 93, 141, 150, 152, 156, 157, 338, 339, 341, 344, 346, 348, 350
\bibglshrefchar 141
\bottomrule 427, 430, 432, 433, 435, 436, 438, 439, 441, 443, 444, 446

C

\c@wrglossary 26
\catcode 57, 141
category attributes:
  accessinsertdots 170
  aposplural 205
  discardperiod 199
  entrycount 104, 107–109, 119
  externallocation 341
  firstshortaccess 172
  firstuc 185
  glossdesc 181
  glossdescfont 181
  glossname 183
  glossnamefont 183, 185
  headuc 322
  indexname 190
  indexonlyfirst 91
  insertdots 204
  linkcount 160
  linkcountmaster 160
  markshortwords 204
  markwords 204, 205, 301, 303, 311
  nameshortaccess 171
  nohyper 90
  nohyperfirst 76–78
  noshortplural 205
  regular 67, 113, 225–230, 232, 235, 237, 238, 240–242, 244, 245, 247, 248, 251–253, 255, 258–262, 265–267, 269, 272–274, 276, 277, 279, 280, 282, 284, 286, 288–290, 293, 299–301, 303–306, 311, 312, 316, 318
  textformat 73
  textshortaccess 171
\cdot 26
\centering 420
\cGLS 20, 21, 107, 119

\cGls 20, 21, 107, 119
\cgls 20, 21, 107, 119
\cGLSformat 111
\cGlsformat 111
\cglsformat 110, 112
\cGLSpl 20, 21, 107, 119
\cGlspl 20, 21, 107, 119
\cglspl 20, 21, 107, 119
\cGLSplformat 112
\cGlsplformat 111
\cglsplformat 111, 113
\changes 324
\char 131
\columnwidth 63, 64
\count@ 110, 118, 119
\csappto 42
\csdef 31, 37, 38, 42, 87–89, 109, 114, 115, 117, 176, 187, 197, 198, 223, 224, 232, 255, 269, 290, 294, 296–298, 308, 310, 313, 315, 384, 402
\cseappto 47
\csedef 115
\csgdef 38, 44–46, 58, 65, 109, 115, 117, 118, 402, 424
\cslet 38, 43, 44, 127
\csletcs 38, 224
\csname 6, 32, 45, 53, 58, 63, 66, 69, 72, 74, 83–89, 91, 100, 115, 124, 133, 136, 139, 140, 148, 153–156, 161, 181, 202, 209–219, 225, 411
\cspreto 42
\csuse 9, 12, 33, 34, 45, 55, 65, 88, 89, 101–103, 114–116, 118, 126, 131, 133, 134, 145, 146, 149–151, 176, 187, 197, 198, 223, 224, 344, 347, 402, 403, 405, 406, 427
\csxdef 49, 53, 54, 115, 118
\currentglossary 127, 145, 146, 423
\CurrentOption 27, 383, 384
\CurrentTrackedLanguage 381
\CurrentTrackedLanguageTag 143
\CurrentTrackedScript 381, 382
\CurrentTrackedTag 338, 382
\CustomAbbreviationFields 206, 226–230, 232, 234, 236, 239, 241–247, 249, 253, 254, 257–260, 262, 264, 267, 269, 271, 272, 274–279, 281, 283, 286, 288, 290, 293, 294, 296, 298–301, 303–306, 308, 309, 311–313, 315–317

D

datatool-base package 12
\DeclareAcronymList 120
\DeclareOption 5, 383
\DeclareOptionX 5, 27
\def 9, 10, 13–17, 27, 29, 30, 33, 36, 43, 45, 49, 53, 56, 59–62, 64, 66–87, 93, 95–99, 105, 110–113, 120, 124, 126–130, 132–136, 148, 150, 151, 153, 156–159, 169–171, 191–196, 200–205, 208–219, 222, 302, 304, 307, 310, 312, 313, 341, 343, 347, 348, 382, 396, 411–413, 417, 418, 427, 428
\defglsentryfmt 44–47
\define@boolkey 17, 18, 70, 90
\define@choicekey 7, 15, 17, 19, 22, 23, 26, 70, 128
\define@key 13, 14, 19, 24, 25, 31, 53, 70, 74, 87, 129, 169, 202
\DefineAcronymSynonyms 22
\delimN 65, 66
\delimR 65, 66
\descriptionname 427, 430, 432, 433, 435, 436, 438, 439, 441, 442, 444, 446
\detokenize 59
\dimen@ 122, 402–410, 427, 428
\dimen@i 405, 406
\dimen@ii 402, 403, 405, 406, 427
\dimexpr 63, 64, 401, 428, 429
\disable@keys 20, 30, 58, 141
\do 6, 24, 35, 56, 75, 107, 108, 119, 123, 126, 133, 148, 155, 161, 180, 187, 195, 348
\do@gls@link@checkfirsthyper 33, 67, 69, 72, 83–87, 209–219
\do@glsdisablehyperinlist 72, 90
doc package 193
\dolistcsloop 35
\DTLifinlist 71, 124, 125, 129, 348
\DTLifint 131

E

\eappto 13, 24, 44–46, 148, 151, 170–172, 190, 383
\edef 6, 8, 9, 11, 40, 44–47, 52, 54, 55, 57, 58, 72–74, 89, 91, 93, 95, 114, 115, 117, 123, 124, 129, 131, 132, 134–137, 141, 145, 146, 156, 160, 181–184, 186–188, 191, 193, 194, 202, 343, 348, 381, 405, 406, 421, 422, 429, 431–438, 440–446
\eglssetwidest 404–410

\egroup 43, 44, 127
\else 8–14, 17–20, 22–24, 27–30, 33, 36, 41, 42, 57–59, 64, 66, 69, 73, 74, 91, 92, 110, 122, 123, 125, 127–129, 132, 134, 135, 138, 140, 142, 156, 157, 189–191, 193, 194, 201, 203, 205, 212–219, 221, 222, 226, 228, 229, 231–240, 243, 245–259, 261–273, 275, 277–292, 294, 295, 297, 298, 300–302, 304, 307–310, 313, 314, 316, 317, 343–347, 385, 388–398, 400, 411, 412, 418, 420, 422, 429–447
\emph 270, 271
\empty 134, 135, 343, 347, 348
\encapchar 193
\end 133, 138, 139, 148, 385, 387–394, 414–418, 421, 429–435, 437–443, 445, 446
\end@getprefix 30
\end@glsxtr@display@loc 134
\endcsname 6, 32, 45, 53, 58, 63, 66, 69, 72, 74, 83–89, 91, 100, 115, 124, 133, 136, 139, 140, 148, 153–156, 161, 181, 202, 209–219, 225, 411
\endfoot 427, 430, 432–434, 436, 438, 439, 441, 442, 444, 445
\endgroup 8, 10, 33, 71, 75, 91, 145, 147, 160, 339, 343, 348
\endhead 427, 430, 432–434, 436, 438, 439, 441, 442, 444, 445
\ensuremath 26
entry categories:
  abbreviation 219
  general 175, 177
  index 179
\entryname 427, 428, 430, 432, 433, 435, 436, 438, 439, 441, 442, 444, 446
\epreto 190
\equal 139, 140, 198
equation (counter) 18, 71
\escapechar 339
etoolbox package 5
\expandafter 27, 32, 33, 35, 39, 49, 52, 55, 56, 59–61, 71, 88, 89, 93, 100, 106, 112–114, 124–126, 129, 133–135, 148, 151, 153, 160, 170, 171, 181, 184, 185, 187, 189, 190, 193, 201, 204, 205, 301, 339, 343, 421
\expandonce 120, 151, 170–172, 191, 227, 305, 429–435, 437–446
\ExtraCustomAbbreviationFields 170–172, 204, 206

F

\fi 7–20, 22–24, 27, 28, 30, 33, 36, 41, 42, 49, 53–55, 57–60, 63, 64, 66, 69, 70, 72–74, 91, 92, 110, 118, 119, 122, 125–129, 132, 134, 135, 137, 138, 140, 142–144, 156, 157, 189–192, 194, 201, 203, 205, 212–219, 221, 222, 226, 228, 229, 231–240, 243, 245–259, 261–273, 275, 277–292, 294, 295, 297, 298, 300, 302, 304, 307–310, 313, 314, 316, 317, 343–347, 385, 388–398, 400, 402–412, 418, 420, 423, 428, 430, 431, 433–437, 439–447
first use 448
  flag 448
  text 448
\firstacronymfont 121, 122
fontspec package 143
\footnote 230
\forallglossaries 55, 147, 179, 181, 404–410
\forallglsentries 57, 110, 118
\ForEachTrackedDialect 338, 382
\foreignlanguage 39, 41
\forglsentries 6, 55, 179, 181, 404–410
\forlistcsloop 35, 118, 133
\forlistloop 106, 130, 196
\futurelet 200

