glossaries-extra.sty v1.48: documented code

Nicola L.C. Talbot

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

2021-11-22

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 . . . 34
- 1.3 Modifications to Commands Provided by glossaries . . . 52
  - 1.3.1 Existence Checks . . . 57
  - 1.3.2 Document Definitions . . . 68
  - 1.3.3 Existing Glossary Style Modifications . . . 74
  - 1.3.4 Entry Formatting, Hyperlinks and Indexing . . . 78
  - 1.3.5 Entry Counting . . . 119
  - 1.3.6 Acronym Modifications . . . 135
  - 1.3.7 Indexing and Displaying Glossaries . . . 139
- 1.4 Link Counting . . . 182
- 1.5 Integration with glossaries-accsupp . . . 184
- 1.6 Categories . . . 201
- 1.7 Abbreviations . . . 230
  - 1.7.1 Abbreviation Styles Setup . . . 250
  - 1.7.2 Predefined Styles (Default Font) . . . 254
  - 1.7.3 Predefined Styles (Small Capitals) . . . 273
  - 1.7.4 Predefined Styles (Fake Small Capitals) . . . 290
  - 1.7.5 Predefined Styles (Emphasized) . . . 307
  - 1.7.6 Predefined Styles (User Parentheses Hook) . . . 331
  - 1.7.7 Predefined Styles (Hyphen) . . . 345
  - 1.7.8 Predefined Styles (No Short on First Use) . . . 360
- 1.8 Using Entries in Headings . . . 365
- 1.9 Multi (Combined/Compound) Entries . . . 388
- 1.10 Multi-Lingual Support . . . 438
- 1.11 glossaries-extra-bib2gls.sty . . . 439

2 Style Adjustments (glossaries-extra-stylemods.sty) 488
- 2.1 Package Initialisation . . . 488
- 2.2 List-Like Styles . . . 489
- 2.3 Longtable Styles . . . 493
- 2.4 Long Ragged Styles . . . 495
- 2.5 Supertabular Styles . . . 497
- 2.6 Super Ragged Styles . . . 499
- 2.7 Inline Style . . . 501
- 2.8 Tree Styles . . . 501

2.9 Multicolumn Styles 521

3 bookindex style (glossary-bookindex.sty) 528
3.1 Package Initialisation and Options 528

4 longextra styles (glossary-longextra.sty) 535
4.1 Package Initialisation and Options 535

5 topic styles (glossary-topic.sty) 558
5.1 Package Initialisation and Options 558

Glossary 564

Change History 565

Index 595

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra}[2021/11/22 v1.48 (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]{%
    \protected@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

     \protected@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]{%
```

Save the label in case it's needed. This needs to be outside the existence check to allow the post-link hook to reference it.

```
 \protected@edef\@gls@label{\glsdetoklabel{#2}}%
 \let\glslabel\@gls@label
 \ifglsentryexists{#2}{}%
 {%
   \@@glsxtrwrglossmark
   \begingroup
     \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 or record=hybrid 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
```

Protect against non-expandable commands occurring in the location.

```
    \protected@edef\@glsxtr@theentrycounter{\theglsentrycounter}%
    \protected@edef\@glsxtr@theHentrycounter{\theHglsentrycounter}%
    \@onelevel@sanitize\@glsxtr@theentrycounter
    \@onelevel@sanitize\@glsxtr@theHentrycounter
    \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
      {\@glsxtr@theentrycounter}{\@glsxtr@theHentrycounter}%
    }%
    \@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}}%
```

```
245      {\theHglsentrycounter}}%
246   }%
247 }
```

r@recordcounter

```
248 \newcommand*{\@@glsxtr@recordcounter}{%
249   \@glsxtr@noop@recordcounter
250 }
```

p@recordcounter

```
251 \newcommand*{\@glsxtr@noop@recordcounter}[1]{%
252   \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
253    requires record=only or record=hybrid package option}{}%
254 }
```

p@recordcounter

```
255 \newcommand*{\@glsxtr@op@recordcounter}[1]{%
256   \protected@eappto\@glsxtr@counterrecordhook{\noexpand\@glsxtr@docounterrecord{#1}}%
257 }
```

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

```
258 \newcommand*{\@glsxtr@recordsee}[2]{%
259  \@@glsxtrwrglossmark
260  \def\@gls@xref{#2}%
261  \@onelevel@sanitize\@gls@xref
262  \protected@write\@auxout{}{\string\glsxtr@recordsee{#1}{\@gls@xref}}%
263 }
```

srtglossaryunit

```
264 \newcommand{\printunsrtglossaryunit}{%
265   \print@noop@unsrtglossaryunit
266 }
```

tr@setup@record Initialise.

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

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

```
268 \newcommand*{\glsxtr@indexonly@saveentrycounter}{%
269  \ifKV@glslink@noindex
270  \else
271    \glsxtr@saveentrycounter
272  \fi
273 }
```

addloclistfield

```
274 \newcommand*{\glsxtr@addloclistfield}{%
275  \key@ifundefined{glossentry}{loclist}%
276  {%
```

```
    \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` As from v1.46, the record=alsoindex is renamed to record=hybrid with record=alsoindex as a deprecated synonym to avoid confusion. The internal commands that include `alsoindex` in the name will remain unchanged to avoid breaking things, but this command will need to be redefined by record=hybrid.

```
\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}
```

id@noprintgloss Used by hybrid method if \printglossary isn't used.

```
\newcommand\@glsxtr@warn@hybrid@noprintgloss{%
  \ifdefstring{\@glo@types}{,}%
  {%
    \GlossariesExtraWarningNoLine{No glossaries have been defined}%
  }%
  {%
    \GlossariesExtraWarningNoLine{No \string\printglossary\space
      or \string\printglossaries\space
      found. ^^JYou have requested the hybrid setting
      record=\@glsxtr@record@setting\space which requires a
      combination of bib2gls (to fetch entries) and makeindex/xindy
      (to sort and collate the entries). If you only want to use
      bib2gls then change the option to record=only or record=nameref}%
  }%
}
```

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. As from v1.46, record=alsoindex is a deprecated synonym of record=hybrid to avoid confusion.

```
\define@choicekey{glossaries-extra.sty}{record}
 [\@glsxtr@record@setting\glsxtr@record@nr]%
 {off,only,alsoindex,nameref,hybrid}%
 [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`. Index in this sense refers to the indexing mechanism used with indexing applications such as makeindex and xindy, but this could be confused with recording locations so "alsoindex" is now deprecated in favour of "hybrid", which is more obvious.

```
     \def\glsxtr@setup@record{%
       \renewcommand*{\@glsxtr@record@setting@alsoindex}{alsoindex}%
```

```
      \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}}%
    {}%
  \or
```

Hybrid record (use bib2gls to fetch definitions) and index (use makeindex/xindy to sort and collate).

```
    \def\glsxtr@setup@record{%
      \renewcommand*{\@glsxtr@record@setting@alsoindex}{hybrid}%
      \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
    }%
  \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.

`\glsxtr@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}{}
```

glossaries-accsupp can't be loaded after glossaries-extra. glossaries-accsupp v4.29+ checks \@glsxtr@doaccsupp to determine if it's been loaded too late.

accsupp If accsupp, load glossaries-accsupp package.

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

tr@doloadprefix

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

prefix If prefix, load glossaries-prefix package.

```
\@glsxtr@declareoption{prefix}{%
 \renewcommand*{\@glsxtr@doloadprefix}{\RequirePackage{glossaries-prefix}}}
```

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}}%
        }%
        {%
```

```
693            \PackageError{glossaries-extra}%
694              {Glossaries style package 'glossary-\@glsxtr@tmp.sty'
695               doesn't exist (did you mean to use the 'style' key?)}%
696              {The list of values (#1) in the 'stylemods' key should
697               match the glossary-xxx.sty files provided with
698               glossaries.sty}%
699          }%
700        }%
701        \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
702     }
703   }%
704 }
```

glsxtr@do@style

```
705 \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.

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

Defer actual style change:

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

Set this as the default style:

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

Set this style:

```
709    \setglossarystyle{#1}%
710  }%
711 }
```

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.

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

cationHyperlink

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

The first two arguments are always control sequences.

```
713 \newcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
714   \glsxtrhyperlink{#1#2#3}{#3}%
715 }
```

cationhyperlink

```
716 \newcommand*{\@glsxtr@wrglossary@locationhyperlink}[3]{%
717   \pageref{wrglossary.#3}%
718 }
```

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}}
```

tr@doshowtarget

```
\newcommand\@glsxtr@doshowtarget[2]{#2}
```

debug Provide extra debug options.

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

showtargetouter

```
\newcommand*{\glsxtrshowtargetouter}{\glsshowtargetouter}
```

showtargetinner

```
\newcommand*{\glsxtrshowtargetinner}[1]{\glsshowtargetinner{#1}}
```

Debugging show targets.

rshowtargetleft

```
\newcommand{\@glsxtrshowtargetleft}[2]{\@glsshowtarget{#1}#2\@glsxtrshowtargetmark}%
```

showtargetright

```
\newcommand{\@glsxtrshowtargetright}[2]{\@glsxtrshowtargetmark#2\@glsshowtarget{#1}}%
```

rshowtargetmark

```
\newcommand{\@glsxtrshowtargetmark}{}%
```

showtargets Implements debug=showtargets and provides extra adjustments.

```
\define@choicekey{glossaries-extra.sty}{showtargets}
 [\@glsxtr@showtargetsval\@glsxtr@showtargetsnr]%
 {left,right,innerleft,innerright,annoteleft,annoteright}%
 {%
```

```
  \glsxtr@dooption{debug=showtargets}%
  \ifcase\@glsxtr@showtargetsnr\relax
    \def\@glsxtr@doshowtarget{\@glsxtrshowtargetleft}%
    \def\glsxtrshowtargetouter{\glsshowtargetouter}%
    \def\glsxtrshowtargetinner{\glsshowtargetinner}%
    \let\@glsxtrshowtargetmark\empty
  \or
    \def\@glsxtr@doshowtarget{\@glsxtrshowtargetright}%
    \def\glsxtrshowtargetouter{\glsshowtargetouter}%
    \def\glsxtrshowtargetinner{\glsshowtargetinner}%
    \let\@glsxtrshowtargetmark\empty
  \or
    \def\@glsxtr@doshowtarget{\@glsxtrshowtargetleft}%
    \def\glsxtrshowtargetouter{\glsxtrshowtargetinner}%
    \def\glsxtrshowtargetinner{\glsshowtargetinnersymleft}%
    \let\@glsxtrshowtargetmark\empty
  \or
    \def\@glsxtr@doshowtarget{\@glsxtrshowtargetright}%
    \def\glsxtrshowtargetouter{\glsxtrshowtargetinner}%
    \def\glsxtrshowtargetinner{\glsshowtargetinnersymright}%
    \let\@glsxtrshowtargetmark\empty
  \or
    \def\@glsxtr@doshowtarget{\@glsxtrshowtargetleft}%
    \def\glsxtrshowtargetouter{\glsxtrshowtargetinner}%
    \def\glsxtrshowtargetinner{\glsshowtargetinnersymleft}%
    \def\@glsxtrshowtargetmark{\@glsshowtargetmarkfmt\glsxtrshowtargetsymbolright}%
  \or
    \def\@glsxtr@doshowtarget{\@glsxtrshowtargetright}%
    \def\glsxtrshowtargetouter{\glsxtrshowtargetinner}%
    \def\glsxtrshowtargetinner{\glsshowtargetinnersymright}%
    \def\@glsxtrshowtargetmark{\@glsshowtargetmarkfmt\glsxtrshowtargetsymbolleft}%
  \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
```

Load the glossaries-prefix package if required.

```
\@glsxtr@doloadprefix
```

Redefine `\glspostdescription` if required.

```
\@glsxtr@defpostpunc
```

glsdoshowtarget Added to glossaries v4.50 so many not be defined. Need to redefine it so use \def.

```
\def\glsdoshowtarget{\@glsxtr@doshowtarget}
```

rgetsymbolright

```
\newcommand{\glsxtrshowtargetsymbolright}{{\tiny$\triangleleft$}}%
```

argetsymbolleft

```
\newcommand{\glsxtrshowtargetsymbolleft}{{\tiny$\triangleright$}}%
```

showtargetinner Only added to glossaries in v4.50 so may not be defined.

```
\providecommand*{\glsshowtargetinner}[1]{{\glsshowtargetfont [#1]}}
```

sshowtargetfont Only added to glossaries in v4.45 so may not be defined.

```
\providecommand*{\glsshowtargetfont}{\ttfamily\footnotesize}
```

rcontentsymleft

```
\newcommand*{\glsshowtargetinnersymleft}[1]{%
 \glsshowtargetinner{#1}\allowbreak\glsxtrshowtargetsymbolleft}
```

contentsymright

```
\newcommand*{\glsshowtargetinnersymright}[1]{%
 \glsxtrshowtargetsymbolright\allowbreak\glsshowtargetinner{#1}}
```

showtargetouter Only added to glossaries in v4.45 so may not be defined.

```
\providecommand*{\glsshowtargetouter}[1]{%
 \glsshowtargetsymbol\marginpar{\glsshowtargetsymbol\glsshowtargetfont #1}}
```

\@glsshowtarget Only added to glossaries in v4.32 so may not be defined.

```
\providecommand*{\@glsshowtarget}[1]{}
```

\glsshowtarget This command was introduced to glossaries v4.32 so it may not be defined. Therefore it's defined here using \def. \glsshowtargetouter was introduced in glossaries v4.45, so that also may not be defined.

```
\def\glsshowtarget#1{%
 \glsxtrtitleorpdforheading
 {%
   \ifmmode
     \nfss@text{\glsxtrshowtargetinner{#1}}%
   \else
     \ifinner
       \glsxtrshowtargetinner{#1}%
     \else
       \glsxtrshowtargetouter{#1}%
     \fi
   \fi
 }%
 {[#1]}%
 {{\protect\glsshowtargetinner{#1}}}%
}
```

owtargetmarkfmt

```
\newcommand*{\@glsshowtargetmarkfmt}[1]{%
 \glsxtrtitleorpdforheading
 {%
   \ifmmode \nfss@text{#1}\else #1\fi
 }%
 {}%
 {\ifmmode \nfss@text{#1}\else #1\fi}%
}
```

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}}%
```

Disable options that can only be used when the package is loaded:

```
\disable@keys{glossaries-extra.sty}{accsupp}
```

Now define the user command:

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

@@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

usedOrUndefined

\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}%
}
```

Starred form of \ifglossaryexists was only introduced to glossaries v4.46 so provide it if it hasn't been defined.

```
\ifdef\s@ifglossaryexists
{}
{
```

fglossaryexists

```
  \renewcommand{\ifglossaryexists}{%
    \@ifstar\s@ifglossaryexists\@ifglossaryexists
  }
```

fglossaryexists

```
  \newcommand{\@ifglossaryexists}[3]{%
    \ifcsundef{@glotype@#1@out}{#3}{#2}%
  }
```

fglossaryexists

```
  \newcommand{\s@ifglossaryexists}[3]{%
    \ifcsundef{glolist@#1}{#3}{#2}%
  }
```

```
967 }
```

ifemptyglossary

\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.)

```
968 \newcommand{\glsxtrifemptyglossary}[3]{%
969   \ifcsdef{glolist@#1}%
970   {%
971     \ifcsstring{glolist@#1}{,}{#2}{#3}%
972   }%
973   {%
974     \glsxtrundefaction{Glossary type `#1' doesn't exist}{}%
975     #2%
976   }%
977 }
```

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

```
978 \newcommand*{\glsxtrifkeydefined}[3]{%
979   \key@ifundefined{glossentry}{#1}{#3}{#2}%
980 }
```

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

```
981 \newcommand*{\glsxtrprovidestoragekey}{%
982   \@ifstar\@sglsxtr@provide@storagekey\@glsxtr@provide@storagekey
983 }
```

vide@storagekey Unstarred version.

```
984 \newcommand*{\@glsxtr@provide@storagekey}[3]{%
985   \key@ifundefined{glossentry}{#1}%
986   {%
987     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
988     \appto\@gls@keymap{,{#1}{#1}}%
989     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
990     \appto\@newglossaryentryposthook{%
991       \letcs{\@glo@tmp}{@glo@#1}%
992       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
993     }%
```

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

```
994     \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}}{\glsxtrpdfentryfmt{#1}{#2}}%
  }
}
{
  \newcommand*{\glsxtrentryfmt}{\@glsxtrentryfmt}
}
```

`sxtrpdfentryfmt` Use for the PDF bookmarks.

```
\newcommand*{\glsxtrpdfentryfmt}[2]{#2}
```

`@glsxtrentryfmt`

```
\newrobustcmd*{\@glsxtrentryfmt}[2]{%
```

Locally define `\glslabel` in case the helper command needs to access the label.

```
 {%
   \protected@edef\glslabel{#1}%
   \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}%
}
```

fieldformatlist

```
\newrobustcmd*{\glsxtrfieldformatlist}[2]{%
 \begingroup
  \def\@dtl@formatlist@itemsep{}%
  \def\@dtl@formatlist@lastitem{}%
  \def\@dtl@formatlist@prelastitem{}%
  \def\@dtl@formatlist@prelastitemsep{}%
  \forlistcsloop{\@dtl@formatlist@handler}{glo@\glsdetoklabel{#1}@#2}%
  \@dtl@formatlist@prelastitem\@dtl@formatlist@lastitem
 \endgroup
}
```

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}%
}
```

sxtrforcsvfield

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

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

sxtrforcsvfield Unstarred version.

```
\newcommand*{\@glsxtrforcsvfield}[3]{%
```

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

sxtrforcsvfield Starred version.

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

ldformatcsvlist

```
\newrobustcmd*{\glsxtrfieldformatcsvlist}[2]{%
 \@glsxtrifhasfield{#2}{#1}%
 {\@dtlformatlist\glscurrentfieldvalue}%
 {}%
}
```

dValueInCsvList

\GlsXtrIfFieldValueInCsvList{⟨*label*⟩}{⟨*field*⟩}{⟨*list*⟩}{⟨*true*⟩}{⟨*false*⟩}

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

Note \DTLifinlist performs one level on the list but not the element.

dValueInCsvList Unstarred version.

```
\newcommand*{\@GlsXtrIfFieldValueInCsvList}[5]{%
 \@glsxtrifhasfield{#2}{#1}%
 {%
   \expandafter\DTLifinlist\expandafter{\glscurrentfieldvalue}%
   {#3}{#4}{#5}%
 }%
 {#5}%
}
```

dValueInCsvList Starred version.

```
\newcommand*{\s@GlsXtrIfFieldValueInCsvList}[5]{%
```

```
  \s@glsxtrifhasfield{#2}{#1}%
  {%
    \expandafter\DTLifinlist\expandafter{\glscurrentfieldvalue}%
    {#3}{#4}{#5}%
  }%
  {#5}%
}
```

eInFieldCsvList

\GlsXtrIfValueInFieldCsvList{⟨*label*⟩}{⟨*field*⟩}{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩}

Essentially the reverse. Tests if the given value is in the given field which should contain a comma-separated list.

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

eInFieldCsvList Unstarred version.

```
\newcommand*{\@GlsXtrIfValueInFieldCsvList}[5]{%
  \@glsxtrifhasfield{#2}{#1}%
  {%
    \DTLifinlist{#3}{\glscurrentfieldvalue}{#4}{#5}%
  }%
  {#5}%
}
```

eInFieldCsvList Unstarred version.

```
\newcommand*{\s@GlsXtrIfValueInFieldCsvList}[5]{%
  \s@glsxtrifhasfield{#2}{#1}%
  {%
    \DTLifinlist{#3}{\glscurrentfieldvalue}{#4}{#5}%
  }%
  {#5}%
}
```

eInFieldCsvList

\xGlsXtrIfValueInFieldCsvList{⟨*label*⟩}{⟨*field*⟩}{⟨*value*⟩}{⟨*true*⟩}{⟨*false*⟩}

As above but fully expand ⟨*value*⟩.

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

eInFieldCsvList Unstarred version.

```
\newcommand*{\@xGlsXtrIfValueInFieldCsvList}[5]{%
 \@glsxtrifhasfield{#2}{#1}%
 {%
   \protected@edef\@gls@tmp{#3}%
   \expandafter\DTLifinlist\expandafter{\@gls@tmp}{\glscurrentfieldvalue}{#4}{#5}%
 }%
 {#5}%
}
```

eInFieldCsvList Unstarred version.

```
\newcommand*{\s@xGlsXtrIfValueInFieldCsvList}[5]{%
 \s@glsxtrifhasfield{#2}{#1}%
 {%
   \protected@edef\@gls@tmp{#3}%
   \expandafter\DTLifinlist\expandafter{\@gls@tmp}{\glscurrentfieldvalue}{#4}{#5}%
 }%
 {#5}%
}
```

lsxtrifhasfield

\glsxtrifhasfield{⟨*field*⟩}{⟨*label*⟩}{⟨*true*⟩}{⟨*false*⟩}

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}{%
  \@ifstar\s@GlsXtrIfFieldNonZero\@GlsXtrIfFieldNonZero
}
```

rIfFieldNonZero

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

rIfFieldNonZero

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

XtrIfFieldEqNum

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

Designed for numeric fields.

```
\newcommand{\GlsXtrIfFieldEqNum}{%
  \@ifstar\s@GlsXtrIfFieldEqNum\@GlsXtrIfFieldEqNum
}
```

XtrIfFieldEqNum

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

XtrIfFieldEqNum

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

trIfFieldCmpNum

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

Designed for numeric fields.

```
\newcommand{\GlsXtrIfFieldCmpNum}{%
  \@ifstar\s@GlsXtrIfFieldCmpNum\@GlsXtrIfFieldCmpNum
}
```

trIfFieldCmpNum

```
\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
  }%
}
```

trIfFieldCmpNum

```
\newcommand{\s@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
}
```

XtrIfFieldUndef

\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}}}%
  }
}
```

`entryparentname`

```
\newcommand*{\glsxtrentryparentname}[1]{%
  \ifcsdef{glo@\glsdetoklabel{#1}@parent}%
  {\csuse{glo@\csuse{glo@\glsdetoklabel{#1}@parent}@name}}%
  {}%
}
```

`\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}}
```

`trapptocsvfield` Similar to the above but will append value with a leading comma if the field is already defined. This is used by bib2gls. There's no check if the entry has been defined. (Because of the way that bib2gls's save-from-alias etc options are implemented, the entry may not have yet been written to the glstex file when this command is used.)

```
\newcommand*{\glsxtrapptocsvfield}[3]{%
 \ifcsdef{glo@\glsdetoklabel{#1}@#2}%
 {\csappto{glo@\glsdetoklabel{#1}@#2}{,#3}}%
 {\csdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

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 Starred version uses starred version of \glsxtrifhasfield (that is, no grouping).

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

XtrIfFieldEqStr

```
\newrobustcmd*{\@GlsXtrIfFieldEqStr}[5]{%
  \@glsxtrifhasfield{#1}{#2}%
  {%
    \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
  }%
  {#5}%
}
```

XtrIfFieldEqStr

```
\newrobustcmd*{\s@GlsXtrIfFieldEqStr}[5]{%
  \s@glsxtrifhasfield{#1}{#2}%
  {%
    \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
  }%
  {#5}%
}
```

rIfFieldEqXpStr Like the above but first expands the string. Starred version uses starred version of \glsxtrifhasfield (that is, no grouping).

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

rIfFieldEqXpStr

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

rIfFieldEqXpStr

```
\newrobustcmd*{\s@GlsXtrIfFieldEqXpStr}[5]{%
  \s@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. Starred version uses starred version of \glsxtrifhasfield (that is, no grouping).

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

fXpFieldEqXpStr

```
\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}%
}
```

fXpFieldEqXpStr

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

sXtrForeignText

\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.