G

\gdef 65, 192, 193
\Genacrfullformat 120, 121
\genacrfullformat 120, 121
\GenericAcronymFields 120
\Genplacrfullformat 121
\genplacrfullformat 121
\GetTrackedDialectFromLanguageTag 39
\GetTrackedDialectToMapping 40
\glo@grabfirst 151
\glo@name 184, 185, 189
\gloaliaslabel 94, 95
\global 10, 43, 44, 57, 127, 151
\glolinkprefix 70, 72–74, 95, 129, 143
glossaries package 12, 15, 27–29, 41, 49, 52–54, 126, 384
glossaries-accsupp package 23, 27, 162, 205
glossaries-extra package 2, 382
glossaries-extra-bib2gls package 15, 16, 30, 338, 382, 428

glossaries-extra-stylemods package 24, 197, 338, 345
glossaries-prefix package 348
glossaries-stylemods package 419
glossaries.sty package 44
\GlossariesExtraWarning 6, 16, 18, 23, 30, 40–42, 60, 62, 73, 121, 124, 134, 138, 140, 141, 147, 181–184, 186, 188, 195, 224, 339, 346, 347
\GlossariesExtraWarningNoLine 18, 110, 119
\GlossariesWarning 65, 124, 126, 130, 131, 222
\GlossariesWarningNoLine 124, 137
glossary styles:
  altlist 385
  altlistgroup 386
  altlisthypergroup 387
  alttree 344, 345, 401, 402, 411, 427
  alttreegroup 412
  alttreehypergroup 413
  index 396
  indexgroup 397
  indexhypergroup 397
  inline 395
  list 385
  listdotted 384
  listdottedstyle 385
  listgroup 386
  listhypergroup 386
  longragged-booktabs 429
  mcolalttree 416
  mcolalttreegroup 417
  mcolalttreehypergroup 417
  mcolalttreespannav 418
  mcolindexgroup 413
  mcolindexhypergroup 413
  mcolindexspannav 414
  mcoltreegroup 414
  mcoltreehypergroup 415
  mcoltreenonamegroup 415
  mcoltreenonamehypergroup 416
  mcoltreenonamespannav 416
  mcoltreespannav 415
  name-desc 432
  sublistdotted 385
  tree 397
  treegroup 398
  treehypergroup 399
  treenoname 399
  treenonamegroup 400
  treenonamehypergroup 400
glossary-bookindex package 384
glossary-hypernav package 93
glossary-long package 389
glossary-longbooktabs package 389
glossary-longextra package 344–346
glossary-tree package 345, 346, 396
\glossaryentrynumbers 66, 127, 151, 152
\glossaryheader 133, 148, 385–394, 396–400, 411, 413–417, 421, 429–435, 437–443, 445, 446
\glossaryname 126
\glossarypostamble 133, 148, 150
\glossarypreamble 133, 147
\glossarysection 133, 139, 141, 147, 150
\glossarytitle 45, 126, 127, 133, 139, 141, 147
\glossarytoctitle 45, 126, 133, 139, 141, 147
\glossentry 127, 152, 384–394, 396, 398, 400, 411, 421, 430, 431, 433, 434, 436, 437, 439, 440, 442, 443, 445, 447
\glossentrydesc 384, 385, 387–395, 397–399, 401, 425
\glossentryname 145, 384–394, 396, 398, 400, 411, 412, 419, 425
\glossentrynameother 147
\glossentrysymbol 388–395, 398, 399, 401, 425
\glossxtrsetpopts 197
\GLS 107, 119, 155
\Gls 61, 107, 119, 155
\gls 41, 42, 60, 62, 107, 119, 125, 138, 155
\gls@assign@desc 43, 44
\gls@assign@field 13, 14, 32, 87
\gls@checkseeallowed 57, 59, 123
\gls@codepage 137
\gls@defdocnewglossaryentry 57, 108, 116
\gls@defglossaryentry 43, 44, 57, 60, 61
\gls@dotoctitle 127
\gls@glossary 53
\gls@grplabel 93
\gls@ifnotmeasuring 12, 107
\gls@level 151
\gls@noidxglossary 124
\gls@org@glossaryentryfield 127
\gls@org@glossarysubentryfield 127
\gls@orgTrackLangRequireDialectPrefix 382
\gls@save@numberlist 64, 66
\gls@set@xr@key 53
\gls@tmplen 404–410, 412, 428

\gls@type 124
\glsabbrvdefaultfont 208, 226, 228, 231, 233, 234, 236, 239, 292, 302, 305, 315
\glsabbrvemfont 270–279, 281, 283, 285, 287, 289–291
\glsabbrvfont 97, 98, 121, 208, 212, 213, 216, 217, 219, 221, 222, 225–234, 236, 239, 241, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 299–301, 303, 305, 307–309, 311, 313, 314, 316
\glsabbrvhyphenfont 302–304, 308, 310–313, 315
\glsabbrvonlyfont 315, 316, 318
\glsabbrvscfont 242–248, 250, 251, 253–255
\glsabbrvsmfont 256–260, 262, 264, 265, 267, 269
\glsabbrvuserfont 292–298, 300
\GLSaccessdesc 79
\Glsaccessdesc 79, 182, 194
\glsaccessdesc 79, 182, 198, 199
\GLSaccessdescplural 80
\Glsaccessdescplural 80
\glsaccessdescplural 80
\GLSaccessfirst 77
\Glsaccessfirst 77
\glsaccessfirst 77
\GLSaccessfirstplural 78
\Glsaccessfirstplural 78
\glsaccessfirstplural 78
\Glsaccesslong 86, 207, 215, 226, 235, 239, 240, 243, 249, 250, 252, 257, 263–266, 272, 273, 281–287, 294, 295, 303, 305, 306, 309, 311, 317
\glsaccesslong 85, 86, 206, 214, 215, 226, 228, 229, 231–234, 236–240, 243, 245–254, 256, 257, 259, 261–268, 270, 271, 273, 275, 277, 278, 280–291, 294, 295, 298, 300, 303, 305, 306, 309, 311, 316, 317
\Glsaccesslongpl 87, 207, 218, 226, 235, 240, 243, 249, 250, 252, 258, 263–266, 272, 273, 281–287, 294, 295, 303, 305, 306, 309, 311, 317
\glsaccesslongpl 86, 87, 207, 218, 219, 226, 229, 231–240, 243, 245–254, 256, 257, 259, 261–268, 270, 271, 273, 275, 277–292, 294, 295, 298, 300, 301, 303, 305, 306, 309, 311, 316, 317
\GLSaccessname 51, 79
\Glsaccessname 50, 51, 79
\glsaccessname 50, 51, 79
\GLSaccessplural 78
\Glsaccessplural 77
\glsaccessplural 77
\GLSaccessshort 51
\Glsaccessshort 50, 51, 84, 213, 222, 229, 231–233, 237, 238, 245, 247, 248, 254–256, 259, 261, 262, 268, 270, 275, 277, 278, 280, 289–291, 297, 298, 300, 309, 314
\glsaccessshort 50, 51, 83, 84, 206, 207, 212, 213, 221, 226, 228, 231, 233–238, 240, 243, 245–250, 252–257, 259–273, 275, 277–282, 284–287, 289, 291, 294, 295, 297, 298, 300, 303, 305, 306, 308, 311, 314, 317
\Glsaccessshortpl 85, 216, 222, 229, 231–234, 237, 238, 245, 247, 248, 254, 256, 259, 261, 262, 268, 270, 275, 277, 278, 280, 289–292, 297, 298, 300, 309, 314, 317
\glsaccessshortpl 84, 85, 207, 216, 217, 222, 226–228, 231, 233–238, 240, 243, 245–259, 261–263, 265, 266, 268–273, 275, 277–282, 284, 286, 287, 289–291, 294, 295, 297, 298, 300, 303, 306, 308, 311, 314, 317
\GLSaccesssymbol 80
\Glsaccesssymbol 80, 194
\glsaccesssymbol 80, 194, 199
\GLSaccesssymbolplural 81
\Glsaccesssymbolplural 81
\glsaccesssymbolplural 81
\GLSaccesstext 76
\Glsaccesstext 76
\glsaccesstext 50, 76
\glsacrshortcutstrue 22, 23
\glsacspacemax 122
\glsadd 33, 56, 71, 75, 138
\glsadd options
  theHvalue 9
  thevalue 9, 10
\glsaddpostsetkeys 10, 75
\glsaddpresetkeys 10, 74
\glsaddstoragekey 55, 175, 341
\glsbackslash 59
\glscapscase 69, 75–89, 209–221

\glscategory 67, 75, 89, 97, 98, 176–178, 181–188, 194, 197, 198, 209–213, 216, 217
\glscategorylabel 89, 170–172, 202, 204, 205, 232, 255, 269, 290, 294, 296–298, 308, 310, 313, 315
\glsclosebrace 52, 139, 140
\glscounter 9, 18
\glscurrententrylabel 64–66, 127, 136, 145, 146, 148, 149, 196, 197
\glscurrentfieldvalue 33–36, 39, 40, 50, 51, 292, 340
\glscustomtext 67, 69, 83–87, 209–219, 221
\glsdefaulttype 6, 19, 42, 126, 137, 147, 149
\glsdescriptionaccessdisplay 166, 182
\glsdescriptionpluralaccessdisplay 166, 167
\glsdescwidth 387–394, 426, 428, 429
\glsdetoklabel 8, 9, 31, 34–38, 41–44, 47–49, 52, 56, 58, 59, 72, 74, 91, 95, 108, 109, 114–118, 124, 127, 130, 131, 145, 146, 151, 154–156, 181, 184, 185, 189, 405, 406
\glsdisp 350
\glsdisplaynumberlist 125, 130
\glsdohyperlink 93, 95, 96, 344
\glsdohypertarget 95, 129
\glsdoifexists 17, 27, 37, 43, 47, 49–51, 53, 67, 69, 74, 83–87, 95, 107, 124, 130, 131, 145, 146, 209–219
\glsdoifexistsordo 33, 34, 69
\glsdoifexistsorwarn 17, 181, 182, 194
\glsdoifnoexists 43
\glsdonohyperlink 74, 95, 96
\glsdosanitizesort 125
\glsenableentrycount 107, 110, 118
\glsenableentryunitcount 109, 119
\glsentrycounter 26, 135, 343
\GlsEntryCounterLabelPrefix 41
\glsentrycurrcount 108, 110, 116
\Glsentrydesc 166, 174, 182
\glsentrydesc 166, 174, 183
\Glsentrydescplural 167, 174
\glsentrydescplural 166, 167, 174
\Glsentryfirst 113, 159, 164, 173
\glsentryfirst 113, 159, 164, 173, 334
\Glsentryfirstplural 113, 160, 164, 173
\glsentryfirstplural 113, 159, 164, 165, 173, 335
\glsentryfmt 44–47
\Glsentryfull 121
\glsentryfull 121
\Glsentryfullpl 121
\glsentryfullpl 121
\glsentryitem 145, 146, 384–394, 396, 398, 400, 411, 422, 425
\Glsentrylong 98, 99, 113, 159, 168, 175
\glsentrylong 98, 99, 113, 159, 168, 175, 232, 255, 269, 290, 297, 298, 313, 335
\Glsentrylongpl 98, 99, 113, 169, 175
\glsentrylongpl 98, 99, 113, 169, 175, 336
\Glsentrylongplural 160
\glsentrylongplural 159
\Glsentryname 162, 172, 183, 185, 186
\glsentryname 145, 162, 172, 190, 332, 333, 404–410, 424
\glsentrynumberlist 125, 131, 408–410
\Glsentryplural 163, 173
\glsentryplural 163, 172, 173, 333, 334
\glsentryprevcount 108, 110, 116
\glsentryprevmaxcount 116
\glsentryprevtotalcount 116
\Glsentryshort 97, 98, 167, 174
\glsentryshort 97, 98, 122, 167, 174, 294, 296, 307, 331, 332
\Glsentryshortpl 98, 99, 168, 174
\glsentryshortpl 98, 99, 168, 174, 175, 331, 332
\Glsentrysymbol 165, 173
\glsentrysymbol 165, 173, 407–409
\Glsentrysymbolplural 166, 173
\glsentrysymbolplural 165, 166, 173, 174
\Glsentrytext 163, 172
\glsentrytext 95, 162, 163, 172, 333
\glsentrytype 146
\Glsentryuseri 81
\glsentryuseri 81
\Glsentryuserii 81
\glsentryuserii 82
\Glsentryuseriii 82
\glsentryuseriii 82
\Glsentryuseriv 82
\glsentryuseriv 82
\Glsentryuserv 82
\glsentryuserv 83
\Glsentryuservi 83
\glsentryuservi 83
\glsextrapostnamehook 187
\glsfieldfetch 94