The original `\@gls@entry@field` causes a problem for undefined entries when used in section headings or captions. Since entries must be defined with just the base package this isn't a significant issue, but it will cause a problem with `bib2gls` where no entries are defined on the first LaTeX call, so redefine `\@gls@entry@field` to use `\csuse` instead of `\csname`.

gls@entry@field

`\@gls@entry@field{⟨label⟩}{⟨field⟩}`

This command was introduced to glossaries version 4.03 but older versions are likely to be incompatible with glossaries-extra.

```
\ifdef\@gls@entry@field
{
 \renewcommand*{\@gls@entry@field}[2]{\csuse{glo@\glsdetoklabel{#1}@#2}}
}
{}
```

\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. See also `\GlsXtrIfUnusedOrUndefined`.

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

ignoredglossary Starred form.

```
\newcommand*{\glsxtr@s@newignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {%
    \glsxtrundefaction{Glossary type `#1' already exists}{}%
  }%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \protected@edef\@ignored@glossaries{#1}%
    }%
    {%
      \protected@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
    {%
      \protected@edef\@ignored@glossaries{#1}%
    }%
    {%
      \protected@eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
```

```
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
    \ifdefempty\@gls@nohyperlist
    {%
       \renewcommand*{\@gls@nohyperlist}{#1}%
    }%
    {%
       \protected@eappto\@gls@nohyperlist{,#1}%
    }%
  }%
}
```

ignoredglossary Starred form.

```
\newcommand*{\glsxtr@s@provideignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {}%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \protected@edef\@ignored@glossaries{#1}%
    }%
    {%
      \protected@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}
    {%
      \protected@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).

```
    \protected@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 restored the original definition as it makes more sense to use the name in the cross-reference list. Unfortunately this doesn't take style changes into account, so as from v1.42, this now uses \glsfmttext and \glsfmtname instead. (The text field is chosen rather than the short field to allow for the "noshort" styles.)

```
\renewcommand*{\glsseeitemformat}[1]{%
 \ifglshasshort{#1}{\glsfmttext{#1}}{\glsfmtname{#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. This now uses \glsfmttext and \glsfmtname instead of \glsaccessshort and \glsaccessname to allow for style formatting.

```
\newcommand*{\glsxtrhiername}[1]{%
```

```
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {\glsxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
    {}%
    \ifglshasshort{#1}{\glsfmttext{#1}}{\glsfmtname{#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}{\glsfmttext{#1}}{\glsfmtname{#1}}%
    }%
    {\ifglshasshort{#1}{\Glsfmttext{#1}}{\Glsfmtname{#1}}}%
  }%
}
```

\GlsXtrhiername

\GlsXtrhiername{⟨*label*⟩}

As above but converts the first letter of each name to a capital. (Note that this isn't applying title case, just capitalising the start of each hierarchical element.)

```
\newcommand*{\GlsXtrhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {\GlsXtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
    {}%
    \ifglshasshort{#1}{\Glsfmttext{#1}}{\Glsfmtname{#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}{\glsfmttext{#1}}{\glsfmtname{#1}}%
    }%
    {\ifglshasshort{#1}{\GLSfmttext{#1}}{\GLSfmtname{#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}{\GLSfmttext{#1}}{\GLSfmtname{#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}%
    }%
  }%
}
```

\glsxtrusealias Apply \glsseeformat to the alias key if not empty. There's no optional tag to worry about here. The value also isn't a comma-separated list, but use the same interface.

```
\newcommand*{\glsxtrusealias}[1]{%
  \glsdoifexists{#1}%
  {%
    \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@alias}%
    \ifdefempty\@glo@see
    {}%
    {%
```

Expansion isn't necessary because the value is a single label not a list.

```
      \glsxtruseseeformat{\seename}{\@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]{%
  \protected@edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
}
```

\glsseelist Redefine to make \glsseelist more flexible.

```
\renewrobustcmd*{\glsseelist}[1]{%
  \let\@gls@dolast\relax
  \let\@gls@donext\relax
  \let\@glsseeitem\@glsxtr@seefirstitem
  \let\@glsseelastsep\glsseelastsep
  \@for\@gls@thislabel:=#1\do{%
    \ifx\@xfor@nextelement\@nnil
      \@gls@dolast
    \else
      \@gls@donext
    \fi
    \expandafter\@glsseeitem\expandafter{\@gls@thislabel}%
    \let\@gls@dolast\@glsseelastsep
    \let\@gls@donext\glsseesep
    \let\@glsseeitem\@glsxtr@seeitem
    \let\@glsseelastsep\glsseelastoxfordsep
  }%
}
```

@glsxtr@seeitem

```
\newcommand*{\@glsxtr@seeitem}[1]{%
 \glsxtrifmulti{#1}{\mglsseeitem{#1}}{\glsseeitem{#1}}%
}
```

tr@seefirstitem

```
\newcommand*{\@glsxtr@seefirstitem}[1]{%
 \glsxtrifmulti{#1}{\mglsseefirstitem{#1}}{\glsseefirstitem{#1}}%
}
```

\mglsseeitem Multi-entry cross-reference

```
\newcommand*{\mglsseeitem}[1]{%
 \mglsname[all={noindex},setup={hyper=allmain}]{#1}%
}
```

glsseefirstitem Multi-entry cross-reference

```
\newcommand*{\mglsseefirstitem}{\mglsseeitem}
```

glsseefirstitem

```
\newcommand*{\glsseefirstitem}{\glsseeitem}
```

eelastoxfordsep

```
\newcommand*{\glsseelastoxfordsep}{\glsseelastsep}
```

\seealsoname In case this command hasn't been defined. Languages packages actually provide \alsoname so use that if it's defined.

```
\ifdef\alsoname
{\providecommand{\seealsoname}{\alsoname}}
{\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
    {}%
    {%
      \protected@edef\@do@glssee{\noexpand\glsxtrindexseealso
        {\@glo@label}{\@glo@seealso}}%
      \@do@glssee
    }%
  }%
  {%
```

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

```
    \ifdefvoid\@glo@see
    {%
      \protected@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}%
      {}%
      {%

        \protected@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
      {%

        \protected@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
 {%
   \glsxtrifmulti\@glsxtr@label
   {%
     \letcs\@glsxtr@labellist{@gls@combined@\@glsxtr@label @list}%
     \@for\@glsxtr@multilabel:=\@glsxtr@labellist\do
     {\@glsxtr@addunused\@glsxtr@multilabel\@end@glsxtr@addunused}%
   }%
   {%
     \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}{%
 \@domakeglossaries
 {%
  \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

      \protected@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@saveprelocatag}[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
```

Save local setting.

```
    \@gls@save@glslocal
    \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
    \@gls@restore@glslocal
  }%
  \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}%
    \@gls@do@glsunset{#2}%
  }%
  \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]{}
```

AutoAddOnFormat

\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
  }%
}
```

\glslinkwrcontent This may resolve issue #189 but it may have unexpected consequences. This is currently provided as a trial. If it causes a problem then redefine without the grouping. If no issues are reported it will be added to the base package.

```
\providecommand*{\glslinkwrcontent}[1]{{#1}}
```

\@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

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

Save local setting.

```
  \@gls@save@glslocal
```

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}%
   {%
```

```
      \protected@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
```

Encapsulate link text and indexing.

```
  \glslinkwrcontent
  {%
```

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
  \@gls@restore@glslocal
}

\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

        \protected@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. These commands shouldn't be used with `\newabbreviation`, but the redefinitions below allow for users reverting `\newacronym` back to its base definition.

`ase@acrcmd@warn` Warn user that they need to use to new abbreviation commands.

```
\newcommand*{\@@glsxtr@base@acrcmd@warn}[2]{%
  \GlossariesExtraWarning{Base acronym command \string#1\space
   should not be used with new abbreviation definitions. Use
   \string#2\space instead}%
}
```

`xtr@base@acrcmd` Warn user that they need to use to new abbreviation commands.

```
\let\@glsxtr@base@acrcmd\@@glsxtr@base@acrcmd@warn
```

`\@acrshort` No case change.

```
\def\@acrshort#1#2[#3]{%
  \@glsxtr@base@acrcmd\acrshort\glsxtrshort
  \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]{%
  \@glsxtr@base@acrcmd\Acrshort\Glsxtrshort
  \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]{%
  \@glsxtr@base@acrcmd\ACRshort\GLSxtrshort
  \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]{%
  \@glsxtr@base@acrcmd\acrshortpl\glsxtrshortpl
  \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]{%
  \@glsxtr@base@acrcmd\Acrshortpl\Glsxtrshortpl
  \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]{%
  \@glsxtr@base@acrcmd\ACRshortpl\GLSxtrshortpl
  \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]{%
  \@glsxtr@base@acrcmd\acrlong\glsxtrlong
  \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]{%
```

```
  \@glsxtr@base@acrcmd\Acrlong\Glsxtrlong
  \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]{%
  \@glsxtr@base@acrcmd\ACRlong\GLSxtrlong
  \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]{%
  \@glsxtr@base@acrcmd\acrlongpl\glsxtrlongpl
  \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]{%
  \@glsxtr@base@acrcmd\Acrlongpl\Glsxtrlongpl
  \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]{%
  \@glsxtr@base@acrcmd\ACRlongpl\GLSxtrlongpl
  \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
}
```

The full formats use the internal long and short commands (such as \@acrshort and \@acrlong). Therefore they don't need adjustments, but they do need clearer warnings. This means three warnings per use (once for the full command and once each for the short and long commands), but at least this way the most important warning (replace \acrfull with \glsxtrfull etc) is present.

\@acrfull

```
\def\@acrfull#1#2[#3]{%
  \@glsxtr@base@acrcmd\acrfull\glsxtrfull
```

```
    \acrfullfmt{#1}{#2}{#3}%
}
```

\@Acrfull

```
\def\@Acrfull#1#2[#3]{%
  \@glsxtr@base@acrcmd\Acrfull\Glsxtrfull
  \Acrfullfmt{#1}{#2}{#3}%
}
```

\@ACRfull

```
\def\@ACRfull#1#2[#3]{%
  \@glsxtr@base@acrcmd\ACRfull\GLSxtrfull
  \ACRfullfmt{#1}{#2}{#3}%
}
```

\@acrfullpl

```
\def\@acrfullpl#1#2[#3]{%
  \@glsxtr@base@acrcmd\acrfullpl\glsxtrfullpl
  \acrfullplfmt{#1}{#2}{#3}%
}
```

\@Acrfullpl

```
\def\@Acrfullpl#1#2[#3]{%
  \@glsxtr@base@acrcmd\Acrfullpl\Glsxtrfullpl
  \Acrfullplfmt{#1}{#2}{#3}%
}
```

\@ACRfullpl

```
\def\@ACRfullpl#1#2[#3]{%
  \@glsxtr@base@acrcmd\ACRfullpl\GLSxtrfullpl
  \ACRfullplfmt{#1}{#2}{#3}%
}
```

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`). Note that if the entry is undefined (as with `bib2gls` on the first LaTeX run), `\ifglsused` does neither true nor false parts. However, in that case, this macro won't be called anyway (since it's used in the argument of `\glsdoifexistsordo`).

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

Store the category label for convenience.

```
  \protected@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
```

s@save@glslocal Defined in glossaries v4.50 so may not be defined.

```
\providecommand*{\@gls@save@glslocal}{%
  \let\if@org@KV@glslink@local\ifKV@glslink@local
}
```

estore@glslocal Defined in glossaries v4.50 so may not be defined.

```
\providecommand*{\@gls@restore@glslocal}{%
  \ifKV@glslink@local
    \let\@gls@do@glsunset\glslocalunset
  \else
    \let\@gls@do@glsunset\glsunset
  \fi
}
```

gls@do@glsunset Defined in glossaries v4.50 so may not be defined.

```
\providecommand*{\@gls@do@glsunset}[1]{\glsunset{#1}}
```

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` The change made in v1.46 to remove the grouping has had the knock-on effect of redefining `\glscurrentfieldvalue`, which may be a problem, so v1.47 has changed this to use `\ifcsvoid`.

```
\newcommand*{\@glsxtrsetaliasnoindex}{%
 \ifcsvoid{glo@\glsdetoklabel{\glslabel}@alias}%
 {}%
 {%
   \let\glsxtrindexaliased\@glsxtrindexaliased
   \glsxtrsetaliasnoindex
   \let\glsxtrindexaliased\@no@glsxtrindexaliased
 }%
}
```

`xtrindexaliased`

```
\newcommand{\@glsxtrindexaliased}{%
 \ifKV@glslink@noindex
 \else
   \begingroup
   \let\@glsnumberformat\@glsxtr@defaultnumberformat
   \protected@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. This needs to use \GlsXtrIfUnusedOrUndefined instead of \ifglsused to allow it to work with bib2gls.

```
\renewcommand*{\glswriteentry}[2]{%
  \glsxtrifindexing
  {%
   \ifglsindexonlyfirst
     \GlsXtrIfUnusedOrUndefined{#1}
     {#2}%
     {\glsxtrdoautoindexname{#1}{dualindex}}%
   \else
     \glsifattribute{#1}{indexonlyfirst}{true}%
     {%
       \GlsXtrIfUnusedOrUndefined{#1}%
       {#2}%
       {\glsxtrdoautoindexname{#1}{dualindex}}%
     }%
     {#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
```

Check that the supplied character isn't + or *

```
  \ifstrequal{#1}{+}%
  {\PackageError{glossaries-extra}%
   {Can't use '#1' as modifier (it's already in use)}{}}%
  {%
    \ifstrequal{#1}{*}%
    {\PackageError{glossaries-extra}%
     {Can't use '#1' as modifier (it's already in use)}{}}%
    {}%
  }%
  \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` Since `\glsnavhyperlink` uses `\@glslink`, it's necessary to patch it uses `\glsdohyperlink` instead of `\glsxtrdohyperlink`. The simplest way to achieve this is to locally let `\glsxtrdohyperlink` to `\glsdohyperlink`.

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

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

Scope:

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

Patch if glossaries pre 4.50.

```
\ifdef\@@gls@navhypertarget
{}
{%
```

snavhypertarget

```
\renewcommand*{\glsnavhypertarget}{\protect\@@gls@navhypertarget}
```

@navhypertarget

```
\newcommand*{\@@gls@navhypertarget}[3][\@glo@type]{%
  \@glsnavhypertarget{#1}{#2}{#3}%
}

}%
```

snavhypertarget Similarly for \@glsnavhypertarget

```
\ifdef\@glsnavhypertarget
{%
  \renewcommand*{\@glsnavhypertarget}[3]{%
    \protected@write\@auxout{}{\string\@gls@hypergroup{#1}{#2}}%
    \@glsxtr@do@org@target{\glsnavhyperlinkname{#1}{#2}}{#3}%
    \ifcsdef{@gls@hypergrouplist@#1}%
    {%
      \letcs\@gls@list{@gls@hypergrouplist@#1}%
      \protected@edef\@gls@thishypernavlabel{#2}%
      \expandafter\DTLifinlist\expandafter{\@gls@thishypernavlabel}\@gls@list{}%
      {%
        \GlossariesWarningNoLine{Navigation panel
           for glossary type `#1'^^Jmissing group `#2'}%
        \gdef\gls@hypergrouprerun{%
          \GlossariesWarningNoLine{Navigation panel
          has changed. Rerun LaTeX}}%
      }%
    }%
    {%
      \GlossariesWarningNoLine{Navigation panel
         for glossary type `#1'^^Jmissing group `#2'}%
      \gdef\gls@hypergrouprerun{%
        \GlossariesWarningNoLine{Navigation panel
        has changed. Rerun LaTeX}}%
    }%
  }%
}
{}
```

The redefinition of \glsdohyperlink has been causing problems so introduce a new command instead.

sxtrdohyperlink 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.

```
\newcommand*{\glsxtrdohyperlink}[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}}%
   }%
   {%
```

Is the alias a multi-entry?

```
     \glsxtrifmulti\gloaliaslabel
     {%
```

Get the main target.

```
       \letcs\gloaliaslabel{@gls@combined@\gloaliaslabel @main}%
     }%
     {}%
```

Redirect link to the alias target.

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

```
}
```

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

```
\newcommand{\glsxtrhyperlink}[2]{%
  \glsdoshowtarget{#1}{\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}%
   {\protected@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{\glsxtrdohyperlink}%
 \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{\glsxtrdohyperlink}
}
```

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@
                3628 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
                3629   {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
                3630 \def\@glsxtr@p@acrlong@#1#2[#3]{{\glsentrylong{#2}#3}}

sxtr@p@acrlong@
                3631 \def\@Glsxtr@p@acrlong@#1#2[#3]{{\Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
                3632 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
                3633  {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}

tr@p@acrlongpl@
                3634 \def\@glsxtr@p@acrlongpl@#1#2[#3]{{\glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
                3635 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{{\Glsentrylongpl{#2}#3}}

tr@p@acrlongpl@
                3636 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
                3637  {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}
```

Commands to minimise conflict.

```
  \@glsxtrp@opt
                3638 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}
```

`\glsxtrsetpopts` Used in glossary to switch hyperlinks on for the `\glsxtrp` type of commands.

```
                3639 \newcommand*{\glsxtrsetpopts}[1]{%
                3640   \renewcommand*{\@glsxtrp@opt}{#1}%
                3641 }
```

`lossxtrsetpopts` Used in glossary to switch hyperlinks on for the `\glsxtrp` type of commands.

```
                3642 \newcommand*{\glossxtrsetpopts}{%
                3643   \glsxtrsetpopts{noindex}%
                3644 }

     \@@glsxtrp
                3645 \newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
                3646   {%
                3647     \let\glspostlinkhook\relax
                3648     \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
                3649   }%
                3650 }
```

\@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
}
```

dUnsetBuffering Discards pending buffer and restores \glsunset.

```
\newcommand*{\GlsXtrDiscardUnsetBuffering}{%
  \let\@glsunset\@glsxtr@unset
  \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
}
```

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
}
```

rifcounttrigger

\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}{%
```

```
  \protected@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}%
  {%
    \protected@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}%
  {%
    \protected@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}%
  {%
    \protected@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}%
  {%
    \protected@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}{%
```

Make sure `\RestoreAcronyms` has been used.

```
  \ifdefequal\@addtoacronymlists\@glsxtr@org@addtoacronynlists
  {}%
  {%
    \GlossariesWarning{\string\SetGenericNewAcronym\space used
    without restoring base acronym functions with
    \string\RestoreAcronyms}%
  }%
  \let\@Gls@entryname\@Gls@acrentryname
```

Redefine `\newacronym`:

```
  \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
```

Save the list of acronyms in case they are required.

tr@acronymlists

```
\let\@glsxtr@acronymlists\@glsacronymlists
```

dtoacronynlists

```
\let\@glsxtr@org@addtoacronynlists\@addtoacronymlists
```

setacronymlists

```
\let\@glsxtr@org@setacronymlists\SetAcronymLists
```

Need to provide a replacement for `\forallacronyms` since `\@glsacronymlists` isn't available.

lsxtr@abbrlists

```
\newcommand{\@glsxtr@abbrlists}{}
```

breviationlists

```
\newcommand*{\forallabbreviationlists}[2]{%
```

```
  \@for#1:=\@glsxtr@abbrlists\do{\ifdefempty{#1}{}{#2}}%
}
```

bbreviationlist

```
\newcommand*{\@glsxtr@addabbreviationlist}[1]{%
  \protected@edef\@glo@type{#1}%
  \ifdefempty\@glsxtr@abbrlists
  {\let\@glsxtr@abbrlists\@glo@type}%
  {%
    \ifdefequal\@glsxtr@abbrlists\@glo@type
    {}%
    {%
      \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glsxtr@abbrlists}{}%
      {\protected@eappto\@glsxtr@abbrlists{,\@glo@type}}%
    }%
  }%
}
```

\forallacronyms Modify to add warning.

```
\renewcommand*{\forallacronyms}[2]{%
  \@glsxtr@base@acrcmd\forallacronyms\forallabbreviationlists
  \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
}
```

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}{%
```

Undo acronym display style:

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \csgdef{gls@\@gls@type @entryfmt}{\glsentryfmt}%
  }%
```

Save and clear acronym list.

```
  \let\@glsxtr@acronymlists\@glsacronymlists
  \let\@glsacronymlists\@empty
  \let\@addtoacronymlists\@gobble
  \let\SetAcronymLists\@gobble
```

Warn if \acrshort etc are used.

```
  \let\@glsxtr@base@acrcmd\@@glsxtr@base@acrcmd@warn
```

Redefine \newacronym to use same interface as \newabbreviation.

```
  \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[acronym]\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}{%
```

Restore acronym list.

```
  \let\@glsacronymlists\@glsxtr@acronymlists
  \let\@addtoacronymlists\@glsxtr@org@addtoacronynlists
  \let\SetAcronymLists\@glsxtr@org@setacronymlists
```

Suppress warnings if `\acrshort` etc are used.

```
  \let\@glsxtr@base@acrcmd\@gobbletwo
```

Restore acronym display style:

```
  \@for\@gls@type:=\@glsacronymlists\do{%
    \SetDefaultAcronymDisplayStyle{\@gls@type}%
  }%
```

Switch to the generic acronym mechanism.

```
  \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
```

noprintglossary This command was only introduced to glossaries v4.47 so it may not be defined.

```
\providecommand\@makeglossaries@warn@noprintglossary{%
  \ifdefstring{\@glo@types}{,}%
  {%
    \GlossariesWarningNoLine{No glossaries have been defined}%
  }%
  {%
    \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}%
  }%
}%
```

omakeglossaries glossaries v4.45 introduced \@domakeglossaries to provide a way of disabling \makeglossaries. If it hasn't been defined, define here to do its argument:

```
\providecommand{\@domakeglossaries}[1]{#1}
```

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][]{%
 \@domakeglossaries
 {%
   \@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=hybrid options}%
   }%
   {%
     \ifblank{#1}%
     {%
        \@glsxtr@org@makeglossaries

        \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
          \let\warn@noprintglossary\@glsxtr@warn@hybrid@noprintgloss
        \fi
     }%
     {%
       \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
         \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
         not permitted\MessageBreak with record=\@glsxtr@record@setting\space package option}%
         {You may only use the hybrid \string\makeglossaries[...]\space with
          record=off option}%
       \else
```

\@gls@@automake@immediate was introduced to glossaries v4.42 so it may not be defined.

```
         \ifdef\@gls@@automake@immediate{\@gls@@automake@immediate}{}%
         \protected@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\@gobble
```

Version 1.42 removed letting `\makeglossary` to `\relax` (no kernel redefs may be in effect).

```
        \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}%
          {%

            \protected@edef\@gls@label{\glsdetoklabel{##1}}%
            \protected@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{%

          \protected@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
        \let\warn@noprintglossary\@makeglossaries@warn@noprintglossary
```

Only warn for glossaries not listed.

```
        \renewcommand{\@gls@noref@warn}[1]{%
```

```
      \protected@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
```

```
4671              \fi
4672            }%
4673          }%
```

Unlike `\makenoidxglossaries` we can't automatically set sanitizesort=false. All entries must be defined in the preamble.

```
4674          \renewcommand*\new@glossaryentry[2]{%
4675            \PackageError{glossaries-extra}{Glossary entries must be defined
4676             in the preamble\MessageBreak when you use the optional argument
4677             of \string\makeglossaries}{Either move your definitions to the
4678             preamble or don't use the optional argument of
4679             \string\makeglossaries}%
4680          }%
```

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).

```
4681          \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
4682          \renewcommand*{\@printgloss@setsort}{%
```

Need to extract just the type value.

```
4683            \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%
4684              type=\glsdefaulttype,\@end@glsxtr@gettype
4685            \def\@glo@sorttype{\@glo@default@sorttype}%
4686          }%
```

Check automake setting:

```
4687          \ifglsautomake
4688            \renewcommand*{\@gls@doautomake}{%
4689              \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
4690                \ifdefempty{\@gls@type}{}{\@gls@automake{\@gls@type}}%
4691              }%
4692            }%
4693          \fi
```

Check the sort setting (glossaries v4.30 onwards):

```
4694          \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
4695        \fi
4696      }%
4697    }%
4698  }%
4699 }
```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

Earlier versions of glossaries-extra simply saved the original version of `\@printglossary` with `\let \@glsxtr@orgprintglossary`. This was later changed to actually defining `\@glsxtr@orgprintglossary` to something similar with some alterations to allow for ignored glossaries, which don't have an associated title and to by-pass the existence check with `\ifglossaryexists` which doesn't recognise ignored glossaries. (bib2gls writes `\provideignoredglossary` to the glstex file for some settings, so the glossary might not been defined on the first LaTeX run and it needs to be allowed with `\printunsrtglossary` on subsequent runs.)

Unfortunately, removing the existence check will cause an error if \printglossary is used with an ignored glossary.

As from glossaries v4.46, some new commands have been included to allow the existence check to be varied depending on whether or not ignored glossaries should be allowed, so check for them:

oss@checkexists

```
\ifdef\@printgloss@checkexists
{\newcommand{\glsxtr@printgloss@checkexists}{\@printgloss@checkexists}}
{\newcommand{\glsxtr@printgloss@checkexists}[2]{#2}}
```

rgprintglossary (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]{%
      \protected@xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
      \gls@org@glossaryentryfield{##1}%
    }%
    \renewcommand{\subglossentry}[2]{%
      \protected@xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
      \gls@org@glossarysubentryfield{##1}{##2}%
    }%
    \@gls@preglossaryhook
    \glsxtr@printgloss@checkexists{\@glo@type}{#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}%
}

\define@key{printgloss}{label}{%
  \glsxtrsetglossarylabel{#1}%
}
```

etglossarylabel Set the label for subsequent glossaries. If the label is fixed (that is, doesn't change with each glossary) this will need to be scoped or changed again to prevent duplicate labels.

```
\newcommand{\glsxtrsetglossarylabel}[1]{%
  \renewcommand*{\@@glossaryseclabel}{%
    \protected@edef\@currentlabelname{\glossarytoctitle}%
    \label{#1}%
  }%
}
```

xtr@leveloffset

```
\newcount\@glsxtr@leveloffset
```

New to v1.44:

```
\define@key{printgloss}{leveloffset}{%
  \@glsxtr@assign@leveloffset#1\relax
}
```

ign@leveloffset

```
\newcommand*{\@glsxtr@assign@leveloffset}{%
 \@ifnextchar+{\p@glsxtr@assign@leveloffset}{\np@glsxtr@assign@leveloffset}%
}
```

ign@leveloffset Discard initial + character.

```
\newcommand*{\p@glsxtr@assign@leveloffset}[1]{%
 \@ifnextchar+{\pp@glsxtr@assign@leveloffset}{\np@glsxtr@assign@leveloffset}%
}
```

ign@leveloffset

```
\def\np@glsxtr@assign@leveloffset#1\relax{\@glsxtr@leveloffset=#1\relax}
```

ign@leveloffset

```
\def\pp@glsxtr@assign@leveloffset#1\relax{\advance\@glsxtr@leveloffset by #1\relax}
```

```
\define@boolkey{printgloss}[glsxtr@printgloss@]{groups}[true]{}
\glsxtr@printgloss@groupstrue
```

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
```

r@do@org@target Provide a way to locally do the original.

```
\newcommand{\@glsxtr@do@org@target}[2]{%
 {%
  \let\glsdohypertarget\@glsxtr@org@glsdohypertarget
  \@glstarget{#1}{#2}%
 }%
}
```

@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]{%

  \protected@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]{%
  \protected@edef\glsxtrlocrangefmt{#1}%
  \ifx\glsxtrlocrangefmt\empty
    \def\glsxtrlocrangefmt{glsnumberformat}%
  \fi
  \expandafter\glsxtrdisplaysingleloc
    \expandafter{\glsxtrlocrangefmt}{#2}%
}
```

trdisplayendloc End of a location range.

```
\newcommand*{\glsxtrdisplayendloc}[2]{%
  \protected@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\@glo@tmp{\the\toks@}%
   \ifx\@glo@tmp\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
}
```

cationhyperlink

\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 \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=\@glsxtr@record@setting} 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=\@glsxtr@record@setting\space 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.

`ResourceOptions` Default resource options.

```
\newcommand*{\GlsXtrDefaultResourceOptions}{}
```

`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
\ifdefempty\GlsXtrDefaultResourceOptions
{%
  \protected@write\@auxout{\glsxtrresourceinit}%
  {\string\glsxtr@resource{#1}{#2}}%
}%
{%
 \protected@write\@auxout{\glsxtrresourceinit}%
  {\string\glsxtr@resource{\GlsXtrDefaultResourceOptions,#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. This has to be done before the aux file is opened (so package options automake=immediate and automake=true are identical if just bib2gls is used). 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
```

Check if order=letter has been used by mistake (but not if record=alsoindex has been used).

```
  \@glsxtr@if@record@only
  {\ifdefstring{\glsorder}{letter}%
   {\GlossariesExtraWarningNoLine{Package option ‘order=letter’ isn’t
    supported with ‘record=\@glsxtr@record@setting’. Use ‘break-at=none’
    resource option instead}}%
   {}%
  }%
  {}%
}
```

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
        \protected@edef\glscurrententrylabel{\glsdetoklabel{#1}}%
        \protected@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

        \protected@edef\glscurrententrylabel{\glsdetoklabel{#1}}%
        \protected@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. Check for \@printgloss@checkexists which was introduced to glossaries v4.46.

```
\ifdef\@printgloss@checkexists
{
  \newcommand*{\printunsrtglossary}{%
    \let\@printgloss@checkexists\@printgloss@checkexists@allowignored
    \@ifstar\s@printunsrtglossary\@printunsrtglossary
  }
}
{
  \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
        {%
          \ifglsxtr@printgloss@groups
```

This still uses `\ifglshasparent` to determine whether or not to check for a change in the letter group. (It doesn't take the level offset into account because `bib2gls` only saves the group information for parentless entries.)

```
            \ifglshasparent{\glscurrententrylabel}{}%
            {%
              \@glsxtr@checkgroup\glscurrententrylabel
              \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
                {\@glsxtr@groupheading}%
            }%
          \fi
```

```
          \protected@eappto\@glsxtr@doglossary{%
            \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
        }%
      }%
    }%
    \appto\@glsxtr@doglossary{\end{theglossary}}%
    \printunsrtglossarypredoglossary
    \@glsxtr@doglossary
  }%
  \glossarypostamble
}
```

rtinnerglossary Similar to \printunsrtglossary but doesn't add the section heading, preamble, postamble or start and end of theglossary. Grouping is automatically applied so it may cause a problem within tabular-like environments. The beginning and ending of theglossary should be added around this command (but ensure the style has been set first). The simplest way of doing this is to place \printunsrtinnerglossary inside the printunsrtglossarywrap environment.

```
\newcommand*{\printunsrtinnerglossary}[3][]{%
  \begingroup
   \def\@glsxtr@printglossopts{#1}%
   \def\@glo@type{\glsdefaulttype}%
   \setkeys{printgloss}[title,toctitle,style,numberedsection,sort,label]{#1}%
   \let\currentglossary\@glo@type
   #2%
   \@print@unsrt@innerglossary
   #3%
  \endgroup
}
```

srtglossarywrap

```
\newenvironment{printunsrtglossarywrap}[1][]%
{%
  \def\@glsxtr@printglossopts{#1}%
  \def\@glo@type{\glsdefaulttype}%
  \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%
  \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
```

```
  \@printgloss@setsort
  \setkeys{printgloss}{#1}%
```

The type key simply allows the title to be set if the title key isn't supplied.

```
  \ifglossaryexists*{\@glo@type}%
  {%
   \ifx\glossarytitle\org@glossarytitle
   \else
     \expandafter\let\csname @glotype@\@glo@type @title\endcsname
                    \glossarytitle
   \fi
   \let\currentglossary\@glo@type
  }%
  {}%
  \let\org@glossaryentrynumbers\glossaryentrynumbers
  \let\glsnonextpages\@glsnonextpages
  \let\glsnextpages\@glsnextpages
  \let\nopostdesc\@nopostdesc
  \gls@dotoctitle
  \@glossarystyle
  \let\gls@org@glossaryentryfield\glossentry
  \let\gls@org@glossarysubentryfield\subglossentry

  \renewcommand{\glossentry}[1]{%
    \protected@xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
    \gls@org@glossaryentryfield{##1}%
  }%
  \renewcommand{\subglossentry}[2]{%
    \protected@xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
    \gls@org@glossarysubentryfield{##1}{##2}%
  }%
  \@gls@preglossaryhook
  \glossarysection[\glossarytoctitle]{\glossarytitle}%
  \glossarypreamble
  \begin{theglossary}%
  \glossaryheader
  \glsresetentrylist
}%
{%
  \end{theglossary}%
  \glossarypostamble
  \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
  \global\let\warn@noprintglossary\relax
}
```

t@innerglossary This is much like `\@print@unsrt@innerglossary` but only contains what would normally be the content of the theglossary.

```
\newcommand*{\@print@unsrt@innerglossary}{%
```

No section header or preamble.

```
  \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{}%
```

No header or reset.

```
    \def\@glsxtr@doglossary{}%
    \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
        {%
          \ifglsxtr@printgloss@groups
```

This still uses `\ifglshasparent` to determine whether or not to check for a change in the letter group. (It doesn't take the level offset into account because `bib2gls` only saves the group information for parentless entries.)

```
            \ifglshasparent{\glscurrententrylabel}{}%
            {%
              \@glsxtr@checkgroup\glscurrententrylabel
              \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
                {\@glsxtr@groupheading}%
            }%
          \fi

          \protected@eappto\@glsxtr@doglossary{%
            \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
        }%
      }%
    }%
    \printunsrtglossarypredoglossary
    \@glsxtr@doglossary
  }%
```

No postamble.

```
}
```

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]{%
  \protected@xdef\glscurrententrylabel{#1}%
  \printunsrtglossaryhandler\glscurrententrylabel
}
```

glossaryhandler

```
\newcommand{\printunsrtglossaryhandler}[1]{%
  \glsxtrunsrtdo{#1}%
}
```

triflabelinlist

\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}}%
    \protected@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}%
```

Use level number to determine whether or not this entry has a parent.

```
    \gls@level=\numexpr\csuse{glo@\glsdetoklabel{#1}@level}+\@glsxtr@leveloffset\relax
    \ifnum\gls@level>0
      \let\@glsxtr@ifischild\@firstoftwo
    \else
      \let\@glsxtr@ifischild\@secondoftwo
    \fi
```

Some glossary styles (such as topicmcols) save the level using \def so make sure \gls@level is expanded before being passed to \subglossentry.

```
    \@glsxtr@ifischild
    {%
      \ifdefvoid{\@gls@location}%
      {%
        \ifdefvoid{\@gls@loclist}%
        {%
          \expandafter\subglossentry\expandafter{\number\gls@level}{#1}{}%
        }%
        {%
          \expandafter\subglossentry\expandafter{\number\gls@level}{#1}%
          {%
            \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
          }%
        }%
      }%
      {%
        \expandafter\subglossentry\expandafter
          {\number\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}%
   }%
 }%
}
```

ifrecordtrigger

\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]{%
  \protected@edef\glslabel{\glsdetoklabel{#2}}%
  \let\@gls@link@label\glslabel
  \def\@glsxtr@thevalue{}%
  \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
  \def\@glsnumberformat{glstriggerrecordformat}%
  \protected@edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
```

```
  \protected@edef\glstype{\csname glo@\glslabel @type\endcsname}%
  \def\@glsxtr@thevalue{}%
  \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

Save local setting.

```
  \@gls@save@glslocal
  \glsxtrinitwrgloss
  \glslinkpresetkeys
  \setkeys{glslink}{#1}%
  \glslinkpostsetkeys
  \ifdefempty{\@glsxtr@thevalue}%
  {%
    \@gls@saveentrycounter
  }%
  {%
    \let\theglsentrycounter\@glsxtr@thevalue
    \def\theHglsentrycounter{\@glsxtr@theHvalue}%
  }%
  \glslinkwrcontent
  {%
    \ifglsxtrinitwrglossbefore
      \@do@wrglossary{#2}%
    \fi
    #3%
    \ifglsxtrinitwrglossbefore
    \else
      \@do@wrglossary{#2}%
    \fi
  }%
  \@gls@restore@glslocal
  \@gls@do@glsunset{#2}%
}
```

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
6024 \newcommand*{\@rGlspl}[2][]{%
6025   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
6026 }

\@rGlspl@
6027 \def\@rGlspl@#1#2[#3]{%
6028   \glsxtrifrecordtrigger{#2}%
6029   {%
6030     \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
6031   }%
6032   {%
6033     \@Glspl@{#1}{#2}[#3]%
6034   }%
6035 }%

\rGLS
6036 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\@rGLS}

\@rGLS
6037 \newcommand*{\@rGLS}[2][]{%
6038   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}[]}%
6039 }

\@rGLS@
6040 \def\@rGLS@#1#2[#3]{%
6041   \glsxtrifrecordtrigger{#2}%
6042   {%
6043     \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
6044   }%
6045   {%
6046     \@GLS@{#1}{#2}[#3]%
6047   }%
6048 }%

\rGLSpl
6049 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\@rGLSpl}

\@rGLSpl
6050 \newcommand*{\@rGLSpl}[2][]{%
6051   \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2}[]}%
6052 }

\@rGLSpl@
6053 \def\@rGLSpl@#1#2[#3]{%
6054   \glsxtrifrecordtrigger{#2}%
6055   {%
6056     \@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\@glo@tmp{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
         {\glsgetattribute{\glslabel}{linkcountmaster}}}%
       \@glo@tmp
     }%
     {}%
   }%
```

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}
```

bleLinkCounting

\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}{pluralaccess}{%
  \def\@gls@pluralaccess{#1}%
}
\define@key{glsxtrabbrv}{firstaccess}{%
  \def\@gls@firstaccess{#1}%
}
\define@key{glsxtrabbrv}{firstpluralaccess}{%
  \def\@gls@firstpluralaccess{#1}%
}
\define@key{glsxtrabbrv}{shortaccess}{%
  \def\@gls@shortaccess{#1}%
}
\define@key{glsxtrabbrv}{shortpluralaccess}{%
  \def\@gls@shortaccesspl{#1}%
}
\define@key{glsxtrabbrv}{longaccess}{%
  \def\@gls@longaccess{#1}%
}
\define@key{glsxtrabbrv}{shortlonglaccess}{%
  \def\@gls@longaccesspl{#1}%
}
```

@initaccesskeys

```
\newcommand*{\@gls@initaccesskeys}{%
  \def\@gls@nameaccess{}%
  \def\@gls@textaccess{}%
  \def\@gls@pluralaccess{}%
  \def\@gls@firstaccess{}%
  \def\@gls@firstpluralaccess{}%
  \def\@gls@shortaccess{}%
  \def\@gls@shortaccesspl{}%
  \def\@gls@longaccess{}%
  \def\@gls@longaccesspl{}%
}
```

ssattribute@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}%
    }%
  }%
}
```

As from glossaries v4.45, the replacement text support has been corrected so that the accessibility support for abbreviations use the "E" (expanded value) element. This should actually contain the long form since it's supposed to explain the abbreviation. This is a bit redundant on first use for styles like long-short.

aultshortaccess

\glsdefaultshortaccess{⟨*long*⟩}{⟨*short*⟩}

This command was only introduced to glossaries-accsupp 1.42 so it may not be defined.

```
  \def\glsdefaultshortaccess#1#2{#1 (#2)}
```

signactualsetup

```
  \newcommand{\glsxtrassignactualsetup}{%
   \let\@\empty
   \let\emph\@firstofone
   \let\textbf\@firstofone
   \let\textmd\@firstofone
   \let\textit\@firstofone
   \let\textsl\@firstofone
   \let\textsc\@firstofone
   \let\textrm\@firstofone
   \let\textsf\@firstofone
   \let\texttt\@firstofone
  }
```

s@assign@actual

```
  \ifdef\pdfstringdef
  {
     \newcommand{\@gls@assign@actual}{%
```

```
      \begingroup
       \glsxtrassignactualsetup
       \pdfstringdef\@gls@actualshort{\glsxtrorgsshort}%
       \pdfstringdef\@gls@actuallong{\glsxtrorglong}%
       \pdfstringdef\@gls@actualshortpl{\@gls@shortpl}%
       \pdfstringdef\@gls@actuallongpl{\@gls@longpl}%
      \protected@edef\@gls@tmp{\endgroup
         \def\noexpand\@gls@actualshort{\expandonce\@gls@actualshort}%
         \def\noexpand\@gls@actuallong{\expandonce\@gls@actuallong}%
         \def\noexpand\@gls@actualshortpl{\expandonce\@gls@actualshortpl}%
         \def\noexpand\@gls@actuallongpl{\expandonce\@gls@actuallongpl}%
       }%
       \@gls@tmp
    }
  }
  {
    \newcommand{\@gls@assign@actual}{%
      \begingroup
       \glsxtrassignactualsetup
       \protected@edef\@gls@tmp{\endgroup
         \def\noexpand\@gls@actualshort{\glsxtrorgsshort}%
         \def\noexpand\@gls@actuallong{\glsxtrorglong}%
         \def\noexpand\@gls@actualshortpl{\@gls@shortpl}%
         \def\noexpand\@gls@actuallongpl{\@gls@longpl}%
       }%
       \@gls@tmp
    }
  }
```

`lt@short@access` Renamed `\@gls@setup@default@access` and removed argument since it can be obtained from `\glsxtrorgsshort`.

`@default@access` Assign the default value of the shortaccess key. The argument is the short value passed to `\newabbreviation`. The shortaccess value should explain the abbreviation.

```
  \newcommand{\@gls@setup@default@access}{%
    \@gls@assign@actual
    \ifdefempty\@gls@shortaccess
    {%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```
      \@gls@ifaccessattribute@set{insertdots}%
      {%
        \expandafter\@glsxtr@insertdots\expandafter\@gls@actualshort\expandafter
         {\@gls@actualshort}%
      }%
      {}%
      \ifdefempty\@gls@longaccess
      {%
        \protected@edef\@gls@shortaccess{\glsdefaultshortaccess
```

```
      {\expandonce\@gls@actuallong}{\expandonce\@gls@actualshort}}%
  }%
  {%
    \protected@edef\@gls@shortaccess{\glsdefaultshortaccess
      {\expandonce\@gls@longaccess}{\expandonce\@gls@actualshort}}%
  }%
  \eappto\ExtraCustomAbbreviationFields{shortaccess={\@gls@shortaccess},}%
```

If shortaccessplural hasn't been set, assign plural form.

```
  \ifdefempty\@gls@shortaccesspl
  {%
    \@gls@ifaccessattribute@set{aposplural}%
    {%
      \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
        \@gls@actualshort'\glsxtrabbrvpluralsuffix}%
    }%
    {%
      \@gls@ifaccessattribute@set{noshortplural}%
      {%
         \let\@gls@shortaccesspl\@gls@shortaccess
      }%
      {%
        \let\@gls@shortaccesspl\@gls@actualshortpl
      }%
    }%
  \ifdefempty\@gls@longaccesspl
  {%
    \protected@edef\@gls@shortaccesspl{\glsdefaultshortaccess
      {\expandonce\@gls@actuallongpl}{\expandonce\@gls@actualshortpl}}%
  }%
  {%
    \protected@edef\@gls@shortaccesspl{\glsdefaultshortaccess
      {\expandonce\@gls@longaccesspl}{\expandonce\@gls@actualshort}}%
  }%
  \eappto\ExtraCustomAbbreviationFields{shortpluralaccess={\@gls@shortaccesspl},}%
  }%
  {}%
}%
{%
  \ifdefempty\@gls@shortaccesspl
  {\let\@gls@shortaccesspl\@gls@shortaccess}%
  {}%
}%
```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```
\ifdefempty\@gls@nameaccess
{%
  \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
  {%
```

```
      \eappto\ExtraCustomAbbreviationFields{access={\@gls@shortaccess},}%
    }%
    {}%
  }%
  {}%
```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```
  \ifdefempty\@gls@textaccess
  {%
    \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
    {%
      \eappto\ExtraCustomAbbreviationFields{textaccess={\@gls@shortaccess},}%
    }%
    {}%
  }%
  {}%
  \ifdefempty\@gls@pluralaccess
  {%
    \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
    {%
      \eappto\ExtraCustomAbbreviationFields{%
         pluralaccess={\@gls@shortaccesspl},%
      }%
    }%
    {}%
  }%
  {}%
```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
  \ifdefempty\@gls@firstaccess
  {%
    \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
    {%
      \eappto\ExtraCustomAbbreviationFields{firstaccess={\@gls@shortaccess},}%
    }%
    {}%
  }%
  {}%
  \ifdefempty\@gls@firstpluralaccess
  {%
    \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
    {%
      \eappto\ExtraCustomAbbreviationFields{%
         firstpluralaccess={\@gls@shortaccesspl},%
      }%
    }%
    {}%
  }%
  {}%
}
```

Provide hooks for \setabbreviationstyle that automatically set the attributes appropriate for the style. If the name is just the short form and the description contains the long form, then it may not be necessary to set nameshortaccess but it would depend on the glossary style.

Need to provide \glsxtr⟨*category*⟩⟨*field*⟩accsupp if not already defined.

ovideaccsuppcmd

```
\newcommand*{\glsxtrprovideaccsuppcmd}[2]{%
  \ifcsundef{glsxtr#1#2accsupp}%
  {\csdef{glsxtr#1#2accsupp}{\glsshortaccsupp}}%
  {}%
}
```

rSetNoLongAttrs For styles where the name, first and text are just the abbreviation.

```
\newcommand*{\glsxtrAccSuppAbbrSetNoLongAttrs}[1]{%
 \glssetcategoryattribute{#1}{nameshortaccess}{true}%
 \glssetcategoryattribute{#1}{firstshortaccess}{true}%
 \glssetcategoryattribute{#1}{textshortaccess}{true}%
 \glsxtrprovideaccsuppcmd{#1}{name}%
 \glsxtrprovideaccsuppcmd{#1}{first}%
 \glsxtrprovideaccsuppcmd{#1}{firstpl}%
 \glsxtrprovideaccsuppcmd{#1}{text}%
 \glsxtrprovideaccsuppcmd{#1}{plural}%
}
```

tFirstLongAttrs For styles where the name and text are just the abbreviation. The first form may just be long or may be short and long.

```
\newcommand*{\glsxtrAccSuppAbbrSetFirstLongAttrs}[1]{%
 \glssetcategoryattribute{#1}{nameshortaccess}{true}%
 \glssetcategoryattribute{#1}{textshortaccess}{true}%
 \glsxtrprovideaccsuppcmd{#1}{name}%
 \glsxtrprovideaccsuppcmd{#1}{text}%
 \glsxtrprovideaccsuppcmd{#1}{plural}%
}
```

tTextShortAttrs For styles where only the text is just the abbreviation. The name and first form may just be long or may be short and long. The name may also be short but followed by the long form in the description.

```
\newcommand*{\glsxtrAccSuppAbbrSetTextShortAttrs}[1]{%
 \glssetcategoryattribute{#1}{textshortaccess}{true}%
 \glsxtrprovideaccsuppcmd{#1}{text}%
 \glsxtrprovideaccsuppcmd{#1}{plural}%
}
```

tNameShortAttrs For styles where only the name is just the abbreviation. The first and subsequent form may just be long or may be short and long.

```
\newcommand*{\glsxtrAccSuppAbbrSetNameShortAttrs}[1]{%
 \glssetcategoryattribute{#1}{nameshortaccess}{true}%
 \glsxtrprovideaccsuppcmd{#1}{name}%
}
```

etNameLongAttrs For styles where the first and text are just the abbreviation. The name may just be long or may be short and long or the name may be short.

```
\newcommand*{\glsxtrAccSuppAbbrSetNameLongAttrs}[1]{%
 \glssetcategoryattribute{#1}{firstshortaccess}{true}%
 \glssetcategoryattribute{#1}{textshortaccess}{true}%
 \glsxtrprovideaccsuppcmd{#1}{first}%
 \glsxtrprovideaccsuppcmd{#1}{firstpl}%
 \glsxtrprovideaccsuppcmd{#1}{text}%
 \glsxtrprovideaccsuppcmd{#1}{plural}%
}
```

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}{}
```

@default@access This does nothing if there's no accessibility support.

```
\newcommand{\@gls@setup@default@access}{}
```

rSetNoLongAttrs This does nothing if there's no accessibility support.

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

tFirstLongAttrs This does nothing if there's no accessibility support.