\glsfieldxdef 180, 181
\glsFindWidestLevelTwo 347
\glsFindWidestTopLevelName 346
\glsfindwidesttoplevelname 403
\GLSfirst 326
\Glsfirst 327
\glsfirst 326
\glsfirstabbrvdefaultfont 208, 226, 228, 231, 233, 234, 236, 239, 305
\glsfirstabbrvemfont 271–292
\glsfirstabbrvfont 121, 206, 207, 226–240, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 306, 308, 311, 313, 316
\glsfirstabbrvhyphenfont 302–304, 307, 308, 310–315
\glsfirstabbrvonlyfont 316, 317
\glsfirstabbrvscfont 242–256
\glsfirstabbrvsmfont 257–270
\glsfirstabbrvuserfont 293–301
\glsfirstaccessdisplay 164
\glsfirstlongdefaultfont 226, 228, 234, 236, 239, 242–252, 257–267, 271, 272, 274, 275, 278–283, 285, 286
\glsfirstlongemfont 273, 274, 276, 277, 283, 284, 286–288
\glsfirstlongfont 206, 207, 226–229, 231, 233–241, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 308, 311, 313, 316
\glsfirstlongfootnotefont 230–234, 253–256, 267–270, 288–292
\glsfirstlonghyphenfont 302–313
\glsfirstlongonlyfont 316–318
\glsfirstlonguserfont 293–301
\GLSfirstplural 327
\Glsfirstplural 327
\glsfirstplural 327
\glsfirstpluralaccessdisplay 164, 165
\glsforeachincategory 222
\glsgenentryfmt 67
\glsgetattribute 73, 94, 110, 114–116, 136, 156, 160, 181–184, 186, 188, 190
\glsgetcategoryattribute 176
\glsgetgrouptitle 386, 387, 397, 399–401, 412–418
\glsgetwidestname 402
\glsgroupheading 151, 385–394, 396–401, 411–418, 423, 430, 431, 433, 434, 436, 437, 439, 440, 442, 443, 445, 447
\glsgroupskip 151, 385, 387–395, 397, 398, 400, 412, 423, 430, 431, 433, 434, 436, 437, 439, 441, 442, 444, 445, 447
\glshasattribute 73, 94, 110, 114, 115, 117, 118, 136, 156, 160, 181–186, 188, 190, 226–230, 232, 236–238, 241–244, 246, 253, 255, 257–260, 267, 269, 271–274, 276, 277, 285, 288–290, 293, 294, 296, 297, 299–301, 303–306, 308, 310–313, 315, 316, 318
\glshascategoryattribute 177
\glshex 352–358, 360–366, 368–371, 373–381
\glshyperlink 95, 340
\glshypernavsep 133
\glshypernumber 136, 189, 341–343
\glsifattribute 69, 70, 76, 90, 92, 101, 160, 179, 182–185, 188, 189, 196, 199, 200, 322–331
\glsifcategory 179
\glsifcategoryattribute 89, 170–172, 177, 178, 204, 205
\glsifnotregular 76
\glsifnotregularcategory 178
\glsifplural 69, 75, 77, 78, 80, 81, 83–87, 199, 200, 209–220
\glsifregular 67, 75, 113, 159, 160
\glsifregularcategory 178
\glsifusetranslator 45
\glsignore 65, 66
\glsinlinedescformat 395
\glsinlinesubdescformat 395
\glsinsert 69, 75, 83–87, 209–221, 301, 308, 310, 313, 315
\glskeylisttok 120, 204, 206
\glslabel 8, 9, 33, 47, 67, 69–74, 89–91, 94, 95, 122, 156, 160, 161, 197–199, 219–221, 232, 255, 269, 290, 294, 296–298, 308, 310, 313, 315
\glslabeltok 120, 204, 206, 226–232, 234, 236–239, 241–247, 250, 253, 255, 257–260, 262, 264, 267, 269, 271–279, 281, 283, 285, 287–291, 293, 294, 296–301, 303–306, 308, 310–313, 315, 316, 318

\glsletentryfield 191
\glslink 120, 121, 349
\glslink options
  counter 10
  format 189
  hyper 318
  hyperoutside 70
  noindex 8, 9, 90, 318
  textformat 73
  theHvalue 73
  thevalue 73, 152
  wrgloss 8, 69, 70
\glslinkcheckfirsthyperhook 90
\glslinkpostsetkeys 10, 73, 156
\glslinkpresetkeys 10, 72, 156
\glslinkvar 92, 93
\glslistchildpostlocation 385
\glslistchildprelocation 385, 386
\glslistdesc 385, 386
\glslistdottedwidth 384
\glslistgroupheaderfmt 386, 387
\glslistnavigationitem 386, 387
\glslistprelocation 385, 386
\glslocalunset 69, 156
\glslongaccessdisplay 168
\glslongdefaultfont 208, 226, 228, 230, 234, 236, 239, 243, 245, 246, 248–252, 257, 259, 260, 262, 264–266, 271, 275, 278, 279, 281, 282, 285, 293, 302, 315
\glslongemfont 271, 273, 276, 283, 286, 287
\glslongextraDescAlign 429–432, 434, 435, 437–446
\glslongextraDescFmt 426, 430, 431, 433, 434, 436, 437, 439, 440, 442, 443, 445, 447
\glslongextraDescNameHeader 433
\glslongextraDescNameTabularFooter 432
\glslongextraDescNameTabularHeader 432
\glslongextraDescSymNameHeader 445
\glslongextraDescSymNameTabularFooter 444, 445
\glslongextraDescSymNameTabularHeader 444, 445
\glslongextraGroupHeading 430, 431, 433, 434, 436, 437, 439, 440, 442, 443, 445, 447
\glslongextraHeaderFmt 427, 428, 430, 432, 433, 435, 436, 438, 439, 441, 442, 444–446
\glslongextraLocationAlign 431, 433, 434, 437, 440, 443, 446
\glslongextraLocationDescNameHeader 434
\glslongextraLocationDescNameTabularFooter 433, 434
\glslongextraLocationDescNameTabularHeader 433, 434
\glslongextraLocationDescSymNameHeader 446
\glslongextraLocationDescSymNameTabularFooter 445, 446
\glslongextraLocationDescSymNameTabularHeader 445, 446
\glslongextraLocationFmt 431, 434, 437, 440, 443, 447
\glslongextraLocationSymDescNameHeader 443
\glslongextraLocationSymDescNameTabularFooter 442, 443
\glslongextraLocationSymDescNameTabularHeader 442, 443
\glslongextraLocSetDescWidth 431, 433, 434
\glslongextraNameAlign 429–432, 434, 435, 437, 438, 440–446
\glslongextraNameDescHeader 430
\glslongextraNameDescLocationHeader 431
\glslongextraNameDescLocationTabularFooter 430, 431
\glslongextraNameDescLocationTabularHeader 430, 431
\glslongextraNameDescSymHeader 435
\glslongextraNameDescSymLocationHeader 437
\glslongextraNameDescSymLocationTabularFooter 436, 437
\glslongextraNameDescSymLocationTabularHeader 436, 437
\glslongextraNameDescSymTabularFooter 434, 435
\glslongextraNameDescSymTabularHeader 434, 435
\glslongextraNameDescTabularFooter 427, 429
\glslongextraNameDescTabularHeader 427, 429
\glslongextraNameFmt 430, 431, 433, 434, 436, 437, 439, 440, 442, 443, 445, 447
\glslongextraNameSymDescHeader 439
\glslongextraNameSymDescLocationHeader 440

\glslongextraNameSymDescLocationTabularFooter 439, 440
\glslongextraNameSymDescLocationTabularHeader 439, 440
\glslongextraNameSymDescTabularFooter 438
\glslongextraNameSymDescTabularHeader 438
\glslongextraSetDescWidth 428, 429, 432
\glslongextraSubDescFmt 430, 431, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447
\glslongextraSubLocationFmt 431, 434, 437, 440, 444, 447
\glslongextraSubNameFmt 430, 431, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447
\glslongextraSubSymbolFmt 436, 437, 439, 440, 442, 444, 445, 447
\glslongextraSymbolAlign 435, 437, 438, 440–446
\glslongextraSymbolFmt 426, 436, 437, 439, 440, 442, 443, 445, 447
\glslongextraSymDescNameHeader 442
\glslongextraSymDescNameTabularFooter 441
\glslongextraSymDescNameTabularHeader 441
\glslongextraSymLocSetDescWidth 436, 437, 440, 443, 446
\glslongextraSymSetDescWidth 429, 435, 438, 441, 442, 444, 445
\glslongextraTabularVAlign 429, 431–433, 435, 436, 438, 440, 441, 443, 444, 446
\glslongextraUpdateWidest 344–346
\glslongextraUpdateWidestChild 344–347
\GlsLongExtraUseTabularfalse 429
\glslongfont 98, 208, 214, 215, 218, 219, 226–229, 231, 233, 234, 236, 239–241, 243, 245, 246, 248, 250, 251, 253, 255, 257, 259, 260, 262, 264, 265, 267, 269, 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 300, 303, 305, 308, 311, 313, 316, 317
\glslongfootnotefont 230, 231, 233, 253, 255, 267, 269, 289, 291
\glslonghyphenfont 302, 303, 305, 306, 308, 311, 313
\glslongonlyfont 316
\glslongpltok 205, 206, 226–230, 239, 241, 242, 244, 245, 249, 253, 257–260, 264, 267, 271–277, 281, 283, 286, 289, 293, 294, 296–301, 303–306, 308, 310–312, 316, 318
\glslongpluralaccessdisplay 169
\glslongtok 120, 204, 206, 226–230, 232, 234, 236, 239, 241–247, 249, 250, 253, 254, 257, 258, 260, 262, 264, 267, 269, 271–279, 281, 283, 286, 288, 290, 293, 294, 296–301, 303–306, 308, 309, 311–313, 316–318
\glslonguserfont 293, 295–298, 300
\glsmcols 414–418
\GLSname 324
\Glsname 324
\glsname 324
\glsnameaccessdisplay 162, 183, 184, 186
\glsnamefont 183, 185, 186, 188, 428
\glsnavhyperlink 133
\glsnavhyperlinkname 94
\glsnavhypertarget 386, 387, 397, 399, 401, 413–418
\glsnavigation 386, 387, 397, 399, 401, 413–418
\glsnextpages 127
\glsnoidxdisplayloc 130, 131, 342
\glsnoidxdisplayloclisthandler 130
\glsnoidxloclist 131, 151, 152
\glsnoidxnumberlistloophandler 130
\glsnonextpages 127
\glsnonumberlistfalse 64
\glsnonumberlisttrue 64
\glsnopostdotfalse 128
\glsnopostdottrue 128
\glsnumberlistloop 125
\glsnumlistlastsep 130
\glsnumlistsep 130
\glsopenbrace 52, 139, 140
\glsorder 123
\glspagelistwidth 388, 390, 392, 394, 427, 429
\glspar 150
\glspatchLToutput 429, 431, 432, 434, 435, 437, 438, 440, 442, 443, 445, 446
\glspenaltygroupskip 430, 431, 433, 434, 436, 437, 439, 441, 442, 444, 445, 447
\GLSpl 107, 119, 155
\Glspl 61, 107, 119, 155
\glspl 61, 107, 119, 155
\GLSplural 325, 326
\Glsplural 326