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

tTextShortAttrs This does nothing if there's no accessibility support.

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

tNameShortAttrs This does nothing if there's no accessibility support.

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

etNameLongAttrs This does nothing if there's no accessibility support.

```
\newcommand*{\glsxtrAccSuppAbbrSetNameLongAttrs}[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.

tegoryattribute

\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}%
}
```

goriesattribute

\glssetcategoriesattribute{⟨*category list*⟩}{⟨*attribute-label*⟩}{⟨*value*⟩}

Similar to above, but globally apply to each category in the list.

```
\newcommand*{\glssetcategoriesattribute}[3]{%
  \@for\@gls@thiscatlabel:=#1\do{%
    \csgdef{@glsxtr@categoryattr@@\@gls@thiscatlabel @#2}{#3}%
  }%
}
```

oriesattributes

\glssetcategoriesattributes{⟨*category list*⟩}{⟨*attribute-label list*⟩}{⟨*value*⟩}

Similar to above, but apply to each category and attribute in the list.

```
\newcommand*{\glssetcategoriesattributes}[3]{%
```

Group to avoid problems with nested `\@for`.

```
 {%
   \@for\@gls@thisattrlabel:=#2\do{%
     \glssetcategoriesattribute{#1}{\@gls@thisattrlabel}{#3}%
   }%
 }%
}
```

tegoryattribute

\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}%
}
```

tegoryattribute

\glsunsetcategoryattribute{⟨*category*⟩}{⟨*attribute-label*⟩}

Unsets the given attribute for the given category.

```
\newcommand*{\glsunsetcategoryattribute}[2]{%
  \csundef{@glsxtr@categoryattr@@#1@#2}%
}
```

tegoryattribute

\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}%
}
```

tegoryattribute

\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 add the regular attribute.

```
\newcommand*{\glssetregularcategory}[1]{%
 \glssetcategoryattribute{#1}{regular}{true}%
}
```

regularcategory

```
\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}%
}
```

regularcategory

```
\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}%
}
```

reachincategory

```
\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}{}%
     }%
  }%
}
```

chwithattribute

```
\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

```
6786   \newcommand*{\glsxtrpostdescindex}{}
6787 }
6788 {}
```

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.

```
6789 \ifdef\printsymbols
6790 {%
```

glsxtrnewsymbol Unlike \newterm, this has a separate argument for the label (since the symbol will likely contain commands).

```
6791   \newcommand*{\glsxtrnewsymbol}[3][]{%
6792     \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
6793   }
```

Symbols are regular by default.

```
6794   \glssetcategoryattribute{symbol}{regular}{true}
```

rpostdescsymbol

```
6795   \newcommand*{\glsxtrpostdescsymbol}{}
6796 }
6797 {}
```

Similar for the numbers option.

```
6798 \ifdef\printnumbers
6799 {%
```

glsxtrnewnumber

```
6800 \ifdef\printnumbers
6801   \newcommand*{\glsxtrnewnumber}[3][]{%
6802     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
6803   }
```

Numbers are regular by default.

```
6804   \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
6805   \newcommand*{\glsxtrpostdescnumber}{}
6806 }
6807 {}
```

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}%
    }%
  }%
}
```

rfieldtitlecase

\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. Check for \glscapitalisewords, which was added to glossaries v4.48.

```
\ifdef\glscapitalisewords
{
 \newcommand*{\glsxtrfieldtitlecasecs}[1]{%
  \expandafter\glscapitalisewords\expandafter{#1}}
}
{
 \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}%
      {%
        \protected@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}%
```

```
      {%
        \protected@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}%
      {%
        \protected@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}%
      {%
        \protected@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}%
      {%
        \protected@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}%
```

```
      {%
        \protected@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.

```
      \protected@edef\@gls@thisval{#1}%
      \@for\@gls@map:=\@gls@keymap\do{%
       \protected@edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
       \ifdefequal{\@this@key}{\@gls@thisval}%
       {%
         \protected@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}%
   {%
     \protected@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}%
    {}%
    {\protected@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{}
```

oindex@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 format and accessibility support. Allow for the possibility of being used in a section heading for standalone entry definitions.

```
\ifdef\texorpdfstring
{
  \renewcommand*{\glossentrysymbol}[1]{%
    \texorpdfstring{\@glossentrysymbol{#1}}{\glsentrypdfsymbol{#1}}%
  }
```

```
}
{
  \renewcommand*{\glossentrysymbol}[1]{\@glossentrysymbol{#1}}
}
```

sentrypdfsymbol May be redefined to a field that expands to a value that's more suitable for PDF bookmarks.

```
\newcommand{\glsentrypdfsymbol}[1]{\glsentrysymbol{#1}}
```

lossentrysymbol There are no case-changing attributes as it's less usual for symbols.

```
\newrobustcmd*{\@glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \begingroup
      \glssetabbrvfmt{\glscategory{#1}}%
      \glshasattribute{#1}{glosssymbolfont}%
      {%
        \protected@edef\@glsxtr@attrval{\glsgetattribute{#1}{glosssymbolfont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glosssymbolfont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          '\@glsxtr@attrval' supplied in glosssymbolfont attribute
          for entry '#1'. Ignoring}%
          \let\@glsxtr@glosssymbolfont\@firstofone
        }%
      }%
      {\let\@glsxtr@glosssymbolfont\@firstofone}%
      \@glsxtr@glosssymbolfont{\glsaccesssymbol{#1}}%
    \endgroup
  }%
}
```

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}
```

lsxtrdopostpunc

```
\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, it needs to be applied by \newabbreviation.

```
\define@key{glsxtrabbrv}{category}{%
 \protected@edef\glscategorylabel{#1}%
}
```

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}%
```

Ignore the shortplural and longplural keys.

```
  \setkeys*{glsxtrabbrv}[shortplural,longplural]{#1}%
```

Set the abbreviation style.

```
  \ifcsdef{@glsabbrv@current@\glscategorylabel}%
  {%
```

Warning should already have been issued.

```
    \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
    \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
    \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@\glscategorylabel\endcsname}%
    \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
  }%
  {%
```

If no style has been associated with this category, fallback on the style for the abbreviation category.

```
    \glsxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
  }%
```

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}%

    \appto\@gls@short{\@}%
  }%
  {\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}%
```

Save in case required.

```
  \let\@gls@org@longpl\@gls@longpl
  \let\@gls@org@shortpl\@gls@shortpl
```

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@access
```

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
```

Obtain the type and add it to the list of abbreviations.

```
  \@glsxtr@addabbreviationlist{\glsentrytype{\the\glslabeltok}}%
  \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]{\glsabbrvdefaultfont{#1}}
```

\glsabbrvfont Font changing command used for the abbreviation on subsequent use. This is redefined by the abbreviation styles, as appropriate.

```
\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:

```
8201            \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
8202          }%
8203          {%
```

Subsequent singular form, all caps:

```
8204            \mfirstucMakeUppercase
8205              {\glsxtrsubsequentfmt{\glslabel}{\glsinsert}}%
8206          }%
8207        }%
8208      }%
8209      {%
```

First use:

```
8210        \glsifplural
8211        {%
```

First use plural form:

```
8212          \glscapscase
8213          {%
```

First use plural form, don't adjust case:

```
8214            \glsxtrfullplformat{\glslabel}{\glsinsert}%
8215          }%
8216          {%
```

First use plural form, make first letter upper case:

```
8217            \Glsxtrfullplformat{\glslabel}{\glsinsert}%
8218          }%
8219          {%
```

First use plural form, all caps:

```
8220            \mfirstucMakeUppercase
8221              {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
8222          }%
8223        }%
8224        {%
```

First use singular form

```
8225          \glscapscase
8226          {%
```

First use singular form, don't adjust case:

```
8227            \glsxtrfullformat{\glslabel}{\glsinsert}%
8228          }%
8229          {%
```

First use singular form, make first letter upper case:

```
8230            \Glsxtrfullformat{\glslabel}{\glsinsert}%
8231          }%
8232          {%
```

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}%
      \protected@edef\glscategorylabel{#1}%
      \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}%
}
```

cated@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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    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}%
    }%
    {}%
  }%
}%
{%
```

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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}%
```

trabbrvfootnote

\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}%
{%
```

Set accessibility attributes if enabled. (Add firstshortaccess since long form is hidden in a footnote on first use.)

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%

    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    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}
```

ootnotedescname

```
\newcommand*{\glsxtrfootnotedescname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
  \protect\glsxtrfullsep{\the\glslabeltok}%
  \protect\glsxtrparen{\protect\glslongfont{\the\glslongtok}}%
}
```

ootnotedescsort

```
\newcommand*{\glsxtrfootnotedescsort}{\the\glsshorttok}
```

t-footnote-desc Like short-footnote but with user supplied description.

```
\newabbreviationstyle{short-footnote-desc}%
{%
```

Set accessibility attributes if enabled

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotedescname},
    sort={\glsxtrfootnotedescsort},
    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}}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    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}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{footnote}%
}
```

footnote-desc Synonym.

```
\letabbreviationstyle{footnote-desc}{short-footnote-desc}
```

postfootnote Similar to footnote 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}%
{%
```

Set accessibility attributes if enabled. (Add firstshortaccess since long form is hidden in a footnote on first use.)

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    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}
```

stfootnote-desc Like short-postfootnote but with user supplied description.

```
\newabbreviationstyle{short-postfootnote-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotedescname},
    sort={\glsxtrfootnotedescsort},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    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
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{postfootnote}%
}
```

stfootnote-desc

```
\letabbreviationstyle{postfootnote-desc}{short-postfootnote-desc}
```

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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
  \protect\glsxtrfullsep{\the\glslabeltok}%
  \protect\glsxtrparen{\protect\glslongfont{\the\glslongtok}}%
}
```

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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}}%
  \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. The accessibility attributes don't need setting here.

```
\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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 \protect needs to come inside \s to avoid interfering with all caps.

```
\newcommand*{\glsxtrscsuffix}{\protect\glstextup{\glsxtrabbrvpluralsuffix}}
```

long-short-sc

```
\newabbreviationstyle{long-short-sc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
}
```

short-sc-long Now the short (long) version

```
\newabbreviationstyle{short-sc-long}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    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}%
    }%
    {}%
  }%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\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}}}%
  }%
}
```

rt-sc-long-desc As before but user provides description

```
\newabbreviationstyle{short-sc-long-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}}%
```

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\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`. No accessibility attributes needed here.

```
\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}{\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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    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}{\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}
```

`c-footnote-desc` Like short-sc-footnote but with user supplied description.

```
\newabbreviationstyle{short-sc-footnote-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotedescname},
    sort={\glsxtrfootnotedescsort},
    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}}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    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}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-sc-footnote}%
}
```

sc-postfootnote

```
\newabbreviationstyle{short-sc-postfootnote}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    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}{\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}
```

stfootnote-desc Like short-sc-footnote but with user supplied description.

```
\newabbreviationstyle{short-sc-postfootnote-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotedescname},
    sort={\glsxtrfootnotedescsort},
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    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
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
    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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
\glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
\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}}},%
  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}%
    }%
    {}%
  }%
}%
{%

  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
\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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%

  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}{\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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
\glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
\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}}},%
  text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
  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}{\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}
```

`m-footnote-desc` Like short-footnote but with user supplied description.

```
\newabbreviationstyle{short-sm-footnote-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotedescname},
    sort={\glsxtrfootnotedescsort},
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
```

```
10285      \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
10286        {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
10287    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
10288      \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
10289        {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
10290    text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
10291    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.

```
10292  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10293    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
10294    \glshasattribute{\the\glslabeltok}{regular}%
10295    {%
10296      \glssetattribute{\the\glslabeltok}{regular}{false}%
10297    }%
10298    {}%
10299  }%
10300 }%
10301 {%
10302  \GlsXtrUseAbbrStyleFmts{short-sm-footnote}%
10303 }
```

sm-postfootnote

```
10304 \newabbreviationstyle{short-sm-postfootnote}%
10305 {%
```

Set accessibility attributes if enabled.

```
10306  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
10307  \renewcommand*{\CustomAbbreviationFields}{%
10308    name={\glsxtrfootnotename},
10309    sort={\the\glsshorttok},
10310    description={\the\glslongtok},%
10311    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
10312    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
10313    text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
10314    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.

```
10315  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10316    \csdef{glsxtrpostlink\glscategorylabel}{%
10317      \glsxtrifwasfirstuse
10318      {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
10319        \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
10320        {\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}{\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}
```

stfootnote-desc Like short-sm-postfootnote but with user supplied description.

```
\newabbreviationstyle{short-sm-postfootnote-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotedescname},
    sort={\glsxtrfootnotedescsort},
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
    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
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
\glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields. `\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}}},%
```

```
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    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}%
    }%
    {}%
  }%
}%
{%
```

Mostly as short-long style:

```
  \renewcommand*{\abbrvpluralsuffix}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields. `\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}}},%
    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}%
```

```
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}{\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}%
{%
```

Set accessibility attributes if enabled. The default name includes the long form but \glsxtrshortdescname could be modified to omit the long form, so include the nameshortaccess attribute.

```
\glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
\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}}}%
\renewcommand*{\GlsXtrPostNewAbbreviation}{%
  \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%

\renewcommand*{\abbrvpluralsuffix}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}{\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 `\glsxtrshort`. The long form is emphasized. No accessibility attributes need to be set.

```
\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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    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}{\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}
```

`m-footnote-desc` Like short-em-footnote but with user supplied description.

```
\newabbreviationstyle{short-em-footnote-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotedescname},
    sort={\glsxtrfootnotedescsort},
    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}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    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}%
    }%
    {}%
```

```
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-em-footnote}%
}
```

em-postfootnote

```
\newabbreviationstyle{short-em-postfootnote}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    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}{\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}
```

stfootnote-desc Like short-em-postfootnote but with user supplied description.

```
\newabbreviationstyle{short-em-postfootnote-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotedescname},
    sort={\glsxtrfootnotedescsort},
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    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
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}},%
    text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
    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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
    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}%
  }%
}
```

Small-caps is awkward, so support for that is added.

abbrvscuserfont

```
\newcommand*{\glsabbrvscuserfont}{\glsabbrvscfont}%
```

abbrvscuserfont

```
\newcommand*{\glsfirstabbrvscuserfont}{\glsabbrvscuserfont}%
```

The default short form suffix:

xtrscusersuffix

```
\newcommand*{\glsxtrscusersuffix}{\glsxtrscsuffix}
```

lsxtrscusername The default name format for this style.

```
\newcommand*{\glsxtrlongshortscusername}{%
  \protect\glsabbrvscuserfont{\the\glsshorttok}%
}
```

stshort-sc-user Like long-short-sc-user but uses smallcaps.

```
\newabbreviationstyle{long-postshort-sc-user}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortscusername},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    text={\protect\glsabbrvscuserfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvscuserfont{\the\glsshortpltok}},%
    description={\protect\glsuserdescription{\the\glslongtok}%
     {\the\glslabeltok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvscuserfont{\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}{\glsxtrscusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscuserfont{##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{\glsfirstabbrvscuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvscuserfont{\glsaccessshortpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvscuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvscuserfont{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
}
```

tscuserdescname

```
\newcommand*{\glsxtrlongshortscuserdescname}{%
  \protect\glslonguserfont{\the\glslongtok}%
  \protect\glsxtruserparen
   {\protect\glsabbrvscuserfont{\the\glsshorttok}}{\the\glslabeltok}%
}
```

rt-sc-user-desc Like long-postshort-sc-user but the user supplies the description.

```
\newabbreviationstyle{long-postshort-sc-user-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortscuserdescname},
    sort={\the\glslongtok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    text={\protect\glsabbrvscuserfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvscuserfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvscuserfont{\glsentryshort{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-postshort-sc-user}%
}
```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```
\newabbreviationstyle{short-postlong-user}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
```

```
    text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
    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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

\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}},%
    text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
    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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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
}
```

longhyphenshort

\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
}
```

nghyphennoshort

\glsxtrlonghyphennoshort{⟨*label*⟩}{⟨*long*⟩}{⟨*insert*⟩}

```
12000 \newcommand*{\glsxtrlonghyphennoshort}[3]{%
```

Grouping is needed to localise the redefinitions.

```
12001  {%
```

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.

```
12002    \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
12003    \glsfirstlonghyphenfont{#2\ifglsxtrinsertinside{#3}\fi}%
12004    \ifglsxtrinsertinside\else{#3}\fi
12005  }%
12006 }
```

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. No accessibility attributes need to be set.

```
12007 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
12008 {%
12009   \renewcommand*{\CustomAbbreviationFields}{%
12010     name={\glsxtrlongnoshortdescname},
12011     sort={\expandonce\glsxtrorglong},
12012     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
12013     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
12014     text={\protect\glslonghyphenfont{\the\glslongtok}},%
12015     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
12016   }%
```

Unset the regular attribute if it has been set.

```
12017   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
12018     \glshasattribute{\the\glslabeltok}{regular}%
12019     {%
12020       \glssetattribute{\the\glslabeltok}{regular}{false}%
12021     }%
12022     {}%
12023   }%
12024 }%
12025 {%
12026   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
```

In case the user wants to mix and match font styles, these are redefined here.

```
12027   \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
12028   \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
12029   \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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-hyphen-noshort-desc-noreg}%
}
```

lsxtrlonghyphen

\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}%
```

```
 }%
}
```

posthyphenshort

\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
   }%
 }%
}
```

yphensubsequent

\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
```

```
    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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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
      {%
```

```
12215         \glsxtrposthyphenshort{\glslabel}{\glsinsert}%
12216       }%
12217       {%
```

Put the insertion into the post-link:

```
12218         \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
12219       }%
12220     }%
12221     \glshasattribute{\the\glslabeltok}{regular}%
12222     {%
12223       \glssetattribute{\the\glslabeltok}{regular}{false}%
12224     }%
12225     {}%
12226   }%
12227 }%
12228 {%
12229   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
12230 }
```

shorthyphenlong

\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.

```
12231 \newcommand*{\glsxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
12232  {%
```

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.)

```
12233    \glsxtrifhyphenstart{#4}{\def\glsxtrwordsep{-}}{}%
12234    \glsfirstabbrvhyphenfont{#2\ifglsxtrinsertinside{#4}\fi}%
12235    \ifglsxtrinsertinside\else{#4}\fi
12236    \glsxtrfullsep{#1}%
12237    \glsxtrparen{\glsfirstlonghyphenfont{#3\ifglsxtrinsertinside{#4}\fi}%
12238     \ifglsxtrinsertinside\else{#4}\fi}%
12239  }%
12240 }
```

hen-long-hyphen Designed for use with the markwords attribute.

```
12241 \newabbreviationstyle{short-hyphen-long-hyphen}%
12242 {%
```

Set accessibility attributes if enabled.

```
12243   \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    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}%
{%
```

Set accessibility attributes if enabled.

```
\glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
\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}%
}
```

sxtrshorthyphen

\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}%
 }%
}
```

rposthyphenlong

\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtronlyname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
    text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
    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}{\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}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
  \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}%
}
```

Small-caps is awkward, so support for that is added.

abbrvsconlyfont

```
\newcommand*{\glsabbrvsconlyfont}{\glsabbrvscfont}%
```

abbrvsconlyfont

```
\newcommand*{\glsfirstabbrvsconlyfont}{\glsabbrvsconlyfont}%
```

The default short form suffix:

xtrsconlysuffix

```
\newcommand*{\glsxtrsconlysuffix}{\glsxtrscsuffix}
```

lsxtrsconlyname The default name format for this style.

```
\newcommand*{\glsxtrsconlyname}{%
  \protect\glsabbrvsconlyfont{\the\glsshorttok}%
}
```

y-short-sc-only

```
\newabbreviationstyle{long-only-short-sc-only}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrsconlyname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
    text={\protect\glsabbrvsconlyfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvsconlyfont{\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}{\glsxtrsconlysuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsconlyfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsconlyfont{##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\glsfirstabbrvsconlyfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsconlyfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsconlyfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsconlyfont{\Glsaccessshortpl{##1}}}%
  }%
}
```

rsconlydescsort

```
\newcommand*{\glsxtrsconlydescsort}{\glsxtronlydescsort}
```

rsconlydescname

```
\newcommand*{\glsxtrsconlydescname}{\glsxtronlydescname}
```

rt-sc-only-desc

```
\newabbreviationstyle{long-only-short-sc-only-desc}%
{%
```

Set accessibility attributes if enabled.

```
  \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
\renewcommand*{\CustomAbbreviationFields}{%
  name={\glsxtrsconlydescname},
  sort={\glsxtrsconlydescsort},%
  first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
  firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
  text={\protect\glsabbrvsconlyfont{\the\glsshorttok}},%
  plural={\protect\glsabbrvsconlyfont{\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-sc-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}[]%
}
```

LSxtrtitleshort Command to display short form of abbreviation in section title and table of contents in all 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}[]%
}
```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents in all 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.

```
\newrobustcmd*{\Glsxtrtitlename}[1]{%
```

```
  \Glsname[noindex,hyper=false]{#1}[]%
}
```

`GLSxtrtitlename` Command to display name value in section title and table of contents in all upper case.

```
\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}[]%
}
```

`GLSxtrtitletext` Command to display text value in section title and table of contents in all 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}[]%
}
```

`Sxtrtitleplural` Command to display plural value in section title and table of contents in all 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}[]%
}
```

LSxtrtitlefirst Command to display first value in section title and table of contents in all 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}[]%
}
```

itlefirstplural Command to display first value in section title and table of contents in all 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}[]%
}
```

GLSxtrtitlelong Command to display long form of abbreviation in section title and table of contents in all 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}[]%
}
```

Sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents in all 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}[]%
}
```

GLSxtrtitlefull Command to display full form of abbreviation in section title and table of contents in all 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}[]%
}
```

Sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents in all 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}}
}
```

`\GLSfmtname` All 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}}
}
```

`\GLSfmttext` All 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}}
}
```

\GLSfmtplural All 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}}%
  }
}
```

```
13295 {
13296   \newcommand*{\Glsfmtfirst}[1]{%
13297    \Glsxtrtitlefirst{#1}}
13298 }
```

\GLSfmtfirst All upper case.

```
13299 \ifdef\texorpdfstring
13300 {
13301   \newcommand*{\GLSfmtfirst}[1]{%
13302     \texorpdfstring
13303     {\GLSxtrtitlefirst{#1}}%
13304     {\glsentryfirst{#1}}%
13305   }
13306 }
13307 {
13308   \newcommand*{\GLSfmtfirst}[1]{%
13309    \Glsxtrtitlefirst{#1}}
13310 }
```

\glsfmtfirstpl As above but for the firstplural value.

```
13311 \ifdef\texorpdfstring
13312 {
13313   \newcommand*{\glsfmtfirstpl}[1]{%
13314     \texorpdfstring
13315     {\glsxtrtitlefirstplural{#1}}%
13316     {\glsentryfirstplural{#1}}%
13317   }
13318 }
13319 {
13320   \newcommand*{\glsfmtfirstpl}[1]{%
13321    \glsxtrtitlefirstplural{#1}}
13322 }
```

\Glsfmtfirstpl First letter converted to upper case.

```
13323 \ifdef\texorpdfstring
13324 {
13325   \newcommand*{\Glsfmtfirstpl}[1]{%
13326     \texorpdfstring
13327     {\Glsxtrtitlefirstplural{#1}}%
13328     {\glsentryfirstplural{#1}}%
13329   }
13330 }
13331 {
13332   \newcommand*{\Glsfmtfirstpl}[1]{%
13333    \Glsxtrtitlefirstplural{#1}}
13334 }
```

\GLSfmtfirstpl All upper case.

```
13335 \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}}
}
```

\GLSfmtlong All 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}}
}
```

\GLSfmtlongpl All upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\GLSfmtlongpl}[1]{%
    \texorpdfstring
    {\GLSxtrtitlelongpl{#1}}%
    {\glsentrylongpl{#1}}%
  }
}
{
  \newcommand*{\GLSfmtlongpl}[1]{%
   \GLSxtrtitlelongpl{#1}}
}
```

\glspdffmtfull Can't use \glsxtrinlinefullformat in PDF bookmarks as it's not fully expandable. This command is for the PDF part of \texorpdfstring for the full form.

```
\newcommand*{\glspdffmtfull}[1]{\glsentrylong{#1} (\glsentryshort{#1})}%
```

`glspdffmtfullpl` Likewise for plural.

```
\newcommand*{\glspdffmtfullpl}[1]{\glsentrylongpl{#1} (\glsentryshortpl{#1})}%
```

`\glsfmtfull` In-line full format.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfull}[1]{%
    \texorpdfstring
    {\glsxtrtitlefull{#1}}%
    {\glspdffmtfull{#1}}%
  }
}
{
  \newcommand*{\glsfmtfull}[1]{%
   \glsxtrtitlefull{#1}}
}
```

`\Glsfmtfull` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfull}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefull{#1}}%
    {\glspdffmtfull{#1}{}}%
  }
}
{
  \newcommand*{\Glsfmtfull}[1]{%
   \Glsxtrtitlefull{#1}}
}
```

`\GLSfmtfull` All upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\GLSfmtfull}[1]{%
    \texorpdfstring
    {\GLSxtrtitlefull{#1}}%
    {\glspdffmtfull{#1}}%
  }
}
{
  \newcommand*{\GLSfmtfull}[1]{%
   \GLSxtrtitlefull{#1}}
}
```

`\glsfmtfullpl` In-line full plural format.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfullpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlefullpl{#1}}%
    {\glspdffmtfullpl{#1}}%
  }
}
{
  \newcommand*{\glsfmtfullpl}[1]{%
   \glsxtrtitlefullpl{#1}}
}
```

`\Glsfmtfullpl` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfullpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefullpl{#1}}%
    {\glspdffmtfullpl{#1}{}}%
  }
}
{
  \newcommand*{\Glsfmtfullpl}[1]{%
   \Glsxtrtitlefullpl{#1}}
}
```

`\GLSfmtfullpl` All upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\GLSfmtfullpl}[1]{%
    \texorpdfstring
    {\GLSxtrtitlefullpl{#1}}%
    {\glspdffmtfullpl{#1}{}}%
  }
}
{
  \newcommand*{\GLSfmtfullpl}[1]{%
   \GLSxtrtitlefullpl{#1}}
}
```

1.9 Multi (Combined/Compound) Entries

(I'd rather call these combined or compound entries but `\cgls` is already taken.)

New to version 1.48, the commands here provide a way of referencing multiple entries as a single unit. For example, biological organisms are often referred to by their genus and species, such as *Clostridium botulinum* and *Clostridium perfringens* (where the genus is Clostridium).

The genus is often abbreviated after first use, regardless of which species in the genus is being referenced. For example, "*Clostridium botulinum* and *C. perfringens*". This can't be supported by any abbreviation styles unless the genus and species names are defined separately. For example:

```
\setabbreviationstyle{long-only-short-only}
\newabbreviation{clostridium}{C.}{Clostridium}
\newglossaryentry{botulinum}{name={botulinum},description={}}
\newglossaryentry{perfringens}{name={perfringens},description={}}
```

This means that the entries then need to be referenced using a rather cumbersome method:

```
\gls{clostridium} \gls{botulinum} and \gls{clostridum}
\gls{perfringens}
```

This section provides a command that will provide a way of defining a label that represents a combination of entries (which must all be first defined). For example:

```
\multiglossaryentry{cbot}{clostridium,botulinum}
```

This label can then be referenced using `\mgls`, which internally uses `\gls` for each component. The last component in the list is considered the "main" component (not to be confused with the main glossary). If this isn't the case, the label of the main component should be added in the optional argument before the label list. Note that the multi-label (`cbot` in this case) can't be referenced using commands like `\gls`.

First define the general set of options that should be applied to all multi-entries. These can be set with:

ssaryentrysetup

```
13493 \newcommand*{\multiglossaryentrysetup}[1]{\setkeys{glsxtrcombined}{#1}}
```

bined@indexmain Numeric value: 0=false (don't index main component), 1=true (always index main component), 2=first (only index main component on first use). Default: 1 (true);

```
13494 \newcommand*{\@gls@combined@indexmain}{1}
13495 \define@choicekey{glsxtrcombined}{indexmain}%
13496  [\@gls@combined@indexmain@val\@gls@combined@indexmain]
13497  {false,true,first}[true]{}
```

ned@indexothers Numeric value: 0=false (don't index other components), 1=true (always index other components), 2=first (only index other components on first use). Default: 2 (first);

```
13498 \newcommand*{\@gls@combined@indexothers}{2}
13499 \define@choicekey{glsxtrcombined}{indexothers}%
13500  [\@gls@combined@indexothers@val\@gls@combined@indexothers]
13501  {false,true,first}[true]{}
```

@combined@hyper Numeric value: 0=none (`\mgls` doesn't create a hyperlink), 1=allmain (all content hyperlinks to the main component), 2=mainonly (only the main component has a hyperlink), 3=individual (each component has a hyperlink to their own target). Default: 3.

```
\newcommand*{\@gls@combined@hyper}{3}
\define@choicekey{glsxtrcombined}{hyper}%
 [\@gls@combined@hyper@val\@gls@combined@hyper]
 {none,allmain,mainonly,individual,otheronly,notmainfirst,nototherfirst,notfirst}{}
```

bined@encapmain Location encap value for main component (corresponding to format key in \gls).

```
\newcommand*{\@gls@combined@encapmain}{glsnumberformat}
\define@key{glsxtrcombined}{encapmain}{%
 \renewcommand*{\@gls@combined@encapmain}{#1}%
}
```

ned@encapothers Location encap value for other components (corresponding to format key in \gls).

```
\newcommand*{\@gls@combined@encapothers}{glsnumberformat}
\define@key{glsxtrcombined}{encapothers}{%
 \renewcommand*{\@gls@combined@encapothers}{#1}%
}
```

ined@textformat Encapsulate entire content with the command identified by the given control sequence name.

```
\newcommand*{\@gls@combined@textformat}{@firstofone}
\define@key{glsxtrcombined}{textformat}{%
 \renewcommand*{\@gls@combined@textformat}{#1}%
}
```

mbined@category Assign a category to the combined set.

```
\newcommand*{\@gls@combined@category}{}
\define@key{glsxtrcombined}{category}{%
 \renewcommand*{\@gls@combined@category}{#1}%
}
```

Pre-options family:

```
\define@key{glsxtrcombinedpreopts}{category}{%
 \renewcommand*{\@gls@combined@category}{#1}%
}
```

mbined@mglsopts Default options to pass to \mgls.

```
\newcommand*{\@gls@combined@mglsopts}{}
\define@key{glsxtrcombined}{mglsopts}{%
 \renewcommand*{\@gls@combined@mglsopts}{#1}%
}
\define@key{glsxtrcombinedpreopts}{mglsopts}{%
 \@gls@combined@mglsopts@do
 {%
   \renewcommand*{\@gls@combined@mglsopts}{#1}%
 }%
}
```

ned@mglsopts@do

```
\newcommand*{\@gls@combined@mglsopts@do}[1]{#1}
```

isable@mglsopts

```
\newcommand*{\mgls@disable@mglsopts}{%
 \let\@gls@combined@mglsopts@do\@gls@combined@mglsopts@do@not
}
```

enable@mglsopts

```
\newcommand*{\mgls@enable@mglsopts}{%
 \let\@gls@combined@mglsopts@do\@firstofone
}
```

ned@mglsopts@do

```
\newcommand*{\@gls@combined@mglsopts@do@not}[1]{%
  \PackageError{glossaries-extra}{`mglsopts' key not permitted inside
   `setup' value}{}%
}
```

ned@firstprefix Prefix for multi-entry first use.

```
\newcommand*{\@gls@combined@firstprefix}{}
\define@key{glsxtrcombined}{firstprefix}{%
 \renewcommand*{\@gls@combined@firstprefix}{#1}%
}
```

ined@usedprefix Prefix for multi-entry subsequent first use.

```
\newcommand*{\@gls@combined@usedprefix}{}
\define@key{glsxtrcombined}{usedprefix}{%
 \renewcommand*{\@gls@combined@usedprefix}{#1}%
}
```

ned@firstsuffix Suffix for multi-entry first use.

```
\newcommand*{\@gls@combined@firstsuffix}{}
\define@key{glsxtrcombined}{firstsuffix}{%
 \renewcommand*{\@gls@combined@firstsuffix}{#1}%
}
```

ined@usedsuffix Suffix for multi-entry subsequent first use.

```
\newcommand*{\@gls@combined@usedsuffix}{}
\define@key{glsxtrcombined}{usedsuffix}{%
 \renewcommand*{\@gls@combined@usedsuffix}{#1}%
}
```

d@firstskipmain Skip the main element on first use (multi-entry first use not element first use).

```
\define@boolkey{glsxtrcombined}{firstskipmain}[true]{}
\KV@glsxtrcombined@firstskipmainfalse
```

firstskipothers Skip the other elements on first use (multi-entry first use not element first use).

```
\define@boolkey{glsxtrcombined}{firstskipothers}[true]{}
\KV@glsxtrcombined@firstskipothersfalse
```

ed@usedskipmain Skip the main element on subsequent use (multi-entry subsequent use not element subsequent use).

```
\define@boolkey{glsxtrcombined}{usedskipmain}[true]{}
\KV@glsxtrcombined@usedskipmainfalse
```

@usedskipothers Skip the other elements on subsequent use (multi-entry subsequent use not element subsequent use).

```
\define@boolkey{glsxtrcombined}{usedskipothers}[true]{}
\KV@glsxtrcombined@usedskipothersfalse
```

bined@postlinks Determine whether or not to use the individual element post-link hooks.

```
\newcommand*{\@gls@combined@postlinks@nr}{0}
\define@choicekey{glsxtrcombined}{postlinks}%
 [\@gls@combined@postlinks@val\@gls@combined@postlinks@nr]
 {none,all,notlast,mainnotlast,mainonly,othernotlast,otheronly}{}
```

bined@mpostlink Determine whether or not to use the multi-entry post-link hook.

```
\newcommand*{\@gls@combined@mpostlink@nr}{1}
\define@choicekey{glsxtrcombined}{mpostlink}%
 [\@gls@combined@mpostlink@val\@gls@combined@mpostlink@nr]
 {false,true,firstonly,usedonly}[true]{}
```

postlinkelement Determine which element to use for the post-link hook.

```
\newcommand*{\@gls@combined@mpostlinkelement@nr}{0}
\define@choicekey{glsxtrcombined}{mpostlinkelement}%
 [\@gls@combined@mpostlinkelement@val\@gls@combined@mpostlinkelement@nr]
 {last,main,custom}{}
```

\glsxtrifmulti

```
\newcommand*{\glsxtrifmulti}[3]{\ifcsdef{@gls@combined@#1@main}{#2}{#3}}
```

glsxtrmultimain

```
\newcommand*{\glsxtrmultimain}[1]{\csuse{@gls@combined@#1@main}}
```

glsxtrmultilist

```
\newcommand*{\glsxtrmultilist}[1]{\csuse{@gls@combined@#1@list}}
```

titotalelements Total number of elements.

```
\newcommand*{\glsxtrmultitotalelements}[1]{\csuse{@gls@combined@#1@total}}
```

rmultimainindex Index of main element (starting from 1). If the main element is the last element in the list then this should equal the total number of elements.

```
\newcommand*{\glsxtrmultimainindex}[1]{\csuse{@gls@combined@#1@mainindex}}
```

ilastotherindex Index of the last non-main element.

```
\newcommand*{\glsxtrmultilastotherindex}[1]{\csuse{@gls@combined@#1@lastotherindex}}
```

saryentryglobal Make definitions global.

```
\newif\ifmultiglossaryentryglobal
\multiglossaryentryglobalfalse
```

glselementindex Count register to keep track of the current element index.

```
\newcount\mglselementindex
```

tiglossaryentry

\multiglossaryentry[⟨*options*⟩]{⟨*multi-label*⟩}[⟨*main label*⟩]{⟨*label list*⟩}

Defines the label ⟨*multi-label*⟩ that can be used in \mgls.

```
\newrobustcmd{\multiglossaryentry}[1][]{%
 \def\@gls@combined@current@opts{#1}%
 \ifnum\@glsxtr@docdefval=1\relax
  \let\@multi@glossentry@donext\@defmultiglossaryentry
 \else
  \let\@multi@glossentry@donext\@multiglossaryentry
 \fi
 \@multi@glossentry@donext
}
```

tiglossaryentry

```
\newcommand*{\@multiglossaryentry}[1]{%
  \def\@gls@combined@current@label{#1}%
  \@multi@glossaryentry
}
```

i@glossaryentry Check for existence.

```
\newcommand*{\@multi@glossaryentry}[2][]{%
  \ifcsdef{@gls@combined@\@gls@combined@current@label @main}%
  {\PackageError{glossaries-extra}%
    {Multi-entry label `\@gls@combined@current@label' already defined}%
    {}%
  }%
  {%
    \@multi@glossary@entry{#1}{#2}%
  }%
}
```

tiglossaryentry Used if document definitions are on.

```
\newcommand*{\@defmultiglossaryentry}[1]{%
  \def\@gls@combined@current@label{#1}%
  \@def@multi@glossaryentry
}
```

i@glossaryentry Used if document definitions are on.

```
\newcommand*{\@def@multi@glossaryentry}[2][]{%
  \let\@def@multi@glossaryentry@do\@multi@glossary@entry
  \ifundef\@glsxtr@docdefs@multilist
  {%
    \gdef\@glsxtr@docdefs@multilist{}%
    \listxadd
     {\@glsxtr@docdefs@multilist}{\expandonce\@gls@combined@current@label}%
  }%
  {%
    \xifinlist{\@gls@combined@current@label}{\@glsxtr@docdefs@multilist}%
    {%
      \PackageError{glossaries-extra}%
      {Multi-entry label `\@gls@combined@current@label' already defined}%
      {}%
      \let\@def@multi@glossaryentry@do\@gobbletwo
    }%
    {%
      \listxadd
       {\@glsxtr@docdefs@multilist}{\expandonce\@gls@combined@current@label}%
    }%
  }%
  \@def@multi@glossaryentry@do{#1}{#2}%
}
```

sary@doifexists

```
\newcommand*{\@multi@glossary@doifexists}{\glsdoifexists}
```

tiglossaryentry

\providemultiglossaryentry[⟨*options*⟩]{⟨*multi-label*⟩}[⟨*main label*⟩]{⟨*label list*⟩}

Defines a multi-entry unless it has already been defined.

```
\newrobustcmd{\providemultiglossaryentry}[2][]{%
 \def\@gls@combined@current@opts{#1}%
 \def\@gls@combined@current@label{#2}%
 \ifcsdef{@gls@combined@\@gls@combined@current@label @main}%
 {\def\@multi@glossentry@donext{\@provide@multi@glossaryentry@noop}}%
 {%
   \ifnum\@glsxtr@docdefval=1\relax
    \def\@multi@glossentry@donext{\@def@multi@glossaryentry}%
   \else
    \def\@multi@glossentry@donext{\@multi@glossaryentry}%
   \fi
 }%
 \@multi@glossentry@donext
}
```

ssaryentry@noop Do nothing.

```
\newcommand*{\@provide@multi@glossaryentry@noop}[2][]{}
```

ssaryentry@list List of all defined multi-entry sets.

```
\newcommand*{\@multi@glossaryentry@list}{}
```

@glossary@entry

```
\newcommand*{\@multi@glossary@entry}[2]{%
  \protected@edef\@gls@combined@current@main{#1}%
```

Fully expand list.

```
  \protected@edef\@gls@combined@currentlist{#2}%
```

Count items in list, check they are all defined, and find last item at the same time.

```
  \mglselementindex=0\relax
  \@for\@gls@tmp:=\@gls@combined@currentlist\do{%
    \advance\mglselementindex by 1\relax
    \@multi@glossary@doifexists{\@gls@tmp}{}%
    \let\@gls@combined@finalitem\@gls@tmp
    \ifdefvoid\@gls@combined@current@main
    {}%
    {%
      \ifx\@gls@combined@current@main\@gls@tmp
        \ifmultiglossaryentryglobal
          \global\cslet{@gls@combined@\@gls@combined@current@label @main}%
                \@gls@combined@current@main
          \csxdef{@gls@combined@\@gls@combined@current@label @mainindex}%
                 {\the\mglselementindex}%
        \else
          \cslet{@gls@combined@\@gls@combined@current@label @main}%
                \@gls@combined@current@main
          \csedef{@gls@combined@\@gls@combined@current@label @mainindex}%
                 {\the\mglselementindex}%
        \fi
      \else
        \ifmultiglossaryentryglobal
          \csxdef{@gls@combined@\@gls@combined@current@label @lastotherindex}%
                 {\the\mglselementindex}%
        \else
          \csedef{@gls@combined@\@gls@combined@current@label @lastotherindex}%
                 {\the\mglselementindex}%
        \fi
      \fi
    }%
  }%
  \ifmultiglossaryentryglobal
    \csxdef{@gls@combined@\@gls@combined@current@label @total}%
           {\the\mglselementindex}%
  \else
    \csedef{@gls@combined@\@gls@combined@current@label @total}%
```

```
          {\the\mglselementindex}%
  \fi
  \ifnum\mglselementindex<2\relax
     \PackageError{glossaries-extra}{At least 2 labels required in
       multi-entry element list (\number\mglselementindex\space found)}{}%
  \else
    \ifdefvoid\@gls@combined@current@main
    {}%
    {%
```

If `\@gls@combined@⟨label⟩@main` hasn't been set then it wasn't included in the list.

```
      \ifcsundef{@gls@combined@\@gls@combined@current@label @main}%
      {\PackageError{glossaries-extra}%
       {Main element `\@gls@combined@current@main' not found in list}%
       {The final element `\@gls@combined@finalitem' will be used instead}
```

Set to empty so that the default (final element) is used instead.

```
       \let\@gls@combined@current@main\@empty
      }%
      {}%
    }%
    \ifdefvoid\@gls@combined@current@main
    {%
```

Set main to final element.

```
      \ifmultiglossaryentryglobal
        \global\cslet{@gls@combined@\@gls@combined@current@label @main}%
              \@gls@combined@finalitem
        \global\csletcs{@gls@combined@\@gls@combined@current@label @mainindex}%
          {@gls@combined@\@gls@combined@current@label @total}%
        \csxdef{@gls@combined@\@gls@combined@current@label @lastotherindex}%
               {\the\numexpr\mglselementindex-1 }%
      \else
        \cslet{@gls@combined@\@gls@combined@current@label @main}%
              \@gls@combined@finalitem
        \csletcs{@gls@combined@\@gls@combined@current@label @mainindex}%
          {@gls@combined@\@gls@combined@current@label @total}%
        \csedef{@gls@combined@\@gls@combined@current@label @lastotherindex}%
               {\the\numexpr\mglselementindex-1 }%
      \fi
    }%
    {}%
    \ifmultiglossaryentryglobal
```

Globally define element list.

```
      \global\cslet{@gls@combined@\@gls@combined@current@label @list}%
            \@gls@combined@currentlist
```

Globally define options.

```
      \protected@csxdef{@gls@combined@\@gls@combined@current@label @options}%
       {\@gls@combined@current@opts}%
```

Global conditional definition.

```
      \expandafter\@ifdefinable
       \csname if@gls@combined@\@gls@combined@current@label @flag\endcsname
       {\expandafter\global\expandafter
        \newif\csname if@gls@combined@\@gls@combined@current@label @flag\endcsname}%
      \expandafter\global
       \csname @gls@combined@\@gls@combined@current@label @flagfalse\endcsname
    \else
```

Locally define element list.

```
      \cslet{@gls@combined@\@gls@combined@current@label @list}%
            \@gls@combined@currentlist
```

Locally define options.

```
      \protected@csedef{@gls@combined@\@gls@combined@current@label @options}%
       {\@gls@combined@current@opts}%
```

Local conditional definition.

```
      \newboolean{@gls@combined@\@gls@combined@current@label @flag}%
      \csname @gls@combined@\@gls@combined@current@label @flagfalse\endcsname
    \fi
  \fi
  \writemultiglossentry
   {\@gls@combined@current@opts}{\@gls@combined@current@label}%
   {\csuse{@gls@combined@\@gls@combined@current@label @main}}{#2}%
```

Append label to list.

```
  \ifmultiglossaryentryglobal
    \ifdefempty\@multi@glossaryentry@list
    {\let\@multi@glossaryentry@list\@gls@combined@current@label}%
    {%
      \eappto\@multi@glossaryentry@list{,\expandonce\@gls@combined@current@label}%
    }%
  \else
    \ifdefempty\@multi@glossaryentry@list
    {\global\let\@multi@glossaryentry@list\@gls@combined@current@label}%
    {%
      \xappto\@multi@glossaryentry@list{,\expandonce\@gls@combined@current@label}%
    }%
  \fi
}
```

sxtr@multientry

\@glsxtr@multientry{⟨*options*⟩}{⟨*multilabel*⟩}{⟨*main*⟩}{⟨*list*⟩}

Information for aux file. Useful for bib2gls and also for docdef.

```
\newcommand*{\@glsxtr@multientry}[4]{%
 \ifnum\@glsxtr@docdefval=1\relax
```

```
  \bgroup
    \def\@gls@combined@current@opts{#1}%
    \def\@gls@combined@current@label{#2}%
    \let\@multi@glossary@doifexists\@secondoftwo
    \let\writemultiglossentry\@gobblefour
    \multiglossaryentryglobaltrue
    \@multi@glossary@entry{#3}{#4}%
  \egroup
 \fi
}
```

`multiglossentry` This can be redefined to do nothing if the information isn't required.

```
\newcommand*{\writemultiglossentry}[4]{%
  \protected@write\@auxout{}{\string\@glsxtr@multientry{#1}{#2}{#3}{#4}}%
}
```

`\ifmglsused` Determines whether or not the multi-entry set has been referenced by commands like `\mgls` or `\mglsname`.

```
\newcommand*{\ifmglsused}[3]{%
 \ifbool{@gls@combined@#1@flag}{#2}{#3}%
}
```

`\mglsunset` Unset the flag.

```
\newcommand*{\mglsunset}[1]{%
 \gls@ifnotmeasuring
 {%
    \glsxtrifmulti{#1}{\@mglsunset{#1}}%
    {%
      \glsxtrundefaction{Multi entry `#1' hasn't been defined}%
      {You need to define `#1' with \string\multiglossaryentry}%
    }%
 }%
}
```

`\@mglsunset`

```
\newcommand*{\@mglsunset}[1]{%
 \expandafter\global\csname @gls@combined@#1@flagtrue\endcsname
}
```

`\mglsreset` Unset the flag.

```
\newcommand*{\mglsreset}[1]{%
 \gls@ifnotmeasuring
 {%
    \glsxtrifmulti{#1}{\@mglsreset{#1}}%
    {%
      \glsxtrundefaction{Multi entry `#1' hasn't been defined}%
      {You need to define `#1' with \string\multiglossaryentry}%
    }%
```

```
 }%
}
```

\@mglsreset

```
\newcommand*{\@mglsreset}[1]{%
 \expandafter\global\csname @gls@combined@#1@flagfalse\endcsname
}
```

\mglslocalunset Unset the flag.

```
\newcommand*{\mglslocalunset}[1]{%
 \gls@ifnotmeasuring
 {%
    \glsxtrifmulti{#1}{\@mglslocalunset{#1}}%
    {%
      \glsxtrundefaction{Multi entry ‘#1’ hasn’t been defined}%
      {You need to define ‘#1’ with \string\multiglossaryentry}%
    }%
 }%
}
```

\@mglslocalunset

```
\newcommand*{\@mglslocalunset}[1]{%
 \csname @gls@combined@#1@flagtrue\endcsname
}
```

\mglslocalreset Unset the flag.

```
\newcommand*{\mglslocalreset}[1]{%
 \gls@ifnotmeasuring
 {%
    \glsxtrifmulti{#1}{\@mglslocalreset{#1}}%
    {%
      \glsxtrundefaction{Multi entry ‘#1’ hasn’t been defined}%
      {You need to define ‘#1’ with \string\multiglossaryentry}%
    }%
 }%
}
```