\glsplural 325
\glspluralaccessdisplay 163
\glspluralsuffix 143, 204, 208
\glspostdescription 18, 19, 128, 196, 384, 385, 387–395, 397–399, 401, 425
\glspostinline 395
\glspostlinkhook 68, 69, 83–87, 100, 209–219
\glsprestandardsort 125
\glsresetentrylist 133, 148
\glssee 53–55
\glsseeformat 49, 52, 58, 124, 130, 131
\glsseelist 52
\glssetabbrvfmt 67, 75, 97, 98, 181–186, 188, 194, 209–213, 216, 217, 219
\glssetattribute 226–232, 234, 236–239, 241–244, 246, 247, 250, 253, 255, 257–260, 262, 264, 267, 269, 271–274, 276–279, 281, 283, 285, 287–289, 291, 293, 294, 296, 297, 299–301, 303–306, 308, 310–313, 315, 316, 318
\glssetcategoryattribute 108, 119, 122, 155, 161, 176, 177, 179, 180, 195
\glssetnoexpandfield 13, 14
\glssettoctitle 127
\glssetwidest 345, 346
\glsshortaccessdisplay 167
\glsshortpltok 205, 206, 226–230, 232, 234, 236, 243–247, 253, 254, 257–260, 262, 267, 269, 271–279, 288–290, 293, 294, 296–301, 303, 304, 308, 310–313, 315, 316, 318
\glsshortpluralaccessdisplay 168
\glsshorttok 120, 204–206, 225–230, 232, 234, 236, 241, 242, 244–247, 249, 253, 254, 257, 258, 260, 262, 264, 267, 269, 271–279, 281, 283, 288, 290, 293, 294, 296, 298–301, 303, 304, 306, 308, 310–313, 315, 316, 318
\glssubentryitem 146, 384–394, 396, 398, 400, 412, 422, 426
\glssymbolaccessdisplay 165
\glssymbolpluralaccessdisplay 165, 166
\glstarget 145, 147, 384–396, 398, 400, 411, 412, 422, 423, 425, 426
\GLStext 324, 325
\Glstext 325
\glstext 324, 325
\glstextaccessdisplay 162, 163
\glstextformat 69, 73
\glstextup 242
\glstreechilddesc 397, 398
\glstreechildpredesc 398
\glstreechildprelocation 397, 398, 400
\glstreechildsymbol 396, 398
\glstreedefaultnamefmt 395, 396
\glstreedesc 396, 398
\glstreegroupheaderfmt 397, 399–401, 412–418, 420
\glstreeindent 398, 400, 410–412
\glstreeitem 396, 414, 422
\glstreenamebox 411, 412
\glstreenamefmt 395, 396, 398, 400, 402, 404–412
\glstreenavigationfmt 397, 399, 401, 413–418
\glstreenonamechilddesc 400
\glstreenonamedesc 400
\glstreenonamesymbol 400
\glstreepredesc 397, 399
\glstreeprelocation 396, 398, 400, 401
\glstreesubitem 396, 422
\glstreesubsubitem 396, 423
\glstreesymbol 396, 398
\GlstrLetField 38
\glstype 69, 72, 83–87, 156, 209–219
\glsunset 56, 69, 110–112, 157
\glsupdatewidest 344, 345
\glsuserdescription 293, 294, 297, 300
\glswrite 52, 123
\glswriteentry 8, 9
\Glsxtr 62
\glsxtr 62
\glsxtr@@do@wrglossary 8, 10, 13, 15
\glsxtr@addloclistfield 15, 16
\glsxtr@addunused 56
\glsxtr@applyabbrvfmt 219
\glsxtr@applyabbrvstyle 202, 204, 223
\glsxtr@counterrecord 145
\glsxtr@dblfloat 18
\glsxtr@do@alsoindex@wrglossary 16
\glsxtr@do@autoadd 9, 71, 72
\glsxtr@dooption 5, 18, 19, 26, 27, 29
\glsxtr@fields 143
\glsxtr@float 18
\glsxtr@headentry@p 101, 102
\glsxtr@hyperoutsidefalse 70
\glsxtr@hyperoutsidetrue 70
\glsxtr@ifnextpunc 200
\glsxtr@ifpunctoken 201

\glsxtr@inc@linkcount 72, 161
\glsxtr@inc@wrglossaryctr 8, 9, 26, 29
\glsxtr@indexonly@saveentrycounter
. . . . 15, 16, 29
\glsxtr@keylist 60, 61
\glsxtr@label 424
\glsxtr@langtag 143
\glsxtr@linkprefix 143
\glsxtr@loaddialect 338, 382
\glsxtr@locationhypertext 343
\glsxtr@makeglossaries 123
\glsxtr@newabbreviation 121, 203
\glsxtr@next 201
\glsxtr@org@@do@wrglossary 29
\glsxtr@org@dohyperlink 93
\glsxtr@org@getgrouptitle 132
\glsxtr@org@newignoredglossary 44
\glsxtr@orgmakenoidxglossaries 57
\glsxtr@pluralsuffixes 143
\glsxtr@process 148, 149
\glsxtr@provideignoredglossary 46
\glsxtr@punclist 200, 201
\glsxtr@record 11, 143
\glsxtr@record@nameref 12, 143
\glsxtr@record@nr 15
\glsxtr@recordsee 13
\glsxtr@renewcommand 339
\glsxtr@resource 141, 143
\glsxtr@s@newignoredglossary 44
\glsxtr@s@provideignoredglossary . . . 46
\glsxtr@saveentrycounter 8, 10, 13, 91
\glsxtr@setaccessdisplay 188
\glsxtr@setbookindexmark 424
\glsxtr@setup@record 15, 16, 29, 30
\glsxtr@shortcutsval 143
\glsxtr@texencoding 143
\glsxtr@undefaction@nr 7
\glsxtr@undefaction@val 7
\glsxtr@usesee 49
\glsxtr@warnonexistsordo 7, 15, 16, 48
\glsxtr@writefields 141
\glsxtrabbreviationfont 67
\glsxtrabbrvfootnote . . . .
. . . . 230–232, 253–255, 267–269, 288–290
\glsxtrabbrvpluralsuffix 143,
208, 226, 228, 231, 233, 234, 236, 239,
242, 257, 270, 293, 302, 305, 308, 313, 316
\glsxtrabbrvtype 20, 206
\glsxtractivatenopost 127
\glsxtraddallcrossrefs 55
\glsxtralias 91
\glsxtrAltTreeIndent 401, 402
\glsxtralttreeInit 411, 417, 418
\glsxtrAltTreePar 401
\glsxtrAltTreeSetHangIndent . . . 401, 411
\glsxtrAltTreeSetSubHangIndent 412
\glsxtralttreeSubSymbolDescLocation 412
\glsxtralttreeSymbolDescLocation . .
. . . . 401, 412
\glsxtrassignfieldfont 76–83
\glsxtrautoindex 190
\glsxtrautoindexassignsort 190
\glsxtrautoindexentry 190
\glsxtrautoindexesc 190
\glsxtrbibaddress 341
\glsxtrbibauthor 341
\glsxtrbibbooktitle 341
\glsxtrbibchapter 341
\glsxtrbibedition 341
\glsxtrbibhowpublished 341
\glsxtrbibinstitution 341
\glsxtrbibjournal 341
\glsxtrbibmonth 341
\glsxtrbibnote 341
\glsxtrbibnumber 341
\glsxtrbiborganization 341
\glsxtrbibpages 341
\glsxtrbibpublisher 341
\glsxtrbibschool 341
\glsxtrbibseries 341
\glsxtrbibtitle 341
\glsxtrbibtype 341
\glsxtrbibvolume 341
\glsxtrbookindexatendgroup 422
\glsxtrbookindexatsubendgroup 422
\glsxtrbookindexatsubsubendgroup . . 422
\glsxtrbookindexbetween 422
\glsxtrbookindexbookmark 423
\glsxtrbookindexcols 421
\glsxtrbookindexcolspread 421
\glsxtrbookindexfirstmarkfmt 424
\glsxtrbookindexformatheader 423
\glsxtrbookindexgroupskip 423
\glsxtrbookindexlastmarkfmt 424
\glsxtrbookindexmulticolsenv 421
\glsxtrbookindexname 419, 422
\glsxtrbookindexparentchildsep   419, 421