\@mglslocalreset

```
\newcommand*{\@mglslocalreset}[1]{%
 \csname @gls@combined@#1@flagfalse\endcsname
}
```

\mglsunsetall Unset all.

```
\newcommand*{\mglsunsetall}{%
  \@for\@mgls@thislabel:=\@multi@glossaryentry@list\do{\mglsunset\@mgls@thislabel}%
}%
```

\mglsresetall Reset all.

```
\newcommand*{\mglsresetall}{%
  \@for\@mgls@thislabel:=\@multi@glossaryentry@list\do{\mglsreset\@mgls@thislabel}%
}%
```

\mglsSetMain

\mglsSetName{⟨*multi-label*⟩}{⟨*new main*⟩}

Allow the main label to be changed (local).

```
\newrobustcmd{\mglsSetMain}[2]{%
  \ifcsundef{@gls@combined@#1@main}%
  {\PackageError{glossaries-extra}{Multi-entry label `#1' not defined}{}}%
  {%
    \protected@edef\@gls@combined@current@main{#2}%
    \letcs\@gls@combined@currentlist{@gls@combined@#1@list}%
```

Check that the given label is in the list of elements and update main and last other element index.

```
    \mglselementindex=0\relax
    \count@=0\relax
    \@for\@gls@tmp:=\@gls@combined@currentlist\do{%
      \advance\mglselementindex by 1\relax
      \ifx\@gls@combined@current@main\@gls@tmp
        \count@=\mglselementindex\relax
        \let\@gls@combined@finalitem\@gls@tmp
        \ifmultiglossaryentryglobal
          \global\cslet{@gls@combined@#1@main}\@gls@combined@current@main
          \csxdef{@gls@combined@#1@mainindex}{\the\mglselementindex}%
        \else
          \cslet{@gls@combined@#1@main}\@gls@combined@current@main
          \csedef{@gls@combined@#1@mainindex}{\the\mglselementindex}%
        \fi
      \else
        \ifmultiglossaryentryglobal
          \csxdef{@gls@combined@#1@lastotherindex}{\the\mglselementindex}%
        \else
          \csedef{@gls@combined@#1@lastotherindex}{\the\mglselementindex}%
        \fi
      \fi
    }%
    \ifnum\count@=0\relax
      \PackageError{glossaries-extra}{Label `#2' is not in `#1' set
       (\@gls@combined@currentlist)}{}%
```

Default to final item.

```
      \ifmultiglossaryentryglobal
        \global\cslet{@gls@combined@#1@main}\@gls@combined@finalitem
```

```
13879         \csxdef{@gls@combined@#1@mainindex}{\the\mglselementindex}%
13880         \csxdef{@gls@combined@#1@lastotherindex}{%
13881           \number\numexpr\mglselementindex-1 }%
13882       \else
13883         \cslet{@gls@combined@#1@main}\@gls@combined@finalitem
13884         \csedef{@gls@combined@#1@mainindex}{\the\mglselementindex}%
13885         \csedef{@gls@combined@#1@lastotherindex}{%
13886           \number\numexpr\mglselementindex-1 }%
13887       \fi
13888     \fi
13889   }%
13890 }
```

\mglsSetOptions

\mglsSetOptions{⟨*multi-label*⟩}{⟨*new options*⟩}

Allow the options to be changed (local). No expansion is applied.

```
13891 \newrobustcmd{\mglsSetOptions}[2]{%
13892   \ifcsundef{@gls@combined@#1@main}%
13893   {\PackageError{glossaries-extra}{Multi-entry label `#1' not defined}{}}%
13894   {%
13895     \csdef{@gls@combined@#1@options}{#2}%
13896   }%
13897 }
```

\mglsAddOptions

\mglsAddOptions{⟨*multi-label*⟩}{⟨*extra options*⟩}

Allow the options to be changed (local). No expansion is applied.

```
13898 \newrobustcmd{\mglsAddOptions}[2]{%
13899   \ifcsundef{@gls@combined@#1@main}%
13900   {\PackageError{glossaries-extra}{Multi-entry label `#1' not defined}{}}%
13901   {%
13902     \ifcsempty{@gls@combined@#1@options}%
13903     {\csdef{@gls@combined@#1@options}{#2}}%
13904     {\csappto{@gls@combined@#1@options}{,#2}}%
13905   }%
13906 }
```

Options for \mgls:

\@mgls@all Options to apply to all elements.

```
13907 \newcommand*{\@mgls@all}{}
13908 \define@key{mgls}{all}{\renewcommand*{\@mgls@all}{#1}}
```

\@mgls@main Options to apply to the main element only.

```
\newcommand*{\@mgls@main}{}
\define@key{mgls}{main}{\renewcommand*{\@mgls@main}{#1}}
```

\@mgls@others Options to apply to the other (no main) elements.

```
\newcommand*{\@mgls@others}{}
\define@key{mgls}{others}{\renewcommand*{\@mgls@others}{#1}}
```

\@mgls@setup Options to apply to \multiglossaryentrysetup.

```
\newcommand*{\@mgls@setup}{}
\define@key{mgls}{setup}{%
 \@mgls@setup@do{\renewcommand*{\@mgls@setup}{#1}}%
}
```

\@mgls@setup@do

```
\newcommand*{\@mgls@setup@do}[1]{#1}
```

ls@setup@do@not

```
\newcommand*{\@mgls@setup@do@not}[1]{%
  \PackageError{glossaries-extra}{‘setup’ key not permitted inside
   ‘mglsopts’ value}{}%
}
```

s@disable@setup

```
\newcommand*{\mgls@disable@setup}{%
  \let\@mgls@setup@do\@mgls@setup@do@not
}
```

ls@enable@setup

```
\newcommand*{\mgls@enable@setup}{%
  \let\@mgls@setup@do\@firstofone
}
```

gls@unsetaction

```
\newcommand\@mgls@unsetaction{0}
\define@choicekey{mgls}{multiunset}[\@mgls@unsetaction@val\@mgls@unsetaction]%
 {global,local,none}{}
```

gls@presetlocal

```
\define@boolkey{mgls}{presetlocal}[true]{}
\KV@mgls@presetlocalfalse
```

\@mgls@hyper

```
\newcommand*{\@mgls@hyper}{}
\define@choicekey{mgls}{hyper}[\@mgls@hyper@val\@mgls@hyper@nr]{true,false}[true]%
{%
  \renewcommand*{\@mgls@hyper}{hyper=#1}%
  \ifnum\@mgls@hyper@nr=1\relax
```

```
    \let\@mgls@hyperlink\@secondoftwo
  \else
    \let\@mgls@hyperlink\@@mgls@hyperlink
  \fi
}
```

@mgls@hyperlink

```
\newcommand*{\@@mgls@hyperlink}[2]{%
  \ifx\@glslink\glsdonohyperlink
    #2%
  \else
    \glsxtr@org@dohyperlink{\glolinkprefix#1}{#2}%
  \fi
}
```

@mgls@hyperlink

```
\let\@mgls@hyperlink\@@mgls@hyperlink
```

mglsforelements

\mglsforelements{⟨*multi-label*⟩}{⟨*cs*⟩}{⟨*body*⟩}

```
\newcommand*{\mglsforelements}[3]{%
  \expandafter\@for\expandafter#2\expandafter:\expandafter
    =\csname @gls@combined@#1@list\endcsname\do{#3}%
}
```

orotherelements

\mglsforotherelements{⟨*multi-label*⟩}{⟨*cs*⟩}{⟨*body*⟩}

```
\newcommand*{\mglsforotherelements}[3]{%
  \expandafter\@for\expandafter#2\expandafter:\expandafter
    =\csname @gls@combined@#1@list\endcsname\do
  {\expandafter\ifdefequal\csname @gls@combined@#1@main\endcsname{#2}{}{#3}}%
}
```

mglsunsetothers

```
\newcommand*{\mglsunsetothers}[1]{%
  \mglsforotherelements{#1}{\@gls@tmp}{\glsunset{\@gls@tmp}}%
}
```

ocalunsetothers

```
\newcommand*{\mglslocalunsetothers}[1]{%
  \mglsforotherelements{#1}{\@gls@tmp}{\glslocalunset{\@gls@tmp}}%
}
```

glselementreset

```
\newcommand*{\mglselementreset}[1]{%
 \ifKV@mgls@presetlocal
  \glslocalreset{#1}%
 \else
  \glsreset{#1}%
 \fi
}
```

glselementunset

```
\newcommand*{\mglselementunset}[1]{%
 \ifKV@mgls@presetlocal
  \glslocalunset{#1}%
 \else
  \glsunset{#1}%
 \fi
}
```

\@mgls@resetall

```
\newcommand*{\@mgls@resetall}{}
\define@choicekey{mgls}{resetall}%
[\@mgls@resetall@val\@mgls@resetall@nr]{false,true}[true]%
{%
 \ifcase\@mgls@resetall@nr\relax
   \renewcommand*{\@mgls@resetall}{}%
 \or
   \renewcommand*{\@mgls@resetall}{%
     \@for\@gls@resetlabel:=\mglscurrentlist\do{\mglselementreset\@gls@resetlabel}}%
   \renewcommand*{\@mgls@unsetall}{}%
 \fi
}
```

@mgls@resetmain

```
\newcommand*{\@mgls@resetmain}{}
\define@choicekey{mgls}{resetmain}
[\@mgls@resetmain@val\@mgls@resetmain@nr]{false,true}[true]%
{%
 \ifcase\@mgls@resetmain@nr\relax
   \renewcommand*{\@mgls@resetmain}{}%
 \or
   \renewcommand*{\@mgls@resetmain}{\mglselementreset\mglscurrentmainlabel}%
   \renewcommand*{\@mgls@unsetmain}{}%
 \fi
}
```

gls@resetothers

```
\newcommand*{\@mgls@resetothers}{}
\define@choicekey{mgls}{resetothers}
[\@mgls@resetothers@val\@mgls@resetothers@nr]{false,true}[true]%
```

```
{%
 \ifcase\@mgls@resetothers@nr\relax
   \renewcommand*{\@mgls@resetothers}{}%
 \or
   \renewcommand*{\@mgls@resetothers}{%
     \@for\@gls@resetlabel:=\mglscurrentlist\do{%
       \ifx\@gls@resetlabel\mglscurrentmainlabel
       \else
         \mglselementreset\@gls@resetlabel
       \fi
     }%
   }%
   \renewcommand*{\@mgls@unsetothers}{}%
 \fi
}
```

\@mgls@unsetall

```
\newcommand*{\@mgls@unsetall}{}
\define@choicekey{mgls}{unsetall}%
[\@mgls@unsetall@val\@mgls@unsetall@nr]{false,true}[true]%
{%
 \ifcase\@mgls@unsetall@nr\relax
   \renewcommand*{\@mgls@unsetall}{}%
 \or
   \renewcommand*{\@mgls@unsetall}{%
     \@for\@gls@unsetlabel:=\mglscurrentlist\do{\mglselementunset\@gls@unsetlabel}}%
   \renewcommand*{\@mgls@resetall}{}%
 \fi
}
```

@mgls@unsetmain

```
\newcommand*{\@mgls@unsetmain}{}
\define@choicekey{mgls}{unsetmain}
[\@mgls@unsetmain@val\@mgls@unsetmain@nr]{false,true}[true]%
{%
 \ifcase\@mgls@unsetmain@nr\relax
   \renewcommand*{\@mgls@unsetmain}{}%
 \or
   \renewcommand*{\@mgls@unsetmain}{\mglselementunset\mglscurrentmainlabel}%
   \renewcommand*{\@mgls@resetmain}{}%
 \fi
}
```

gls@unsetothers

```
\newcommand*{\@mgls@unsetothers}{}
\define@choicekey{mgls}{unsetothers}
[\@mgls@unsetothers@val\@mgls@unsetothers@nr]{false,true}[true]%
{%
 \ifcase\@mgls@unsetothers@nr\relax
```

```
    \renewcommand*{\@mgls@unsetothers}{}%
  \or
    \renewcommand*{\@mgls@unsetothers}{%
      \@for\@gls@unsetlabel:=\mglscurrentlist\do{%
        \ifx\@gls@unsetlabel\mglscurrentmainlabel
        \else
          \mglselementunset\@gls@unsetlabel
        \fi
      }%
    }%
    \renewcommand*{\@mgls@resetothers}{}%
  \fi
}
```

setup@docurrent Set up the commands to determine whether or not to do the current element.

```
\newcommand{\glsxtr@setup@docurrent}{%
```

\mglscurrentlabel expands to the label of the current element. Should this element be skipped?

```
  \ifx\mglscurrentlabel\mglscurrentmainlabel
```

Main element. Should it be skipped?

```
    \mglsisfirstuse
    {%
      \ifKV@glsxtrcombined@firstskipmain
        \let\@mgls@do@current@element\@gobble
      \else
        \let\@mgls@do@current@element\@firstofone
      \fi
    }%
    {%
      \ifKV@glsxtrcombined@usedskipmain
        \let\@mgls@do@current@element\@gobble
      \else
        \let\@mgls@do@current@element\@firstofone
      \fi
    }%
  \else
```

Other element. Should it be skipped?

```
    \mglsisfirstuse
    {%
      \ifKV@glsxtrcombined@firstskipothers
        \let\@mgls@do@current@element\@gobble
      \else
        \let\@mgls@do@current@element\@firstofone
      \fi
    }%
    {%
      \ifKV@glsxtrcombined@usedskipothers
```

```
      \let\@mgls@do@current@element\@gobble
    \else
      \let\@mgls@do@current@element\@firstofone
    \fi
  }%
 \fi
}
```

hecklastelement If the last element is skipped, `\mglsiflast` needs adjusting. The first argument should be either `first` or `used`. The second argument is the multi-element label.

```
\newcommand*{\glsxtr@mgls@checklastelement}[2]{%
 \ifbool{KV@glsxtrcombined@#1skipmain}%
 {%
   \ifbool{KV@glsxtrcombined@#1skipothers}%
   {%
```

This condition has already been checked for.

```
   }%
   {%
```

Main skipped. The last item will be the last other element.

```
     \ifnum\mglselementindex=\glsxtrmultilastotherindex{#2}\relax
       \let\mglsiflast\@firstoftwo
     \else
       \let\mglsiflast\@secondoftwo
     \fi
   }%
 }%
 {%
```

Main not skipped.

```
   \ifbool{KV@glsxtrcombined@#1skipothers}%
   {%
```

Others skipped. The main element is the only item.

```
     \ifnum\mglselementindex=\glsxtrmultimainindex{#2}\relax
       \let\mglsiflast\@firstoftwo
     \else
       \let\mglsiflast\@secondoftwo
     \fi
   }%
   {%
```

None skipped. This isn't the last element.

```
     \let\mglsiflast\@secondoftwo
   }%
 }%
}
```

sWarnAllSkipped Warning if all elements are skipped. The first argument is the warning message, the second argument is the inserted content (final optional argument), the third command is the encapsulation command (which may be a hyperlink).

```
\newcommand{\glsxtrmglsWarnAllSkipped}[3]{%
  \GlossariesExtraWarning{#1}%
  #3{#2}%
}
```

@mgls@applyopts

```
\newcommand*{\glsxtr@mgls@applyopts}[1]{%
  \edef\@mgls@dooptions{\noexpand\setkeys*{mgls}{\expandonce#1}}%
  \@mgls@dooptions
```

Append any unknown options to all.

```
  \ifdefvoid\XKV@rm{}{\eappto\@mgls@all{,\expandonce\XKV@rm}}%
```

If setup key has been used, check for pre-option keys:

```
  \ifdefvoid\@mgls@setup
  {}%
  {%
    \edef\@mgls@dooptions{%
       \noexpand\setkeys*{glsxtrcombinedpreopts}{\expandonce\@mgls@setup}}%
    \mgls@disable@mglsopts
    \@mgls@dooptions
    \mgls@enable@mglsopts
```

Save remaining setup options.

```
    \ifx\@mgls@setupoptions\@empty
      \let\@mgls@setupoptions\XKV@rm
    \else
      \eappto\@mgls@setupoptions{,\expandonce\XKV@rm}%
    \fi
  }%
```

Apply gls unset/reset options.

```
  \@mgls@resetall
  \@mgls@unsetall
  \@mgls@resetmain
  \@mgls@unsetmain
  \@mgls@resetothers
  \@mgls@unsetothers
```

Disable.

```
  \let\@mgls@resetall\@empty
  \let\@mgls@resetmain\@empty
  \let\@mgls@resetothers\@empty
  \let\@mgls@unsetall\@empty
  \let\@mgls@unsetmain\@empty
  \let\@mgls@unsetothers\@empty
```

First use flags.

```
  \ifmglsused\mglscurrentmultilabel
  {\let\mglsisfirstuse\@secondoftwo}%
  {\let\mglsisfirstuse\@firstoftwo}%
}
```

\@firstofthree

```
\providecommand{\@firstofthree}[3]{#1}
```

\@secondofthree

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

\@thirdofthree

```
\providecommand{\@thirdofthree}[3]{#3}
```

The main internal command for referencing multi-entries:

sxtr@mgls@inner

\glsxtr@mgls@inner{⟨*options*⟩}{⟨*label*⟩}{⟨*insert*⟩}{⟨*first cs*⟩}{⟨*not first cs*⟩}{⟨*main first cs*⟩}{⟨*main other cs*⟩}

```
\newcommand*{\glsxtr@mgls@inner}[7]{%
 \let\mglslastmainlabel\@empty
 \let\mglsiflastmainwasfirstuse\@firstoftwo
 \let\mglsiflastmainwasplural\@secondoftwo
 \let\mglsiflastmaincapscase\@firstofthree
 \let\mglsiflastmainskipped\@firstoftwo
 \bgroup
 \ifcsundef{@gls@combined@#2@main}%
 {%
   \glsxtrundefaction{Multi entry '#2' hasn't been defined}%
   {You need to define '#2' with \string\multiglossaryentry}%
   \gdef\@mgls@post@hookdefs{%
     \protected@edef\mglslastmultilabel{#2}%
     \let\mglswasfirstuse\@firstoftwo
     \let\mglslastcategory\@empty
     \let\mglsiflastelementskipped\@firstoftwo
     \let\mglsiflastelementwasfirstuse\@firstoftwo
     \let\mglsiflastelementwasplural\@secondoftwo
     \let\mglsiflastelementcapscase\@firstofthree
     \let\mglslastelementlabel\@empty
     \let\mgls@do@postlinkhook\relax
   }%
 }%
 {%
   \protected@edef\mglscurrentmultilabel{#2}%
```

```
    \letcs\mglscurrentmainlabel{@gls@combined@#2@main}%
    \letcs\mglscurrentlist{@gls@combined@#2@list}%
    \letcs\mglscurrentoptions{@gls@combined@#2@options}%
```

Initialise (may be changed if `multiunset` is present):

```
    \ifmglsused\mglscurrentmultilabel
    {\let\mglsisfirstuse\@secondoftwo}%
    {\let\mglsisfirstuse\@firstoftwo}%
```

Only obtain pre-option keys:

```
    \edef\@mgls@dooptions{%
       \noexpand\setkeys*{glsxtrcombinedpreopts}{\expandonce\mglscurrentoptions}}%
    \@mgls@dooptions
```

Save remaining setup options.

```
    \let\@mgls@setupoptions\XKV@rm
```

Apply `\mgls` options.

```
    \mgls@disable@setup
    \ifdefvoid\@gls@combined@mglsopts
    {}%
    {\glsxtr@mgls@applyopts\@gls@combined@mglsopts}%
    \mgls@enable@setup
```

Apply options provided in `#1`.

```
    \ifstrempty{#1}{}{\def\@mgls@options{#1}\glsxtr@mgls@applyopts\@mgls@options}%
```

Check for attribute settings.

```
    \ifx\@gls@combined@category\empty
```

No category

```
    \else
```

Attribute options:

```
      \glshascategoryattribute{\@gls@combined@category}{multioptions}%
      {%
        \letcs\@mgls@attroptions{@glsxtr@categoryattr@@\@gls@combined@category
            @multioptions}%
```

Only obtain pre-option keys:

```
        \let\@gls@combined@mglsopts\@empty
        \edef\@mgls@dooptions{%
          \noexpand\setkeys*{glsxtrcombinedpreopts}{\expandonce\@mgls@attroptions}}%
        \@mgls@dooptions
```

Append remaining options:

```
        \eappto\@mgls@setupoptions{,\expandonce\XKV@rm}%
        \ifx\@gls@combined@mglsopts\@empty
        \else
```

mgls options found:

```
          \let\@mgls@setup\@empty
          \mgls@disable@setup
          \glsxtr@mgls@applyopts\@gls@combined@mglsopts
```

```
        \mgls@enable@setup
      \fi
    }%
    {}%
  \fi
```

Apply setup options.

```
  \edef\@mgls@dooptions{%
     \noexpand\setkeys{glsxtrcombined}{\expandonce\@mgls@setupoptions}}%
  \@mgls@dooptions
```

Provide local user-level access to category:

```
  \let\mglscurrentcategory\@gls@combined@category
```

Should the entire content be a hyperlink?

```
  \ifnum\@gls@combined@hyper=1\relax
    \def\@mgls@combinedlink{\@mgls@hyperlink{\mglscurrentmainlabel}}%
  \else
    \def\@mgls@combinedlink{\@firstofone}%
  \fi
```

Entire content encapsulator.

```
  \def\@gls@combined@encapsulator##1{%
      \@mgls@combinedlink{\csuse{\@gls@combined@textformat}{##1}}}%
```

Initialise.

```
  \let\@mgls@do@current@element\@firstofone
```

Check if all elements are being skipped.

```
  \mglsisfirstuse
  {%
    \ifKV@glsxtrcombined@firstskipmain
      \ifKV@glsxtrcombined@firstskipothers
```

Just do the warning and insert. This will ignore the loop.

```
        \let\@gls@org@combined@encapsulator\@gls@combined@encapsulator
        \def\@gls@combined@encapsulator##1{%
          \glsxtrmglsWarnAllSkipped{All elements skipped for
            first use of multi-entry `#2'}{#3}%
            {\@gls@org@combined@encapsulator}%
        }%
        \let\@mgls@do@current@element\@gobble
      \fi
    \fi
  }%
  {%
    \ifKV@glsxtrcombined@usedskipmain
      \ifKV@glsxtrcombined@usedskipothers
```

Just do the warning and insert. This will ignore the loop.

```
        \let\@gls@org@combined@encapsulator\@gls@combined@encapsulator
        \def\@gls@combined@encapsulator##1{%
```

```
          \glsxtrmglsWarnAllSkipped{All elements skipped for
            subsequent use of multi-entry `#2'}{#3}%
            {\@gls@org@combined@encapsulator}%
        }%
        \let\@mgls@do@current@element\@gobble
      \fi
    \fi
  }%
```

Determine prefix and suffix.

```
  \mglsisfirstuse
  {%
    \let\mglscurrentprefix\@gls@combined@firstprefix
    \let\mglscurrentsuffix\@gls@combined@firstsuffix
  }%
  {%
    \let\mglscurrentprefix\@gls@combined@usedprefix
    \let\mglscurrentsuffix\@gls@combined@usedsuffix
  }%
```

Set up post-link hook used after current scope.

```
  \xdef\@mgls@post@hookdefs{%
   \noexpand\def\noexpand\mglslastmultilabel{\expandonce\mglscurrentmultilabel}%
   \noexpand\def\noexpand\mglslastcategory{\mglscurrentcategory}%
  }%
  \ifx\@mgls@do@current@element\@gobble
    \gappto\@mgls@post@hookdefs{%
      \let\mglsiflastelementskipped\@firstoftwo
      \let\mglslastelementlabel\@empty
      \let\mglsiflastelementwasfirstuse\@firstoftwo
      \let\mglsiflastelementwasplural\@secondoftwo
      \let\mglsiflastelementcapscase\@firstofthree
    }%
  \fi
  \mglsisfirstuse
  {%
    \gappto\@mgls@post@hookdefs{\let\mglswasfirstuse\@firstoftwo}%
```

Determine if the multi-entry post-link hook should be applied.

```
    \ifcase\@gls@combined@mpostlink@nr\relax
```

`mpostlink=false`.

```
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\relax}%
    \or
```