\glsxtrbookindexparentsubchildsep 421, 422
\glsxtrbookindexprelocation 419, 422
\glsxtrbookindexsubbetween 422
\glsxtrbookindexsubname 423
\glsxtrbookindexsubprelocation 423
\glsxtrbookindexsubsubbetween 422
\glsxtrbookindexthepage 424
\glsxtrcat 60, 61
\glsxtrchecknohyperfirst 76–78
\glsxtrcombiningdiacriticIIIrules 354
\glsxtrcombiningdiacriticIIrules 354
\glsxtrcombiningdiacriticIrules 353
\glsxtrcombiningdiacriticIVrules 354
\glsxtrComputeTreeIndent 411, 412
\glsxtrComputeTreeSubIndent 412
\glsxtrcounterprefix 135
\glsxtrcurrencyrules 356
\Glsxtrdefaultsubsequentfmt 222, 223
\glsxtrdefaultsubsequentfmt 221, 223
\Glsxtrdefaultsubsequentplfmt 222, 223
\glsxtrdefaultsubsequentplfmt 222, 223
\GlsXtrDefineAbbreviationShortcuts 22, 23
\GlsXtrDefineAcShortcuts 23
\GlsXtrDefineOtherShortcuts 23
\glsxtrdetoklocation 155
\glsxtrdiscardperiod 197
\glsxtrdisplayendloc 134
\glsxtrdisplayendlochook 134
\glsxtrdisplaysingleloc 134
\glsxtrdisplaystartloc 134
\glsxtrdoautoindexname 92, 187
\glsxtrdopostpunc 232, 255, 269, 290
\glsxtrdowrglossaryhook 92
\GlsXtrDualField 340
\glsxtremsuffix 271, 273, 275, 276, 278, 279, 281, 283, 285, 287, 289, 291
\GlsXtrEnableEntryCounting 119
\GlsXtrEnableEntryUnitCounting 108
\GlsXtrEnableOnTheFly 60, 62
\glsxtrendfor 35
\glsxtrfieldlistgadd 144
\glsxtrfieldtitlecase 182–185, 188
\glsxtrfieldtitlecasecs 181
\glsxtrfieldxifinlist 149
\glsxtrfirstscfont 242
\glsxtrfirstsmfont 256
\GlsXtrFmtDefaultOptions 33
\glsxtrfmtdisplay 33
\glsxtrfmtexternalnameref 343
\GlsXtrFmtField 33, 34
\glsxtrfmtinternalnameref 342, 343
\glsxtrfootnotename 230, 232, 253, 254, 267, 269, 288, 290
\GlsXtrForeignTextField 39
\GlsXtrFormatLocationList 64, 66, 408–410
\GLSxtrfull 21, 329, 330
\Glsxtrfull 20, 21, 330
\glsxtrfull 20, 21, 329, 330
\Glsxtrfullformat 207, 221, 223, 224, 226, 229, 231, 233, 235, 237, 240, 243, 245, 247, 248, 251, 252, 254, 255, 257, 259, 261, 263, 265, 266, 268, 270, 271, 273, 275, 277, 279, 280, 282, 284, 286, 288, 289, 291, 294, 295, 297, 300, 303, 306, 309, 311, 314, 316
\glsxtrfullformat 207, 221, 223, 224, 226, 228, 231, 233, 235, 237, 240, 243, 245, 247, 248, 251–253, 255, 257, 259, 261, 262, 265–267, 269, 271, 273, 275, 277, 279, 280, 282, 284, 286, 288, 289, 291, 294, 295, 297, 300, 303, 306, 309, 311, 314, 316
\GLSxtrfullpl 21, 330, 331
\Glsxtrfullpl 20, 21, 331
\glsxtrfullpl 20, 21, 330
\Glsxtrfullplformat 207, 221, 223, 224, 226, 229, 231, 233, 235, 237, 240, 243, 245, 247, 248, 251, 252, 254, 255, 258, 259, 261, 263, 265, 267, 268, 270, 272, 273, 275, 277, 279, 280, 283, 284, 286, 288, 289, 291, 294, 295, 297, 300, 303, 306, 309, 311, 314, 317
\glsxtrfullplformat 221, 223, 224, 226, 228, 231, 233, 235, 237, 240, 243, 245, 247, 248, 251–253, 255, 257, 259, 261, 262, 265, 266, 268, 269, 271, 273, 275, 277, 279, 280, 282, 284, 286, 288, 289, 291, 294, 295, 297, 300, 303, 306, 309, 311, 314, 316
\glsxtrfullsep 206, 207, 226–229, 231–238, 240, 242–250, 252, 254, 256–266, 268, 270–278, 280–282, 284–287, 290–292, 302–307, 310–313, 317
\glsxtrgenabbrvfmt 67
\glsxtrgeneralpuncIIrules 356
\glsxtrgeneralpuncIrules 356

\glsxtrgetgrouptitle 133, 423
\glsxtrgroupfield 151
\Glsxtrheadfirst 321
\glsxtrheadfirst 321
\Glsxtrheadfirstplural 321
\glsxtrheadfirstplural 321
\Glsxtrheadfull 321
\glsxtrheadfull 321
\Glsxtrheadfullpl 321
\glsxtrheadfullpl 321
\Glsxtrheadlong 321
\glsxtrheadlong 321
\Glsxtrheadlongpl 321
\glsxtrheadlongpl 321
\Glsxtrheadname 321
\glsxtrheadname 145, 321
\Glsxtrheadplural 321
\glsxtrheadplural 321
\Glsxtrheadshort 321
\glsxtrheadshort 321
\Glsxtrheadshortpl 321
\glsxtrheadshortpl 321
\Glsxtrheadtext 321
\glsxtrheadtext 321
\glsxtrhiernamesep 50, 51
\glsxtrhyperlink 25, 26, 94
\glsxtrhyphensuffix 303, 311
\glsxtridentifyglslike 153
\glsxtrifcounttrigger 110–112
\glsxtrifcustomdiscardperiod 198
\glsxtrifemptyglossary 133, 139, 147
\GlsXtrIfFieldCmpNum 36
\GlsXtrIfFieldEqNum 146
\GlsXtrIfFieldNonZero 339
\glsxtrifhasfield 39, 50, 51, 91, 340, 419
\glsxtrifhyphenstart 302, 304, 307, 310, 312, 313
\glsxtrifindexing 92
\glsxtrifinmark 74, 101–104, 320, 321
\glsxtrifnextpunc 200, 201
\glsxtrifperiod 198–200
\glsxtrifrecordtrigger 157–159
\glsxtrifwasfirstuse 75–78, 83–87, 89, 122, 198, 199, 209, 212–219, 232, 233, 255, 269, 290, 291, 294, 296–298, 308, 310, 313, 315
\glsxtrinclinkcounter 161
\glsxtrindexaliased 91
\glsxtrindexseealso 54, 55
\glsxtrinithyperoutside 72
\glsxtrinitwrgloss 72, 156
\glsxtrinitwrglossbeforefalse 69, 70
\glsxtrinitwrglossbeforetrue 69, 70
\Glsxtrinlinefullformat 207, 209, 223, 224, 232, 233, 235, 237, 238, 240, 247–250, 252, 254, 256, 261–263, 265, 266, 268, 270, 278, 280–282, 284, 286, 287, 290, 291, 295, 298, 306, 309, 314, 317, 337
\glsxtrinlinefullformat 207, 209, 210, 223, 224, 231, 233, 234, 236, 238, 240, 246, 248–250, 252, 254, 256, 260, 262–264, 266, 268, 270, 278, 279, 281, 282, 284, 285, 287, 289, 291, 295, 298, 305, 309, 314, 317, 336
\Glsxtrinlinefullplformat 208, 211, 223, 224, 232, 233, 235, 237, 238, 240, 247–250, 252, 254, 256, 261–263, 265, 266, 268, 270, 278, 280–282, 284, 286, 287, 290, 292, 295, 298, 306, 309, 314, 317, 337
\glsxtrinlinefullplformat 207, 210, 211, 223, 224, 231, 233, 234, 236, 238, 240, 246, 248–250, 252, 254, 256, 261–264, 266, 268, 270, 278, 280–282, 284, 285, 287, 290, 291, 295, 298, 305, 309, 314, 317, 337
\glsxtrinsertinsidefalse 225
\GlsXtrInternalLocationHyperlink 26, 135
\glsxtrLatinA 358–363
\glsxtrLatinAELigature 360, 362, 363
\glsxtrLatinE 358–363
\glsxtrLatinEszettSs 359–361, 363
\glsxtrLatinEszettSz 360, 362
\glsxtrLatinEth 359–362
\glsxtrLatinH 358–363
\glsxtrLatinI 358–363
\glsxtrLatinInsularG 362
\glsxtrLatinK 358–363
\glsxtrLatinL 358–363
\glsxtrLatinM 358–363
\glsxtrLatinN 358–363
\glsxtrLatinO 358–363
\glsxtrLatinOELigature 360, 362, 363
\glsxtrLatinP 358–363
\glsxtrLatinS 358–363
\glsxtrLatinT 358–363
\glsxtrLatinThorn 362
\glsxtrLatinX 358–363
\GlsXtrLocationField 151

\glsxtrlocationhyperlink 135
\glsxtrlocrangefmt 134
\GLSxtrlong 21, 328
\Glsxtrlong 20, 21, 329
\glsxtrlong 20, 21, 328
\glsxtrlonghyphen 309
\glsxtrlonghyphennoshort 305, 306
\glsxtrlonghyphenshort 303
\glsxtrlongnoshortdescname 239, 286, 305
\glsxtrlongnoshortname 241, 249, 264, 281, 283, 306
\GLSxtrlongpl 21, 328, 329
\Glsxtrlongpl 20, 21, 329
\glsxtrlongpl 20, 21, 328
\glsxtrlongshortdescname 227, 244, 258, 272, 274, 304, 309
\glsxtrlongshortdescsort 227, 244, 258, 272, 274, 299, 304, 309
\glsxtrlongshortname 226, 242, 257, 271, 272, 293, 294, 303, 308
\glsxtrlongshortuserdescname 296, 299
\glsxtrmarkhook 319
\glsxtrMathItalicAlpha 367, 368, 371, 372
\glsxtrMathItalicBeta 367, 368, 371, 372
\glsxtrMathItalicChi 368, 372, 373
\glsxtrMathItalicDelta 367, 368, 371, 372
\glsxtrMathItalicEpsilon 367, 368, 371, 372
\glsxtrMathItalicEta 367, 368, 371, 372
\glsxtrMathItalicGamma 367, 368, 371, 372
\glsxtrMathItalicIota 367, 368, 371, 373
\glsxtrMathItalicKappa 368, 372, 373
\glsxtrMathItalicLambda 368, 372, 373
\glsxtrMathItalicMu 368, 372, 373
\glsxtrMathItalicNu 368, 372, 373
\glsxtrMathItalicOmega 368, 372, 373
\glsxtrMathItalicOmicron 368, 372, 373
\glsxtrMathItalicPhi 368, 372, 373
\glsxtrMathItalicPi 368, 372, 373
\glsxtrMathItalicPsi 368, 372, 373
\glsxtrMathItalicRho 368, 372, 373
\glsxtrMathItalicSigma 368, 372, 373
\glsxtrMathItalicTau 368, 372, 373
\glsxtrMathItalicTheta 367, 368, 371, 372
\glsxtrMathItalicUpsilon 368, 372, 373
\glsxtrMathItalicXi 368, 372, 373
\glsxtrMathItalicZeta 367, 368, 371, 372
\glsxtrmultisupplocation 341
\glsxtrnamereflink 342
\glsxtrnewabbrevpresetkeyhook 205
\glsxtrnewnumber 22
\glsxtrnewsymbol 22
\glsxtrNoGlossaryWarning 15, 24, 136
\GlsXtrNoGlsWarningAutoMake 140
\GlsXtrNoGlsWarningBuildInfo 140
\GlsXtrNoGlsWarningCheckFile 140
\GlsXtrNoGlsWarningEmptyMain 139, 140
\GlsXtrNoGlsWarningEmptyNotMain 140
\GlsXtrNoGlsWarningEmptyStart 139
\GlsXtrNoGlsWarningHead 139
\GlsXtrNoGlsWarningMisMatch 140
\GlsXtrNoGlsWarningNoOut 140
\GlsXtrNoGlsWarningTail 140, 141
\glsxtrnopostpunc 127
\glsxtronlydescname 317
\glsxtronlydescsort 317
\glsxtronlyname 316
\glsxtronlysuffix 316
\glsxtrorg@ifKV@glslink@hyper 67, 68
\glsxtrorglong 204, 227, 305
\glsxtrorgshort 204, 227
\GLSxtrp 101
\Glsxtrp 100
\glsxtrp 100, 102
\glsxtrparen 198, 199, 206, 207, 226–229, 231–238, 240, 242–252, 254, 256–266, 268, 270–282, 284–287, 290–292, 302–307, 310–313, 317
\Glsxtrpl 62
\glsxtrpl 62
\glsxtrpostdescription 128, 179, 196, 395
\glsxtrposthyphenlong 313, 315
\glsxtrposthyphenshort 308, 310
\glsxtrposthyphensubsequent 308, 310, 313, 315
\glsxtrpostlink 198
\glsxtrpostlinkendsentence 198
\glsxtrpostlinkhook 197
\glsxtrpostlocalreset 107, 109, 117
\glsxtrpostlocalunset 106, 109, 117
\glsxtrpostlongdescription 43
\glsxtrpostnamehook 184–186, 189
\GlsXtrPostNewAbbreviation 206, 223, 224, 226–230, 232, 234–239, 241–244, 246, 247, 250, 253, 255, 257–260, 262, 264, 267, 269, 271–274, 276–279, 281, 283, 285, 287–290, 293, 294, 296–301, 303–306, 308, 310–313, 315, 316, 318
\glsxtrpostreset 106, 109, 117

\glsxtrpostunset 105, 109, 117
\glsxtrprelocation 385, 387, 389, 391, 393, 396, 419
\glsxtrprotectlinks 94–96
\GlsXtrRecordCounter 13
\glsxtrrecordtriggervalue 156
\GlsXtrRecordWarning 141
\glsxtrregularfont 67, 75, 76
\glsxtrresourcecount 142
\glsxtrresourcefile 142
\glsxtrresourceinit 141
\glsxtrrestoremarkhook 319
\glsxtrrestorepostpunc 128
\glsxtrscfont 242
\glsxtrscsuffix 243, 245, 246, 248, 250, 251, 253, 255
\GlsXtrSetActualChar 193
\glsxtrsetaliasnoindex 15, 16, 91
\GlsXtrSetEncapChar 193
\GlsXtrSetEscChar 193
\glsxtrsetfieldifexists 38
\GlsXtrSetLevelChar 193
\glsxtrsetpopts 99
\glsxtrsetupfulldefs 209–211, 233, 255, 269, 291
\GLSxtrshort 21, 103, 104, 322, 323
\Glsxtrshort 20, 21, 323
\glsxtrshort 20, 21, 322
\glsxtrshortdescname 236, 247, 262, 279
\glsxtrshorthyphen 314
\glsxtrshorthyphenlong 311
\glsxtrshortlongdescname 229, 245, 259, 275, 277, 312, 315
\glsxtrshortlongdescsort 229, 245, 260, 275, 277, 301, 312, 315
\glsxtrshortlongname 228, 244, 258, 274, 276, 296, 300, 311, 313
\glsxtrshortlonguserdescname 298, 301
\glsxtrshortnolongname 234, 246, 260, 278
\GLSxtrshortpl 21, 322, 323
\Glsxtrshortpl 20, 21, 323
\glsxtrshortpl 20, 21, 322, 323
\glsxtrsmfont 256
\glsxtrsmsuffix 257, 259, 260, 262, 264, 265, 267, 269
\GlsXtrStandaloneEntryName 145
\GlsXtrStandaloneEntryOther 146
\GlsXtrStandaloneGlossaryType 145, 146
\GlsXtrStandaloneSubEntryItem 145, 146
\Glsxtrsubsequentfmt 220, 223, 239, 250, 252, 264, 266, 282, 283, 285, 287, 305, 308, 314
\glsxtrsubsequentfmt 220, 223, 239, 250, 251, 264, 266, 282, 283, 285, 287, 305, 308, 314
\Glsxtrsubsequentplfmt 220, 223, 240, 250, 252, 264, 266, 282, 283, 285, 287, 305, 309, 314
\glsxtrsubsequentplfmt 220, 223, 239, 250, 251, 264, 266, 282, 283, 285, 287, 305, 308, 314
\glsxtrsupplocationurl 135, 136, 341, 343
\glsxtrtagfont 196
\Glsxtrtitlefirst 320, 321, 334
\glsxtrtitlefirst 320, 321, 334
\Glsxtrtitlefirstplural 320–322, 335
\glsxtrtitlefirstplural 320, 321, 335
\Glsxtrtitlefull 320–322, 337
\glsxtrtitlefull 320–322, 336
\Glsxtrtitlefullpl 320–322, 337
\glsxtrtitlefullpl 320–322, 337
\Glsxtrtitlelong 320–322, 335, 336
\glsxtrtitlelong 320–322, 335
\Glsxtrtitlelongpl 320–322, 336
\glsxtrtitlelongpl 320–322, 336
\Glsxtrtitlename 320, 321, 333
\glsxtrtitlename 320, 321, 332
\glsxtrtitleorpdforheading 27, 145, 146, 320, 321
\Glsxtrtitleplural 320, 321, 334
\glsxtrtitleplural 320, 321, 333, 334
\Glsxtrtitleshort 320, 321, 332
\glsxtrtitleshort 320, 321, 331
\Glsxtrtitleshortpl 320, 321, 332
\glsxtrtitleshortpl 320, 321, 331, 332
\Glsxtrtitletext 320, 321, 333
\glsxtrtitletext 320, 321, 333
\GlsXtrTotalRecordCount 155
\glsxtrtreetopindent 402, 410
\glsxtrundefaction 7, 15, 16, 31, 44, 45, 47, 48
\glsxtrundeftag 30, 130, 131
\GlsXtrUnknownDialectWarning 41
\glsxtrunsrtdo 149
\glsxtrUpAlpha 366, 367, 371, 372
\glsxtrUpBeta 366, 367, 371, 372
\glsxtrUpChi 367, 372, 373
\glsxtrUpDelta 366, 367, 371, 372
\glsxtrUpDigamma 366, 367, 371

\glsxtrUpEpsilon 366, 367, 371, 372
\glsxtrUpEta 366, 367, 371, 372
\glsxtrUpGamma 366, 367, 371, 372
\glsxtrUpIota 366, 367, 371, 373
\glsxtrUpKappa 366, 367, 372, 373
\glsxtrUpLambda 366, 367, 372, 373
\glsxtrUpMu 366, 367, 372, 373
\glsxtrUpNu 366, 367, 372, 373
\glsxtrUpOmega 367, 372, 373
\glsxtrUpOmicron 366, 367, 372, 373
\glsxtrUpPhi 367, 372, 373
\glsxtrUpPi 366, 367, 372, 373
\glsxtrUpPsi 367, 372, 373
\glsxtrUpRho 366, 367, 372, 373
\glsxtrUpSigma 366, 367, 372, 373
\glsxtrUpTau 367, 372, 373
\glsxtrUpTheta 366, 367, 371, 373
\glsxtrUpUpsilon 367, 372, 373
\glsxtrUpXi 366, 367, 372, 373
\glsxtrUpZeta 366, 367, 371, 372
\GlsXtrUseAbbrStyleFmts 227, 230, 236, 238, 239, 241, 242, 244, 246, 249, 258, 260, 263, 272, 274, 276, 278, 280, 285, 288, 296, 299, 301, 304, 305, 307, 310, 312, 315, 318
\GlsXtrUseAbbrStyleSetup 235, 237, 238, 241, 242, 249, 251, 263, 265, 280, 284, 285, 288
\glsxtruserfield 292
\glsxtruserparen 293–301
\glsxtrusersuffix 293, 295, 297, 300
\glsxtruseseealsoformat 52
\glsxtruseseeformat 49
\GlsXtrWarnDeprecatedAbbrStyle 202, 224
\GlsXtrWarning 60, 61
\glsxtrword 203
\glsxtrwordsep 203, 302, 304, 307, 310, 312, 313
\glsxtrwrglossmark 27

H

\hangindent 398, 400, 401, 411–413, 417, 418
\hbox 384
\hfill 384
\href 94
\hsize 63, 64
\hss 384
\hyperlink 16, 95, 342
\hyperpage 189
\hyperref 94, 135, 339, 340, 344
hyperref package 95, 189, 318, 331, 342