`mpostlink=true`.

```
      \ifcase\@gls@combined@mpostlinkelement@nr\relax
        \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglslastelementpostlinkhook}%
      \or
        \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglslastmainpostlinkhook}%
      \or
```

```
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglscustompostlinkhook}%
    \fi
  \or
```

mpostlink=firstonly.

```
    \ifcase\@gls@combined@mpostlinkelement@nr\relax
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglslastelementpostlinkhook}%
    \or
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglslastmainpostlinkhook}%
    \or
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglscustompostlinkhook}%
    \fi
  \or
```

mpostlink=usedonly.

```
    \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\relax}%
  \fi
}%
{%
  \gappto\@mgls@post@hookdefs{\let\mglswasfirstuse\@secondoftwo}%
```

Determine if the multi-entry post-link hook should be applied.

```
  \ifcase\@gls@combined@mpostlink@nr\relax
```

mpostlink=false.

```
    \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\relax}%
  \or
```

mpostlink=true.

```
    \ifcase\@gls@combined@mpostlinkelement@nr\relax
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglslastelementpostlinkhook}%
    \or
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglslastmainpostlinkhook}%
    \or
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglscustompostlinkhook}%
    \fi
  \or
```

mpostlink=firstonly.

```
    \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\relax}%
  \or
```

mpostlink=usedonly.

```
    \ifcase\@gls@combined@mpostlinkelement@nr\relax
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglslastelementpostlinkhook}%
    \or
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglslastmainpostlinkhook}%
    \or
      \gappto\@mgls@post@hookdefs{\let\mgls@do@postlinkhook\mglscustompostlinkhook}%
    \fi
  \fi
}%
```

Save current post-link hook.

```
    \let\mgls@org@postlinkhook\glspostlinkhook
```

Prefix.

```
    \mglsprefix
```

Initialise last element label (for \mglssuffix).

```
    \let\mglslastelementlabel\@empty
    \@gls@combined@encapsulator
    {%
```

Save previous label.

```
     \def\@mgls@previouslabel{}%
     \mglselementindex=0\relax
     \@for\mglscurrentlabel:=\mglscurrentlist\do{%
      \advance\mglselementindex by 1\relax
      \glsxtr@setup@docurrent
```

Is this the last element?

```
      \ifx\@xfor@nextelement\@nnil
        \let\mglsiflast\@firstoftwo
      \else
        \let\mglsiflast\@secondoftwo
```

Are any elements being skipped?

```
        \mglsisfirstuse
        {%
         \glsxtr@mgls@checklastelement{first}{#2}%
        }%
        {%
         \glsxtr@mgls@checklastelement{used}{#2}%
        }%
      \fi
```

Should the element post-link hook be used?

```
      \ifcase\@gls@combined@postlinks@nr\relax
```

`postlinks=none`

```
        \let\glspostlinkhook\relax
      \or
```

`postlinks=all`

```
        \let\glspostlinkhook\mgls@org@postlinkhook
      \or
```

`postlinks=notlast`

```
        \mglsiflast
        {%
          \let\glspostlinkhook\relax
        }%
        {%
          \let\glspostlinkhook\mgls@org@postlinkhook
        }%
      \or
```

postlinks=mainnotlast

```
      \ifx\mglscurrentlabel\mglscurrentmainlabel
        \mglsiflast
        {%
          \let\glspostlinkhook\relax
        }%
        {%
          \let\glspostlinkhook\mgls@org@postlinkhook
        }%
      \else
        \let\glspostlinkhook\relax
      \fi
    \or
```

postlinks=mainonly

```
      \ifx\mglscurrentlabel\mglscurrentmainlabel
        \let\glspostlinkhook\mgls@org@postlinkhook
      \else
        \let\glspostlinkhook\relax
      \fi
    \or
```

postlinks=othernotlast

```
      \ifx\mglscurrentlabel\mglscurrentmainlabel
        \let\glspostlinkhook\relax
      \else
        \mglsiflast
        {%
          \let\glspostlinkhook\relax
        }%
        {%
          \let\glspostlinkhook\mgls@org@postlinkhook
        }%
      \fi
    \or
```

postlinks=otheronly

```
      \ifx\mglscurrentlabel\mglscurrentmainlabel
        \let\glspostlinkhook\relax
      \else
        \let\glspostlinkhook\mgls@org@postlinkhook
      \fi
    \fi
```

Save the last element for the multi-entry post-link hook.

```
    \mglsiflast
    {%
      \xappto\@mgls@post@hookdefs{%
        \noexpand\def\noexpand\mglslastelementlabel
          {\expandonce\mglscurrentlabel}}%
    }%
```

```
      {}%
```

Do current element:

```
      \@mgls@do@current@element
      {%
```

Pre element hook.

```
        \mglselementprehook
```

Is this the first use of the current element?

```
        \GlsXtrIfUnusedOrUndefined{\mglscurrentlabel}%
        {\let\@mgls@current@iffirstuse\@firstoftwo}%
        {\let\@mgls@current@iffirstuse\@secondoftwo}%
        \ifx\mglscurrentlabel\mglscurrentmainlabel
```

Main element. Location encap option:

```
          \edef\@mgls@current@options{format=\@gls@combined@encapmain}%
```

Indexing option:

```
          \ifcase\@gls@combined@indexmain
           \appto\@mgls@current@options{,noindex}%
          \or
           \appto\@mgls@current@options{,noindex=false}%
          \or
           \@mgls@current@iffirstuse
           {\appto\@mgls@current@options{,noindex=false}}%
           {\appto\@mgls@current@options{,noindex}}%
          \fi
```

Hyperlink option:

```
          \ifcase\@gls@combined@hyper\relax
            \appto\@mgls@current@options{,hyper=false}% none
          \or
            \appto\@mgls@current@options{,hyper=false}% allmain
          \or
            \eappto\@mgls@current@options{,\@mgls@hyper}% mainonly
          \or
            \eappto\@mgls@current@options{,\@mgls@hyper}% individual
          \or
            \appto\@mgls@current@options{,hyper=false}% otheronly
          \or
           \mglsisfirstuse
           {%
            \appto\@mgls@current@options{,hyper=false}% notmainfirst
           }%
           {%
            \eappto\@mgls@current@options{,\@mgls@hyper}% notmainfirst
           }%
          \or
            \eappto\@mgls@current@options{,\@mgls@hyper}% nototherfirst
          \or
           \mglsisfirstuse
```

```
          {%
           \appto\@mgls@current@options{,hyper=false}% notfirst
          }%
          {%
           \eappto\@mgls@current@options{,\@mgls@hyper}% notfirst
          }%
         \fi
```

Append all and then main:

```
         \eappto\@mgls@current@options{,\@mgls@all,\@mgls@main}%
       \else
```

Other element. Location encap option:

```
         \edef\@mgls@current@options{format=\@gls@combined@encapothers}%
```

Indexing option:

```
         \ifcase\@gls@combined@indexothers\relax
          \appto\@mgls@current@options{,noindex}%
         \or
          \appto\@mgls@current@options{,noindex=false}%
         \or
          \@mgls@current@iffirstuse
          {\appto\@mgls@current@options{,noindex=false}}%
          {\appto\@mgls@current@options{,noindex}}%
         \fi
```

Hyperlink option:

```
         \ifcase\@gls@combined@hyper\relax
           \appto\@mgls@current@options{,hyper=false}% none
         \or
           \appto\@mgls@current@options{,hyper=false}% allmain
         \or
           \appto\@mgls@current@options{,hyper=false}% mainonly
         \or
           \eappto\@mgls@current@options{,\@mgls@hyper}% individual
         \or
           \eappto\@mgls@current@options{,\@mgls@hyper}% otheronly
         \or
           \eappto\@mgls@current@options{,\@mgls@hyper}% notmainfirst
         \or
          \mglsisfirstuse
          {%
           \appto\@mgls@current@options{,hyper=false}% nototherfirst
          }%
          {%
           \eappto\@mgls@current@options{,\@mgls@hyper}% nototherfirst
          }%
         \or
          \mglsisfirstuse
          {%
           \appto\@mgls@current@options{,hyper=false}% notfirst
```

```
14498           }%
14499           {%
14500            \eappto\@mgls@current@options{,\@mgls@hyper}% notfirst
14501           }%
14502          \fi
```

Append all and then others:

```
14503          \eappto\@mgls@current@options{,\@mgls@all,\@mgls@others}%
14504        \fi
```

Is this the first element?

```
14505        \ifx\@mgls@previouslabel\empty
14506         \ifx\mglscurrentlabel\mglscurrentmainlabel
14507           \let\@mgls@cs#6\relax
14508         \else
14509           \let\@mgls@cs#4\relax
14510         \fi
14511        \else
```

Not the first element so add separator.

```
14512          \@mgls@previous@iffirstuse
14513          {%
14514            \@mgls@current@iffirstuse
14515            {\glscombinedfirstsepfirst{\@mgls@previouslabel}{\mglscurrentlabel}}%
14516            {\glscombinedfirstsep{\@mgls@previouslabel}{\mglscurrentlabel}}%
14517          }%
14518          {%
14519            \@mgls@current@iffirstuse
14520            {\glscombinedsepfirst{\@mgls@previouslabel}{\mglscurrentlabel}}%
14521            {\glscombinedsep{\@mgls@previouslabel}{\mglscurrentlabel}}%
14522          }%
14523         \ifx\mglscurrentlabel\mglscurrentmainlabel
14524           \let\@mgls@cs#7\relax
14525         \else
14526           \let\@mgls@cs#5\relax
14527         \fi
14528        \fi
```

Is this the last element?

```
14529        \mglsiflast
14530         {\expandafter\@mgls@cs\expandafter{\@mgls@current@options}{\mglscurrentlabel}[#3]}%
14531         {\expandafter\@mgls@cs\expandafter{\@mgls@current@options}{\mglscurrentlabel}[]}%
```

Is this the main element? If so, save information for post-link hook.

```
14532        \ifx\mglscurrentlabel\mglscurrentmainlabel
14533         \xappto\@mgls@post@hookdefs{%
14534           \noexpand\def\noexpand\mglslastmainlabel
14535             {\expandonce\mglscurrentmainlabel}%
14536         }%
14537         \glsxtrifwasfirstuse
14538         {%
```

```
        \gappto\@mgls@post@hookdefs{\let\mglsiflastmainwasfirstuse\@firstoftwo}%
      }%
      {%
        \gappto\@mgls@post@hookdefs{\let\mglsiflastmainwasfirstuse\@secondoftwo}%
      }%
      \glsifplural
      {%
        \gappto\@mgls@post@hookdefs{\let\mglsiflastmainwasplural\@firstoftwo}%
      }%
      {%
        \gappto\@mgls@post@hookdefs{\let\mglsiflastmainwasplural\@secondoftwo}%
      }%
      \glscapscase
      {%
        \gappto\@mgls@post@hookdefs{%
          \let\mglsiflastmaincapscase\@firstofthree
        }%
      }%
      {%
        \gappto\@mgls@post@hookdefs{%
          \let\mglsiflastmaincapscase\@secondofthree
        }%
      }%
      {%
        \gappto\@mgls@post@hookdefs{%
          \let\mglsiflastmaincapscase\@thirdofthree
        }%
      }%
    \fi
    \let\@mgls@previouslabel\mglscurrentlabel
    \let\@mgls@previous@iffirstuse\@mgls@current@iffirstuse
  }%
```

Post element hook.

```
    \mglselementposthook
  }%
  \ifx\mglslastmainlabel\@empty
    \gappto\@mgls@post@hookdefs{\let\mglsiflastmainskipped\@firstoftwo}%
  \else
    \gappto\@mgls@post@hookdefs{\let\mglsiflastmainskipped\@secondoftwo}%
  \fi
```

Encapsulator may introduce grouping so check here.

```
  \ifx\@mgls@do@current@element\@gobble
    \gappto\@mgls@post@hookdefs{\let\mglsiflastelementskipped\@firstoftwo}%
  \else
    \gappto\@mgls@post@hookdefs{\let\mglsiflastelementskipped\@secondoftwo}%
  \fi
  \glsxtrifwasfirstuse
  {%
```

```
    \gappto\@mgls@post@hookdefs{\let\mglsiflastelementwasfirstuse\@firstoftwo}%
   }%
   {%
    \gappto\@mgls@post@hookdefs{\let\mglsiflastelementwasfirstuse\@secondoftwo}%
   }%
   \glsifplural
   {%
    \gappto\@mgls@post@hookdefs{\let\mglsiflastelementwasplural\@firstoftwo}%
   }%
   {%
    \gappto\@mgls@post@hookdefs{\let\mglsiflastelementwasplural\@secondoftwo}%
   }%
   \glscapscase
   {%
     \gappto\@mgls@post@hookdefs{%
       \let\mglsiflastelementcapscase\@firstofthree
     }%
   }%
   {%
     \gappto\@mgls@post@hookdefs{%
       \let\mglsiflastelementcapscase\@secondofthree
     }%
   }%
   {%
     \gappto\@mgls@post@hookdefs{%
       \let\mglsiflastelementcapscase\@thirdofthree
     }%
   }%
  }%
```

Suffix needs post-link hook commands.

```
  \@mgls@post@hookdefs
  \mglssuffix
```

Unset multi-entry first use flag after current scope.

```
  \ifcase\@mgls@unsetaction\relax
    \xappto\@mgls@post@hookdefs{%
      \noexpand\mglsunset{\expandonce\mglscurrentmultilabel}}%
  \or
    \xappto\@mgls@post@hookdefs{%
      \noexpand\mglslocalunset{\expandonce\mglscurrentmultilabel}}%
  \fi
}%
\glsxtrmglswrite{#2}%
\egroup
\@mgls@post@hookdefs
\mgls@do@postlinkhook
}
```

tompostlinkhook

```
\newcommand*{\mglscustompostlinkhook}{}

entpostlinkhook
\newcommand*{\mglslastelementpostlinkhook}{%
 \let\glsxtrifwasfirstuse\mglsiflastelementwasfirstuse
 \let\glsifplural\mglsiflastelementwasplural
 \let\glscapscase\mglsiflastelementcapscase
 \let\glslabel\mglslastelementlabel
 \glspostlinkhook
}

ainpostlinkhook
\newcommand*{\mglslastmainpostlinkhook}{%
 \let\glsxtrifwasfirstuse\mglsiflastmainwasfirstuse
 \let\glsifplural\mglsiflastmainwasplural
 \let\glscapscase\mglsiflastmaincapscase
 \let\glslabel\mglslastmainlabel
 \glspostlinkhook
}

fcategoryprefix
\newcommand*{\mglsdefcategoryprefix}[2]{%
 \csdef{mglsprefix@#1}{#2}%
}

scategoryprefix
\newcommand*{\mglshascategoryprefix}[3]{%
 \ifcsdef{mglsprefix@#1}{#2}{#3}%
}

ecategoryprefix
\newcommand*{\mglsusecategoryprefix}[1]{%
 \csuse{mglsprefix@#1}%
}

    \mglsprefix
\newcommand*{\mglsprefix}{%
 \ifdefempty\mglscurrentcategory
 {\mglscurrentprefix}%
 {%
   \mglshascategoryprefix{\mglscurrentcategory}%
   {\mglsusecategoryprefix{\mglscurrentcategory}}%
   {\mglscurrentprefix}%
 }%
}

fcategorysuffix
\newcommand*{\mglsdefcategorysuffix}[2]{%
 \csdef{mglssuffix@#1}{#2}%
}
```

scategorysuffix

```
\newcommand*{\mglshascategorysuffix}[3]{%
 \ifcsdef{mglssuffix@#1}{#2}{#3}%
}
```

ecategorysuffix

```
\newcommand*{\mglsusecategorysuffix}[1]{%
 \csuse{mglssuffix@#1}%
}
```

\mglssuffix

```
\newcommand*{\mglssuffix}{%
 \ifdefempty\mglscurrentcategory
 {\ifdefempty{\mglscurrentsuffix}{}{\space(\mglscurrentsuffix)}}%
 {%
   \mglshascategorysuffix{\mglscurrentcategory}%
   {\mglsusecategorysuffix{\mglscurrentcategory}}%
   {\ifdefempty{\mglscurrentsuffix}{}{\space(\mglscurrentsuffix)}}%
 }%
}
```

selementprehook

```
\newcommand*{\mglselementprehook}{}
```

elementposthook

```
\newcommand*{\mglselementposthook}{}
```

Separators.

\glscombinedsep Separator between two elements that have been marked as used. This takes the two element labels as arguments.

```
\newcommand*{\glscombinedsep}[2]{%
  \glshasattribute{#1}{combinedsep}%
  {\glsgetattribute{#1}{combinedsep}}%
  { }%
}
```

edfirstsepfirst Separator following and preceding a first use.

```
\newcommand*{\glscombinedfirstsepfirst}[2]{%
  \glshasattribute{#1}{combinedfirstsepfirst}%
  {\glsgetattribute{#1}{combinedfirstsepfirst}}%
  {\glscombinedsep{#1}{#2}}%
}
```

ombinedfirstsep Separator following a first use.

```
\newcommand*{\glscombinedfirstsep}[2]{%
  \glshasattribute{#1}{combinedfirstsep}%
  {\glsgetattribute{#1}{combinedfirstsep}}%
  {\glscombinedsep{#1}{#2}}%
}
```

ombinedsepfirst Separator preceding a first use.

```
\newcommand*{\glscombinedsepfirst}[2]{%
  \glshasattribute{#1}{combinedsepfirst}%
  {\glsgetattribute{#1}{combinedsepfirst}}%
  {\glscombinedsep{#1}{#2}}%
}
```

nedsepabbrvnbsp Provide shortcut for using non-breakable space following an abbreviation that has already been used.

```
\newcommand*{\glssetcombinedsepabbrvnbsp}{%
 \renewcommand*{\glscombinedsep}[2]{%
  \glshasattribute{##1}{combinedsep}%
  {\glsgetattribute{##1}{combinedsep}}%
  {\ifhasshort{##1}{~}{ }}%
 }%
 \renewcommand*{\glscombinedsepfirst}[2]{%
  \glshasattribute{##1}{combinedsepfirst}%
  {\glsgetattribute{##1}{combinedsepfirst}}%
  {\ifhasshort{##1}{~}{ }}%
 }%
 \renewcommand*{\glscombinedfirstsep}[2]{%
  \glshasattribute{##1}{combinedfirstsep}%
  {\glsgetattribute{##1}{combinedfirstsep}}%
  { }%
 }%
 \renewcommand*{\glscombinedfirstsepfirst}[2]{%
  \glshasattribute{##1}{combinedfirstsepfirst}%
  {\glsgetattribute{##1}{combinedfirstsepfirst}}%
  { }%
 }%
}
```

nedsepabbrvnone Provide shortcut for using nothing if either on next use are abbreviations (otherwise use space).

```
\newcommand*{\glssetcombinedsepabbrvnone}{%
 \renewcommand*{\glscombinedsep}[2]{%
   \glshasattribute{##1}{combinedsep}%
   {\glsgetattribute{##1}{combinedsep}}%
   {\ifhasshort{##1}{}{\ifhasshort{##2}{}{ }}}%
 }%
 \renewcommand*{\glscombinedsepfirst}[2]{%
   \glshasattribute{##1}{combinedsepfirst}%
   {\glsgetattribute{##1}{combinedsepfirst}}%
   {\ifhasshort{##1}{}{ }}%
 }%
 \renewcommand*{\glscombinedfirstsep}[2]{%
  \glshasattribute{##1}{combinedfirstsep}%
  {\glsgetattribute{##1}{combinedfirstsep}}%
  {\ifhasshort{##2}{}{ }}%
```

```
 }%
 \renewcommand*{\glscombinedfirstsepfirst}[2]{%
  \glshasattribute{##1}{combinedfirstsepfirst}%
  {\glsgetattribute{##1}{combinedfirstsepfirst}}%
  { }%
 }%
}
```

mbinedsepnarrow Measures both.

```
\newcommand*{\glssetcombinedsepnarrow}[2]{%
 \renewcommand*{\glscombinedsep}[2]{%
  \glshasattribute{##1}{combinedsep}%
  {\glsgetattribute{##1}{combinedsep}}%
  {%
   \ifhasshort{##1}%
   {\settowidth{\dimen@}{\glsentryshort{##1}}}%
   {\settowidth{\dimen@}{\glsentrytext{##1}}}%
   \ifdim\dimen@<#1\relax
    #2%
   \else
     \ifhasshort{##2}%
     {\settowidth{\dimen@}{\glsentryshort{##2}}}%
     {\settowidth{\dimen@}{\glsentrytext{##2}}}%
     \ifdim\dimen@<#1\relax
      #2%
     \else
      \space
     \fi
   \fi
  }%
 }%
 \renewcommand*{\glscombinedsepfirst}[2]{%
  \glshasattribute{##1}{combinedsepfirst}%
  {\glsgetattribute{##1}{combinedsepfirst}}%
  {%
   \ifhasshort{##1}%
   {\settowidth{\dimen@}{\glsentryshort{##1}}}%
   {\settowidth{\dimen@}{\glsentrytext{##1}}}%
   \ifdim\dimen@<#1\relax
    #2%
   \else
     \ifhaslong{##2}%
     {\settowidth{\dimen@}{\glsentrylong{##2}}}%
     {\settowidth{\dimen@}{\glsentryfirst{##2}}}%
     \ifdim\dimen@<#1\relax
      #2%
     \else
      \space
     \fi
```

```
  \fi
 }%
}%
\renewcommand*{\glscombinedfirstsep}[2]{%
 \glshasattribute{##1}{combinedfirstsep}%
 {\glsgetattribute{##1}{combinedfirstsep}}%
 {%
   \ifhaslong{##1}%
   {\settowidth{\dimen@}{\glsentrylong{##1}}}%
   {\settowidth{\dimen@}{\glsentryfirst{##1}}}%
   \ifdim\dimen@<#1\relax
    #2%
   \else
     \ifhasshort{##2}%
     {\settowidth{\dimen@}{\glsentryshort{##2}}}%
     {\settowidth{\dimen@}{\glsentrytext{##2}}}%
     \ifdim\dimen@<#1\relax
      #2%
     \else
      \space
     \fi
   \fi
 }%
}%
\renewcommand*{\glscombinedfirstsepfirst}[2]{%
 \glshasattribute{##1}{combinedfirstsepfirst}%
 {\glsgetattribute{##1}{combinedfirstsepfirst}}%
 {%
   \ifhaslong{##1}%
   {\settowidth{\dimen@}{\glsentrylong{##1}}}%
   {\settowidth{\dimen@}{\glsentryfirst{##1}}}%
   \ifdim\dimen@<#1\relax
    #2%
   \else
     \ifhaslong{##2}%
     {\settowidth{\dimen@}{\glsentrylong{##2}}}%
     {\settowidth{\dimen@}{\glsentryfirst{##2}}}%
     \ifdim\dimen@<#1\relax
      #2%
     \else
      \space
     \fi
   \fi
 }%
}%
}
```

`lsxtr@mglswrite` Write information to the aux file for `bib2gls` to pick up, but only need to do it once per label since it only indicates which multi-entry has been referenced without any additional

information.

```
\newcommand{\glsxtrmglswrite}[1]{%
 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@off
 \else
  \protected@edef\@glsxtr@mglslabel{#1}%
  \ifdef\@glsxtr@mglsreflist
  {%
    \expandafter\DTLifinlist\expandafter{\@glsxtr@mglslabel}%
    {\@glsxtr@mglsreflist}{}%
    {%
      \xappto\@glsxtr@mglsreflist{,\expandonce\@glsxtr@mglslabel}%
      \if@mgls@writeseparaterefs
        \protected@write\@auxout{}{\string\@glsxtr@mglsrefs{#1}}%
      \fi
    }%
  }%
  {%
    \global\let\@glsxtr@mglsreflist\@glsxtr@mglslabel
    \if@mgls@writeseparaterefs
      \protected@write\@auxout{}{\string\@glsxtr@mglsrefs{#1}}%
    \else
      \AtEndDocument{\immediate\protected@write\@auxout{}%
        {\string\@glsxtr@mglsrefs{\@glsxtr@mglsreflist}}}%
    \fi
    \@mgls@disable@writeseparateref@cond
  }%
 \fi
}
```