I

\if 59
\if@display 12
\if@glsxtr@autoseeindex 28, 49, 54
\if@glsxtr@equations 9, 72
\if@glsxtr@floats 18
\if@glsxtr@format@override 190
\if@glsxtrdocdefrestricted 58
\if@glsxtrindexcrossrefs 17, 55
\ifblank 32, 60, 61, 123
\ifbool 31, 43
\ifcase 7, 15, 22, 24, 26, 58, 70, 128, 396, 422
\ifcsdef 31, 42, 44–47, 70, 73, 87, 88, 100–104, 114, 126, 132, 133, 146, 150, 153–155, 160, 181–184, 186–188, 198, 202, 219, 223, 387–394
\ifcsstring 31, 177, 222
\ifcsundef 37, 40, 42, 44–47, 58, 63, 65, 96, 108, 114–118, 132, 136, 150, 161, 177, 222, 224, 225, 384, 402, 403, 411, 424
\ifcsvoid 55, 176
\ifdef 15, 22, 28, 34, 37, 39, 41, 48, 52, 53, 56, 63, 64, 90, 93, 95, 101–103, 126, 130, 131, 141, 143, 153, 179, 180, 193, 196, 292, 320, 331–337, 339, 344–347, 382, 384–387, 395–401, 412–418, 420, 423, 424
\ifdefempty 7–9, 36, 40, 41, 44–46, 49, 52, 73, 75, 108, 119, 120, 123, 126, 134, 148, 151, 155, 156, 170–172, 195, 203, 219, 347, 421
\ifdefequal 39, 57, 93, 140, 151, 152, 187
\ifdefstring 6, 26, 39, 190, 195, 420
\ifdefvoid 49, 53–56, 94, 114, 131, 135, 151, 152, 343
\ifdim 63, 64, 122, 402–410, 427, 428
\IfFileExists 24, 136, 140, 141, 144, 382, 383
\ifglossaryexists 48
\ifglsacronym 20, 140
\ifglsacrshortcuts 22
\ifglsautomake 126, 140, 144
\ifglsentrycounter 41, 42
\ifglsentryexists 9, 31, 47, 48, 60, 61, 64, 75, 151, 177, 196, 197, 348
\ifglsfieldeq 175
\ifglshasfield 33, 34, 292
\ifglshaslong 113, 159, 160
\ifglshasparent 145, 146, 148, 151, 404–406

\ifglshasshort 50, 51, 67, 75
\ifglshassymbol 199, 398, 399, 401
\ifglsindexonlyfirst 92
\ifGlsLongExtraUseTabular 429, 430, 432, 433, 435, 436, 438, 440, 441, 443, 444, 446
\ifglsnogroupskip 385, 387–395, 397, 398, 400, 412, 420, 430, 431, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447
\ifglsnonumberlist 66
\ifglsnopostdot 18, 19, 128
\ifglssanitizesort 125
\ifglssubentrycounter 41, 42
\ifglsused 55, 56, 89, 92, 110, 118, 122, 219, 404, 405, 407, 408, 410
\ifglsxindy 136, 138
\ifglsxtr@hyperoutside 73, 74
\ifglsxtrinitwrglossbefore 70, 73, 74, 156
\ifglsxtrinsertinside 212–219, 221, 222, 226, 228, 229, 231–240, 243, 245–259, 261–273, 275, 277–292, 294, 295, 297, 298, 300, 302, 304, 307–310, 313, 314, 316, 317
\ifHy@hyperindex 189
\ifinlist 106
\ifinlistcs 35, 58
\ifinner 27
\ifKV@glslink@hyper 67, 68, 72–74
\ifKV@glslink@local 69, 156
\ifKV@glslink@noindex 8, 9, 13, 33, 91
\ifmmode 12, 27
\ifnum 16, 17, 36, 57, 110, 118, 119, 131, 142, 156, 344–346, 398, 400, 411, 412
\ifst@rred 12
\ifstrempty 146, 153, 161, 342, 344–348
\ifstrequal 19, 24, 71, 342
\ifthenelse 139, 140, 198
\IfTrackedDialectHasMapping 40
\IfTrackedLanguageFileExists 338
\ifundef 16, 26, 36, 40, 123, 195, 196, 342, 381, 427
\ifx 8–11, 14, 29, 30, 52, 63, 64, 73, 123, 126, 127, 129, 134, 135, 141, 143, 144, 190, 191, 193, 201, 203, 205, 301, 302, 343, 418
\immediate 57, 109, 110, 118, 136, 137, 144
\index 190
\indexspace 385, 397–401, 412–418, 420, 423
\input 338
\inputencodingname 143
\InputIfFileExists 57
\istfilename 123
\item 138, 139, 384–387, 396, 397, 413, 414

J

\jobname 57, 136, 138–140, 142, 144

K

\key@ifundefined 13, 14, 31, 32, 87, 148, 150
\KV@glslink@hyperfalse 76, 89, 90, 95, 96
\KV@glslink@hypertrue 95
\KV@glslink@noindexfalse 75, 90
\KV@glslink@noindextrue 90, 96

L

\L 363, 366
\l 366
\label 26
\LaTeX 138, 139
\leaders 384
\leavevmode 44, 72
\let 5, 7–11, 13, 15, 16, 18, 20–22, 27–29, 33, 35, 39, 40, 43, 57, 59, 62, 63, 65–69, 72–93, 95, 96, 100, 105, 106, 108, 109, 117–124, 126–133, 141–144, 148, 149, 151, 152, 156, 161, 171, 182, 183, 185–192, 195, 196, 201–205, 209–219, 221–223, 233, 255, 269, 291, 318–322, 339, 342, 343, 382, 396, 401, 403, 414, 421–424
\letabbreviationstyle 232, 234, 235, 237, 240, 241, 247, 249, 261, 263, 279, 280
\letcs 31, 36, 49, 52, 56, 70, 73, 87, 130–132, 150, 151, 181–189, 424
\levelchar 193
\linewidth 428
\listadd 114
\listbreak 195
\listcsadd 34
\listcseadd 34, 115
\listcsgadd 34, 58
\listcsxadd 34, 115
\listxadd 105, 106
\loadglsentries 59, 137
\long 43

M

\m@ne 339
\MakeAcronymsAbbreviations 121
\makeatletter 57, 136, 141, 192
\makeatother 192
\makebox 384, 411, 412

\makefirstuc 196
makeglossaries 129
\makeglossaries 122, 137–140, 144
\makeglossary 123
makeindex **448**
makeindex 16, 122
\makenoidxglossaries 138
\MakeTextUppercase 321
\MakeUppercase 320, 321
\marginpar 27
\markboth 319
\markright 319
\mathit 351
\mathrm 350–352
\maxdimen 63, 64
\mbox 386, 387, 411
\medskip 139, 140, 150
\MessageBreak 58, 59, 62, 110, 119, 123, 125–127, 140, 141, 222
mfirstuc package 195
\mfirstucMakeUppercase 37, 76–87, 89, 97–99, 101, 103, 104, 112, 113, 121, 160, 162–169, 172–175, 184, 185, 189, 210, 211, 213, 215, 217, 219–221
\mfu@checkword@arg 195, 196
\mfu@checkword@do 196
\midrule 427, 430, 432, 433, 435, 436, 438, 439, 441, 442, 444, 446

N

\NeedsTeXFormat 5, 339, 383, 419, 425
\new@atom@glossaryentry 57
\new@command 339
\new@glossaryentry 59, 125
\new@ifnextchar 33, 88, 89, 112, 113, 153, 157–159, 200, 208–219, 348, 349
\newabbr 21
\newabbreviation 21
\newabbreviationhook 206
\newabbreviationstyle 225, 227–230, 232, 234–239, 241–247, 249, 251, 253, 254, 257–261, 263–265, 267, 268, 271, 272, 274–281, 283–286, 288, 290, 293, 294, 296, 298, 299, 301, 303, 305, 306, 308, 309, 311, 313, 315–317
\newacronym 120, 121
\newacronymhook 120
\newacronymstyle 121, 122
\newcommand 5–37, 39, 41–52, 54–56, 59–62, 64–67, 69–71, 74–76, 87–93, 95, 96, 99, 101–108, 110, 112–116, 118, 119, 121, 122, 126–132, 134–142, 144–170, 172–181, 187–204, 206–219, 221–225, 227–230, 234, 236, 239, 241, 242, 256, 257, 270, 271, 292, 293, 296, 298, 302, 304, 307, 310, 312, 313, 315–317, 319–344, 346–349, 352–381, 383, 385, 395–399, 401–403, 410, 411, 419, 420, 423–430, 432–436, 438, 439, 441–446
\newcount 142, 152
\newcounter 26, 160
\newentry 22
\newglossary 20, 123
\newglossaryentry 22, 57, 59, 108, 116, 120, 179, 180, 206
\newglossaryentry options
- access 171
- alias 17, 49, 53–56
- desc 166, 174
- descplural 166, 167, 174
- first 94, 163, 164, 173, 225, 326, 327, 334, 448
- firstaccess 172
- firstplural 164, 173, 225, 327, 334, 448
- group 150, 151
- loclist 34
- long 168, 175, 335
- longplural 169, 175, 336
- name 49, 50, 162, 172, 190, 323, 324, 332
- plural 163, 172, 173, 225, 325, 326, 333
- see 17, 28, 49, 53, 56, 59, 123
- seealso 17, 49, 51, 53, 55, 56, 459
- short 167, 174, 202
- shortaccess 170–172
- shortaccessplural 170
- shortplural 168, 175, 202
- symbol 165, 173
- symbolplural 165, 166, 173, 174
- text 94, 162, 163, 172, 225, 227, 324, 325, 333
- textaccess 171
- user2 41

\newglossarystyle 421, 429, 430, 432, 433, 435, 436, 438, 439, 441, 443, 444, 446
\newif 70, 189, 225, 429
\newlength 401, 402
\newnum 22
\newrobustcmd 32, 34, 35, 38, 39, 52, 53, 57, 71, 75, 88, 89, 99–101,

112, 113, 127, 132, 145, 146, 149, 150, 153, 154, 157–159, 188, 195, 196, 208–219, 301, 320, 322–331, 348–350, 403–410
\newsym 22
\newterm 179
\newtoks 202
\newwrite 57, 123
\nobreak 386, 387, 397, 399–401, 413–416
\NoCaseChange 101–104, 146, 322–331
\noexpand 11, 13, 24, 52, 54, 55, 57, 71, 120, 136, 137, 141, 148, 149, 151, 160, 191, 192, 206, 339, 348, 383, 422, 429, 431–438, 440–446
\nofiles 139
\noindent 140, 399–401, 414–416, 418
\nopagebreak 397, 399–401, 412–419, 423
\nopostdesc 44, 60, 61, 127, 179
\ns@GLSxtrfull 210
\ns@Glsxtrfull 209
\ns@glsxtrfull 208
\ns@GLSxtrfullpl 211
\ns@Glsxtrfullpl 211
\ns@glsxtrfullpl 210
\ns@GLSxtrlong 215
\ns@Glsxtrlong 214
\ns@glsxtrlong 214
\ns@GLSxtrlongpl 219
\ns@Glsxtrlongpl 218
\ns@glsxtrlongpl 217
\ns@GLSxtrshort 213
\ns@Glsxtrshort 212
\ns@glsxtrshort 212
\ns@GLSxtrshortpl 217
\ns@Glsxtrshortpl 216
\ns@glsxtrshortpl 215
\null 24
\number 57, 115, 116, 118, 141, 153
\numexpr 115, 118

O

\O 363, 366
\o 366
\openout 57
\or 7, 16, 22, 23, 26, 27, 59, 70, 396, 422
\org@glossaryentrynumbers 64, 127
\org@glossarytitle 126, 127
\org@ifKV@glslink@hyper 72, 74

P

\p@gls@hyp@opt 92
package options:
  abbreviations 20
  accsupp 23, 162
  acronym 20
  automake 126, 138, 144
    true 144
  autoseeindex 28
    false 28
  counter
    wrglossary 25
  debug
    showtargets 95
  docdef 16, 57, 58, 108, 116
    atom 57
    false 57, 58
    restricted 17
    true 56, 59
  docdefs
    restricted 58
  equations 9, 72
  indexcounter 339
  nonumberlist 64
  nopostdot 18
    false 18
  numbers 21
  postdot 18
  record 7, 14, 15, 57, 67, 122, 140, 141, 209–219, 457
    alsoindex 8, 10
    nameref 12, 29, 30, 141, 144
    only 8, 139
  shortcuts 22
    ac 22
    all 22
    false 22
    none 22
    true 22
  sort
    use 75
  style 25
  stylemods 25
  symbols 21, 180
  undefaction 47
    error 6
    warn 6
  xindy 52, 53
\PackageError 6, 13, 24, 28, 57–59, 62, 63, 70, 88, 89, 91, 100, 101,

108, 109, 116, 118, 119, 121, 123, 125, 133, 144, 148, 150, 153, 198, 222–225, 384
\PackageWarning 18
\PackageWarningNoLine 18
\pagelistname 430, 433, 436, 439, 442, 445
\pageref 25, 41, 42
\par 139, 140, 386, 387, 396, 398–401, 411–418, 420
\parindent 396, 398, 400–402, 411–418, 421
\parskip 396, 398, 400, 414–416, 421
\PassOptionsToPackage 5, 24
\pdfbookmark 420
\preglossarypreamble 42
\preto 90, 347
\print@noop@unsrtglossaryunit 13, 16
\print@op@unsrtglossaryunit 15, 16
\printabbreviations 20
\printglossaries 124, 138
\printglossary 20, 124, 138–140
\printglossary options
  nonumberlist 66
  type 126
\printnoidxglossaries 138
\printnoidxglossary 124, 138
\printnumbers 22, 180
\printsymbols 22, 180
\printunsrtglossary 139, 141, 147
\printunsrtglossaryentryprocesshook 148, 149
\printunsrtglossaryhandler 149
\printunsrtglossarypredoglossary 148
\printunsrtglossaryskipentry 148
\printunsrtglossaryunit 15, 16, 150
\printunsrtglossaryunitsetup 149
\ProcessOptions 384
\ProcessOptionsX 27
\protect 101–104, 172–175, 203, 206, 207, 225–232, 234–236, 238–255, 257–269, 271–279, 281–291, 293, 294, 296–301, 303–306, 308–313, 315–318, 322–331
\protected@csedef 37, 38, 132, 402, 403
\protected@csxdef 38, 132, 402, 403
\protected@edef 29, 39, 63, 71, 93, 120, 131, 132, 149–151, 190, 206, 343
\protected@write 11–13, 58, 66, 93, 123, 124, 141, 143, 145, 152, 153, 348, 424
\providecommand 20, 21, 32, 52, 66, 89, 90, 93, 109, 118, 123, 136, 137, 143, 152, 339, 348, 350–352, 384, 419
\ProvidesFile 338
\ProvidesPackage 5, 339, 383, 419, 425

Q

\quotechar 193

R

\raggedright 389, 390, 393, 394, 421, 426, 427
\refstepcounter 26
\relax 7, 15, 17, 21, 22, 24, 26, 28, 29, 33, 36, 57, 59, 62–65, 69, 70, 72, 73, 92, 100, 109, 110, 118, 123, 124, 126, 127, 130, 131, 134, 141–144, 151, 152, 156, 191–193, 196, 198, 203, 205, 301, 302, 339, 344–346, 396, 398, 400, 403–413, 417, 418, 421–424
relsize package 256
\renewcommand 6, 7, 15–20, 22–29, 43–48, 50, 57–59, 62, 64–67, 69, 71, 87, 89–91, 93–95, 99, 106–109, 113, 116–129, 131–137, 149, 150, 155, 179, 181–186, 194–197, 207, 208, 223, 224, 226–301, 303–306, 308–319, 347, 381, 384–401, 411–418, 421–423, 429–447
\renewenvironment 385, 387–394, 396, 398, 400, 411, 414–416, 418, 421, 429, 431, 432, 434–438, 440, 441, 443–446
\renewglossarystyle 384–394, 396–400, 411–418
\renewrobustcmd 74, 95
\RequireGlossariesExtraLang 338, 382
\RequirePackage 5, 15, 23, 24, 27, 383, 419, 425
\reserved@a 200, 201
\reserved@b 200, 201
\reserved@d 201
\RestoreAcronyms 121
\rGLS 155
\rGls 155
\rgls 155
\rGLSformat 158
\rGlsformat 158
\rglsformat 157, 160
\rGLSpl 155
\rGlspl 155
\rglspl 155
\rGLSplformat 159
\rGlsplformat 158
\rglsplformat 157, 160
\romannumeral 402, 403, 411

S

\s@@glsxtrfmt 33
\s@gls@hyp@opt 92
\s@glsxtr@enabletagging 194, 195
\s@glsxtrfmt 32
\s@glsxtrifhasfield 35
\s@GlsXtrStartUnsetBuffering 105
\s@GlsXtrStopUnsetBuffering 106
\s@printunsrtglossary 147, 149
\seealsoname 52, 53
\seename 49
\setabbreviationstyle 121, 227, 235
\setacronymstyle 121, 122
\setentrycounter 134, 341, 343
\SetGenericNewAcronym 122
\setglossarystyle 25, 127, 384, 386, 387, 397, 399, 400, 412–418, 421
\setkeys 9, 25, 29, 33, 72, 74, 91, 120, 127, 156, 204, 205
\setlength 63, 64, 396, 398, 400, 401, 412, 414–416, 421, 428, 429
\settowidth 122, 402–411, 427, 428
\setupglossaries 5, 29
\sfcode 18, 19, 198, 395
\small 27, 51
soul package 105
\space 6, 13, 52, 58, 60, 62, 91, 108–110, 116, 118, 119, 121–125, 127, 137, 139, 140, 144, 148, 150, 153, 198, 199, 203, 207, 227, 384, 385, 395, 398, 399, 401, 402, 411, 419
\spacefactor 18, 19, 198, 205, 395
\stepcounter 161
\string 6, 11–13, 30, 52, 53, 58–60, 62, 66, 73, 88, 91, 93, 100, 101, 108–110, 116, 118, 119, 121, 123–127, 136–141, 143–145, 148–150, 152, 153, 183, 185, 186, 188, 190, 198, 339, 348, 352–381, 424
\strut 384–395, 400, 426
\subglossentry 127, 151, 384–394, 396, 398, 400, 412, 422, 430, 431, 433, 434, 436, 437, 439, 440, 442, 444, 445, 447
\subitem 396
\subsubitem 396
\symbolname 428, 435, 436, 438, 439, 441, 442, 444, 446

T

\tabcolsep 428, 429
\tablehead 391–394
\tabletail 391–394
\tabularnewline 387–395, 427, 430–447
\TeX 138
\texorpdfstring 34, 37, 101–103, 320, 331–337
\textbf 395, 427
textcase package 318
\textsc 242
\textsmaller 256
\texttt 27, 137–140
\the 120, 135, 142, 193, 194, 206, 225–232, 234, 236–239, 241–247, 249, 250, 253–255, 257–260, 262, 264, 267, 269, 271–279, 281, 283, 285–291, 293, 294, 296–301, 303–306, 308–313, 315–318, 343
\theglsentrycounter 8, 10–12, 73, 75, 156
\theH 30
\theHglsentrycounter 8, 10–12, 73, 75, 156
\theindex 189
\thewrglossary 26
\this@dialect 338, 382
\thisgrptitle 423
\toks@ 135, 193, 194, 343
\toprule 427, 430, 432, 433, 435, 436, 438, 439, 441, 442, 444, 445
\TrackedDialectClosestSubMatch 40
tracklang package 39, 40, 143, 352
\TrackLangGetDefaultScript 381
\TrackLangIfHasDefaultScript 381
\TrackLangRequireDialectPrefix 382
\triangleright 51

U

\u 339
\undef 16, 57, 195
\underline 196
\unskip 44, 56, 384
upgreek package 351
\usepackage 139, 140

W

\warn@nomakeglossaries 124
\warn@noprintglossary 124, 127
wrglossary (counter) 25, 26
\write 52, 109, 110, 118, 123, 136, 137, 144

X

\x 135, 160, 343, 348
\xcapitalisewords 181
\xdef 127, 149, 339

`\xifinlist` 114
`\xifinlistcs` 35
xindy **448**
`xindy` 16, 122
xkeyval package 5
`\XKV@checkchoice` 66
`\XKV@plfalse` 66
`\XKV@resa` 66
`\XKV@sttrue` 66