`glsxtr@mglsrefs`

```
\newcommand{\@glsxtr@mglsrefs}[1]{}
```

`iteseparaterefs` If this conditional is changed, it must be done before the first instance of any `\mgls`-like command.

```
\newif\if@mgls@writeseparaterefs \@mgls@writeseparaterefsfalse
```

`eparateRefsTrue`

```
\newcommand{\mglsWriteSeparateRefsTrue}{\global\@mgls@writeseparaterefstrue}
```

`parateRefsFalse`

```
\newcommand{\mglsWriteSeparateRefsFalse}{\global\@mgls@writeseparaterefsfalse}
```

`eparateref@cond`

```
\newcommand*{\@mgls@disable@writeseparateref@cond}{%
  \gdef\mglsWriteSeparateRefsTrue{\PackageError{glossaries-extra}%
   {Too late to use \string\mglsWriteSeparateRefsTrue}%
   {\string\mglsWriteSeparateRefsTrue\space can only be used before
   the first instance of any \string\mgls-like command}}%
```

```
  \gdef\mglsWriteSeparateRefsFalse{\PackageError{glossaries-extra}%
   {Too late to use \string\mglsWriteSeparateRefsFalse}%
   {\string\mglsWriteSeparateRefsFalse\space can only be used before
   the first instance of any \string\mgls-like command}}%
}
```

\glsxtr@newmgls

```
\newcommand{\glsxtr@newmgls}[5]{%
  \edef\@glsxr@newmgls@do{%
   \noexpand\newrobustcmd*{\expandonce{\csname #1\endcsname}}%
    {\noexpand\@gls@hyp@opt\expandonce{\csname ns@glsxtr@#1\endcsname}}%
   \noexpand\newcommand*{\expandonce{\csname ns@glsxtr@#1\endcsname}}[2][]{%
    \noexpand\new@ifnextchar[%
     {\expandonce{\csname glsxtr@#1\endcsname}{####1}{####2}}%
     {\expandonce{\csname glsxtr@#1\endcsname}{####1}{####2}[]}%
   }%
   \noexpand\def\expandonce{\csname glsxtr@#1\endcsname}####1####2[####3]{%
     \noexpand\def\noexpand\glsxtrcurrentmglscsname{#1}%
     \noexpand\glsxtr@mgls@inner{####1}{####2}{####3}%
      {\noexpand#2}{\noexpand#3}{\noexpand#4}{\noexpand#5}%
    }%
  }%
  \@glsxr@newmgls@do
  \ifx\@glsxtr@record@setting\@glsxtr@record@setting@off
  \else
```

Provide a way for bib2gls to recognise the command (this will make it easier to add extra commands without having to modify bib2gls).

```
    \ifdef\@glsxtr@mglslikelist
    {\xappto\@glsxtr@mglslikelist{,#1}}%
    {%
      \gdef\@glsxtr@mglslikelist{#1}%
      \AtEndDocument{\immediate\protected@write\@auxout{}%
        {\string\@glsxtr@mglslike{\@glsxtr@mglslikelist}}}%
    }%
  \fi
}
```

\glsxtr@mglslike

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

GlsXtrMglsOrGls

\GlsXtrMglsOrGls{⟨*mgls cs*⟩}{⟨*gls cs*⟩}⟨*modifier*⟩[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

```
\newcommand*{\GlsXtrMglsOrGls}[2]{%
```

```
  \def\@glsxtr@mgls@or@gls@mcs{#1}%
  \def\@glsxtr@mgls@or@gls@gcs{#2}%
  \@ifstar{\s@GlsXtrMglsOrGls}%
  {%
   \@ifnextchar+{\PLUS\@firstoftwo{\p@GlsXtrMglsOrGls}}%
   {%
     \ifdefempty\@gls@alt@hyp@opt@char\@GlsXtrMglsOrGls\alt@GlsXtrMglsOrGls
   }%
  }%
}
```

GlsXtrMglsOrGls

```
\newcommand*{\alt@GlsXtrMglsOrGls}{
 \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
 {\@firstoftwo{\@alt@GlsXtrMglsOrGls}}{\@GlsXtrMglsOrGls}%
}
```

GlsXtrMglsOrGls

```
\newcommand*{\@GlsXtrMglsOrGls}[2][]{%
  \glsxtrifmulti{#2}%
  {\@glsxtr@mgls@or@gls@mcs[#1]{#2}}%
  {\@glsxtr@mgls@or@gls@gcs[#1]{#2}}%
}
```

GlsXtrMglsOrGls

```
\newcommand*{\s@GlsXtrMglsOrGls}[2][]{%
  \glsxtrifmulti{#2}%
  {\@glsxtr@mgls@or@gls@mcs*[#1]{#2}}%
  {\@glsxtr@mgls@or@gls@gcs*[#1]{#2}}%
}
```

GlsXtrMglsOrGls

```
\newcommand*{\p@GlsXtrMglsOrGls}[2][]{%
  \glsxtrifmulti{#2}%
  {\@glsxtr@mgls@or@gls@mcs+[#1]{#2}}%
  {\@glsxtr@mgls@or@gls@gcs+[#1]{#2}}%
}
```

GlsXtrMglsOrGls

```
\newcommand*{\@alt@GlsXtrMglsOrGls}[2][]{%
  \glsxtrifmulti{#2}%
  {\expandafter\@glsxtr@mgls@or@gls@mcs\@gls@alt@hyp@opt@char[#1]{#2}}%
  {\expandafter\@glsxtr@mgls@or@gls@gcs\@gls@alt@hyp@opt@char[#1]{#2}}%
}
```

\mgls

\mgls[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \gls for all elements.

```
\glsxtr@newmgls{mgls}{\@gls@}{\@gls@}{\@gls@}{\@gls@}%
```

\mglspl

\mglspl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \glspl for all elements.

```
\glsxtr@newmgls{mglspl}{\@glspl@}{\@glspl@}{\@glspl@}{\@glspl@}%
```

\mglsmainpl

\mglsmainpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Only use \glspl for the main element, otherwise use \gls.

```
\glsxtr@newmgls{mglsmainpl}{\@gls@}{\@gls@}{\@glspl@}{\@glspl@}%
```

\Mgls

\Mgls[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \Gls for first element and \gls for others.

```
\glsxtr@newmgls{Mgls}{\@Gls@}{\@gls@}{\@Gls@}{\@gls@}%
```

\Mglspl

\Mglspl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \Glspl for first element and \glspl for others.

```
\glsxtr@newmgls{Mglspl}{\@Glspl@}{\@glspl@}{\@Glspl@}{\@glspl@}%
```

\Mglsmainpl

\Mglsmainpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Upper case the first element, no case change for others. Use plural for the main element only.

```
\glsxtr@newmgls{Mglsmainpl}{\@Gls@}{\@gls@}{\@Glspl@}{\@glspl@}%
```

\MGls

\MGls[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \Gls for all elements.

```
14942 \glsxtr@newmgls{MGls}{\@Gls@}{\@Gls@}{\@Gls@}{\@Gls@}%
```

\MGlspl

\MGlspl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \Glspl for all elements.

```
14943 \glsxtr@newmgls{MGlspl}{\@Glspl@}{\@Glspl@}{\@Glspl@}{\@Glspl@}%
```

\MGlsmainpl

\MGlsmainpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Start all elements with upper case. Only use plural for main element.

```
14944 \glsxtr@newmgls{MGlsmainpl}{\@Gls@}{\@Gls@}{\@Glspl@}{\@Glspl@}%
```

\MGLS

\MGLS[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \GLS for all elements.

```
14945 \glsxtr@newmgls{MGLS}{\@GLS@}{\@GLS@}{\@GLS@}{\@GLS@}%
```

\MGLSpl

\MGLSpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \GLSpl for all elements.

```
14946 \glsxtr@newmgls{MGLSpl}{\@GLSpl@}{\@GLSpl@}{\@GLSpl@}{\@GLSpl@}%
```

\MGLSmainpl

\MGLSmainpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Upper case all elements. Only use plural for main element.

```
14947 \glsxtr@newmgls{MGLSmainpl}{\@GLS@}{\@GLS@}{\@GLSpl@}{\@GLSpl@}%
```

@glslongortext@

```
\def\@glslongortext#1#2[#3]{%
  \ifglshaslong{#2}{\@glsxtrlong{#1}{#2}[#3]}{\@glstext@{#1}{#2}[#3]}%
}
```

glsshortortext@

```
\def\@glsshortortext#1#2[#3]{%
  \ifglshasshort{#2}{\@glsxtrshort{#1}{#2}[#3]}{\@glstext@{#1}{#2}[#3]}%
}
```

glsfullorfirst@

```
\def\@glsfullorfirst#1#2[#3]{%
  \ifglshasshort{#2}{\@glsxtr@full{#1}{#2}[#3]}{\@glsfirst@{#1}{#2}[#3]}%
}
```

@Glslongortext@

```
\def\@Glslongortext#1#2[#3]{%
  \ifglshaslong{#2}{\@Glsxtrlong{#1}{#2}[#3]}{\@Glstext@{#1}{#2}[#3]}%
}
```

Glsshortortext@

```
\def\@Glsshortortext#1#2[#3]{%
  \ifglshasshort{#2}{\@Glsxtrshort{#1}{#2}[#3]}{\@Glstext@{#1}{#2}[#3]}%
}
```

Glsfullorfirst@

```
\def\@Glsfullorfirst#1#2[#3]{%
  \ifglshasshort{#2}{\@Glsxtr@full{#1}{#2}[#3]}{\@Glsfirst@{#1}{#2}[#3]}%
}
```

\mglsshort

\mglsshort[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use short or text for all elements.

```
\glsxtr@newmgls{mglsshort}%
{\@glsshortortext}{\@glsshortortext}{\@glsshortortext}{\@glsshortortext}%
```

\mglslong

\mglslong[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use long or text for all elements.

```
\glsxtr@newmgls{mglslong}%
{\@glslongortext}{\@glslongortext}{\@glslongortext}{\@glslongortext}%
```

\mglsfull

\mglsfull[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use full or first for all elements.

```
\glsxtr@newmgls{mglsfull}%
{\@glsfullorfirst}{\@glsfullorfirst}{\@glsfullorfirst}{\@glsfullorfirst}%
```

\Mglsshort

\Mglsshort[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use short or text for all elements with initial cap on first element.

```
\glsxtr@newmgls{Mglsshort}%
{\@Glsshortortext}{\@glsshortortext}{\@Glsshortortext}{\@glsshortortext}%
```

\Mglslong

\Mglslong[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use long or text for all elements with initial cap on first element.

```
\glsxtr@newmgls{Mglslong}%
{\@Glslongortext}{\@glslongortext}{\@Glslongortext}{\@glslongortext}%
```

\Mglsfull

\Mglsfull[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use full or first for all elements with initial cap on first element.

```
\glsxtr@newmgls{Mglsfull}%
{\@Glsfullorfirst}{\@glsfullorfirst}{\@Glsfullorfirst}{\@glsfullorfirst}%
```

\mglsname

\mglsname[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use name for all elements.

```
\glsxtr@newmgls{mglsname}%
 {\@glsname@}{\@glsname@}{\@glsname@}{\@glsname@}%
```

\Mglsname

\Mglsname[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use name for all elements with initial cap on first element.

```
\glsxtr@newmgls{Mglsname}%
 {\@Glsname@}{\@glsname@}{\@Glsname@}{\@glsname@}%
```

\MGlsname

\MGlsname[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use name for all elements with initial cap on all elements.

```
\glsxtr@newmgls{MGlsname}%
 {\@Glsname@}{\@Glsname@}{\@Glsname@}{\@Glsname@}%
```

@glssymbolorgls

```
\def\@glssymbolorgls#1#2[#3]{%
  \ifglshassymbol{#2}{\@glssymbol@{#1}{#2}[#3]}{\@gls@{#1}{#2}[#3]}%
}
```

@glssymbolorGls

```
\def\@glssymbolorGls#1#2[#3]{%
  \ifglshassymbol{#2}{\@glssymbol@{#1}{#2}[#3]}{\@Gls@{#1}{#2}[#3]}%
}
```

\mglssymbol

\mglssymbol[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \glssymbol if the symbol key is set otherwise use \gls.

```
\glsxtr@newmgls{mglssymbol}%
 {\@glssymbolorgls}{\@glssymbolorgls}{\@glssymbolorgls}{\@glssymbolorgls}%
```

\Mglssymbol

\Mglssymbol[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

As above but initial the first element if it's not a symbol.

```
\glsxtr@newmgls{Mglssymbol}%
 {\@glssymbolorGls}{\@glssymbolorgls}{\@glssymbolorGls}{\@glssymbolorgls}%
```

\MGlssymbol

\MGlssymbol[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

As above but initial each element if it's not a symbol.

```
\glsxtr@newmgls{MGlssymbol}%
 {\@glssymbolorGls}{\@glssymbolorGls}{\@glssymbolorGls}{\@glssymbolorGls}%
```

\mglsfield

```
\newcommand{\mglsfield}{useri}
```

\@glsfieldorgls

```
\def\@glsfieldorgls#1#2[#3]{%
  \glsxtrifhasfield{\mglsfield}{#2}%
  {\@glsdisp[#1]{#2}{\glscurrentfieldvalue#3}}%
  {\@gls@{#1}{#2}[#3]}%
}
```

\@Glsfieldorgls

```
\def\@Glsfieldorgls#1#2[#3]{%
  \glsxtrifhasfield{\mglsfield}{#2}%
  {\@glsdisp[#1]{#2}{\xmakefirstuc\glscurrentfieldvalue#3}}%
  {\@Gls@{#1}{#2}[#3]}%
}
```

\mglsusefield

\mglsusefield[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use the field given by \useri.

```
\glsxtr@newmgls{mglsusefield}%
 {\@glsfieldorgls}{\@glsfieldorgls}{\@glsfieldorgls}{\@glsfieldorgls}%
```

\Mglsusefield

\Mglsusefield[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

As above but use initial cap for first element only.

```
\glsxtr@newmgls{Mglsusefield}%
 {\@Glsfieldorgls}{\@glsfieldorgls}{\@Glsfieldorgls}{\@glsfieldorgls}%
```

\MGlsusefield

\MGlsusefield[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

As above but use initial cap for all elements.

```
\glsxtr@newmgls{MGlsusefield}%
 {\@Glsfieldorgls}{\@Glsfieldorgls}{\@Glsfieldorgls}{\@Glsfieldorgls}%
```

Use commands provided by glossaries-prefix if it has been loaded.

\mpglsWarning

```
\newcommand*{\mpglsWarning}{%
  \GlossariesExtraWarning{glossaries-prefix.sty is required for
   \string\mpgls\space family of commands (either load after
   glossaries-extra.sty or use the 'prefix' package option)}%
}
```

\@pglsorgls

```
\def\@pglsorgls#1#2[#3]{%
  \ifdef\@pgls@{\@pgls@{#1}{#2}[#3]}{\mpglsWarning\@gls@{#1}{#2}[#3]}%
}
```

\@pglsorglspl

```
\def\@pglsorglspl#1#2[#3]{%
  \ifdef\@pglspl@{\@pglspl@{#1}{#2}[#3]}{\mpglsWarning\@glspl@{#1}{#2}[#3]}%
}
```

\@Pglsorgls

```
\def\@Pglsorgls#1#2[#3]{%
  \ifdef\@Pgls@{\@Pgls@{#1}{#2}[#3]}{\mpglsWarning\@Gls@{#1}{#2}[#3]}%
}
```

\@pglsorglspl

```
\def\@pglsorglspl#1#2[#3]{%
  \ifdef\@pglspl@{\@pglspl@{#1}{#2}[#3]}{\mpglsWarning\@glspl@{#1}{#2}[#3]}%
}
```

\@Pglsorglspl

```
\def\@Pglsorglspl#1#2[#3]{%
  \ifdef\@Pglspl@{\@Pglspl@{#1}{#2}[#3]}{\mpglsWarning\@Glspl@{#1}{#2}[#3]}%
}
```

\@PGLSorgls

```
\def\@PGLSorgls#1#2[#3]{%
  \ifdef\@PGLS@{\@PGLS@{#1}{#2}[#3]}{\mpglsWarning\@GLS@{#1}{#2}[#3]}%
}
```

\@PGLSorglspl

```
\def\@PGLSorglspl#1#2[#3]{%
  \ifdef\@PGLSpl@{\@PGLSpl@{#1}{#2}[#3]}{\mpglsWarning\@GLSpl@{#1}{#2}[#3]}%
}
```

\mpgls

\mpgls[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \pgls for the first element and \gls for the remainder.

```
15039 \glsxtr@newmgls{mpgls}{\@pglsorgls@}{\@gls@}{\@pglsorgls@}{\@gls@}%
```

\mpglspl

\mpglspl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \pglspl for the first element and \glspl for the remainder.

```
15040 \glsxtr@newmgls{mpglspl}{\@pglsorglspl@}{\@glspl@}{\@pglsorglspl@}{\@glspl@}%
```

\mpglsmainpl

\mpglsmainpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Only use plural for main element and only use prefixing command for first element.

```
15041 \glsxtr@newmgls{mpglsmainpl}{\@pglsorgls@}{\@gls@}{\@pglsorglspl@}{\@glspl@}%
```

\Mpgls

\Mpgls[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \Pgls for the first element and \gls for the remainder.

```
15042 \glsxtr@newmgls{Mpgls}{\@Pglsorgls@}{\@gls@}{\@Pglsorgls@}{\@gls@}%
```

\Mpglspl

\Mpglspl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \Pglspl for the first element and \glspl for the remainder.

```
15043 \glsxtr@newmgls{Mpglspl}{\@Pglsorglspl@}{\@glspl@}{\@Pglsorglspl@}{\@glspl@}%
```

\Mpglsmainpl

\Mpglsmainpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Only use plural for main element and only use first letter uppercase prefixing command for first element.

```
15044 \glsxtr@newmgls{Mpglsmainpl}{\@Pglsorgls@}{\@gls@}{\@Pglsorglspl@}{\@glspl@}%
```

\MPGls

\MPGls[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \Pgls for the first element and \Gls for the remainder.

```
15045 \glsxtr@newmgls{MPGls}{\@Pglsorgls@}{\@Gls@}{\@Pglsorgls@}{\@Gls@}%
```

\MPGlspl

\MPGlspl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \Pglspl for the first element and \Glspl for the remainder.

```
15046 \glsxtr@newmgls{MPGlspl}{\@Pglsorglspl@}{\@Glspl@}{\@Pglsorglspl@}{\@Glspl@}%
```

\MPGlsmainpl

\MPGlsmainpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Only use plural for main element and first letter uppercase all elements.

```
15047 \glsxtr@newmgls{MPGlsmainpl}{\@Pglsorgls@}{\@Gls@}{\@Pglsorglspl@}{\@Glspl@}%
```

\MPGLS

\MPGLS[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \PGLS for the first element and \GLS for the remainder.

```
15048 \glsxtr@newmgls{MPGLS}{\@PGLSorgls@}{\@GLS@}{\@PGLSorgls@}{\@GLS@}%
```

\MPGLSpl

\MPGLSpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Use \PGLSpl for the first element and \GLSpl for the remainder.

```
15049 \glsxtr@newmgls{MPGLSpl}{\@PGLSorglspl@}{\@GLSpl@}{\@PGLSorglspl@}{\@GLSpl@}%
```

\MPGLSmainpl

\MPGLSmainpl[⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩]

Only use plural for main element and uppercase all elements.

```
15050 \glsxtr@newmgls{MPGLSmainpl}{\@PGLSorgls@}{\@GLS@}{\@PGLSorglspl@}{\@GLSpl@}%
```

Not currently implementing any other variations.

1.10 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
```

1.11 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}[2021/11/22 v1.48 (NLCT)]
```

Provide convenient shortcut commands for predefined glossary types.

`ntunsrtacronyms`

```
\ifglsacronym
  \providecommand*{\printunsrtacronyms}[1][]{%
   \printunsrtglossary[type=\acronymtype,#1]}%
\fi
```

`printunsrtindex`

```
\ifglossaryexists{index}
{
  \providecommand*{\printunsrtindex}[1][]{%
   \printunsrtglossary[type=index,#1]}%
}{}
```

`intunsrtsymbols`

```
\ifglossaryexists{symbols}
{
  \providecommand*{\printunsrtsymbols}[1][]{%
   \printunsrtglossary[type=symbols,#1]}%
}{}
```

`intunsrtnumbers`

```
\ifglossaryexists{numbers}
{
  \providecommand*{\printunsrtnumbers}[1][]{%
   \printunsrtglossary[type=numbers,#1]}%
}{}
```

`rtabbreviations`

```
\ifglossaryexists{abbreviations}
{
  \providecommand*{\printunsrtabbreviations}[1][]{%
   \printunsrtglossary[type=abbreviations,#1]}%
}{}
```

`splaynumberlist` Allow `\glsdisplaynumberlist` and make it robust.

```
\renewcommand*{\glsdisplaynumberlist}[1]{%
  \glsdoifexists{#1}%
  {%
    {\let\bibglsdelimN\glsnumlistsep
     \let\bibglslastDelimN\glsnumlistlastsep
```

```
      \glsxtrusefield{#1}{location}%
    }%
  }%
}
\robustify\glsdisplaynumberlist
```

entrynumberlist

```
\renewcommand*{\glsentrynumberlist}[1]{\glsxtrusefield{#1}{location}}
```

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}{%
  \@ifstar\s@GlsXtrIfHasNonZeroChildCount\@GlsXtrIfHasNonZeroChildCount
}
```

nZeroChildCount

```
\newcommand*{\@GlsXtrIfHasNonZeroChildCount}[3]{%
  \@GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
}
```

nZeroChildCount

```
\newcommand*{\s@GlsXtrIfHasNonZeroChildCount}[3]{%
  \s@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}
```

ndexCounterLink

\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}
```

XtrDualBackLink

\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}%
}%
}
```

rdisplaysuppl oc

\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 (otherwise the title may section or chapter title, which may be confusing). As from v1.42, this now checks if the control sequence \glsxtr⟨*counter*⟩locfmt is defined.

```
  \newcommand*{\glsxtrdisplaylocnameref}[8]{%
    \ifcsdef{glsxtr#2locfmt}%
    {\glsxtrnamereflink{#3}{\csuse{glsxtr#2locfmt}{#4}{#5}}{#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}}%
```

```
      }%
    }%
  }
}
```

requationlocfmt

\glsxtrequationlocfmt{⟨*location*⟩}{⟨*title*⟩}

```
\newcommand*{\glsxtrequationlocfmt}[2]{(#1)}
```

sxtrnamereflink

\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
}
```

sxtrnameloclink

\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\@glo@tmp{\the\toks@}%
    \ifx\@glo@tmp\empty
    \else
      \protected@edef\@glo@tmp{\glsentrycounter\@glo@counterprefix\the\toks@}%
      \ifdefvoid\glsxtrsupplocationurl
      {%
        \expandafter\glsxtrfmtinternalnameref\expandafter{\@glo@tmp}%
        {\@glsnumberformat}{\glsxtr@locationhypertext}%
      }%
      {%
        \expandafter\glsxtrfmtexternalnameref\expandafter{\@glo@tmp}%
        {\@glsnumberformat}{\glsxtr@locationhypertext}{\glsxtrsupplocationurl}%
      }%
    \fi
  \else
    \@gls@ReturnAfterFi{%
      \@glsxtr@bibgls@removespaces#2\@nil
    }%
  \fi
}
```

internalnameref

\glsxtrfmtinternalnameloc{⟨*target*⟩}{⟨*format*⟩}{⟨*title*⟩}

```
\newcommand*{\glsxtrfmtinternalnameref}[3]{%
  \csuse{#2}{\glsdohyperlink{#1}{#3}}%
}
```

externalnameref

\glsxtrfmtexternalnameloc{⟨*target*⟩}{⟨*format*⟩}{⟨*title*⟩}{⟨*file*⟩}

```
\newcommand*{\glsxtrfmtexternalnameref}[4]{%
  \csuse{#2}{\hyperref{#4}{}{#1}{#3}}%
}
```

\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.

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

tWidestFallback

\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,}}%
  }%
}
```

labelprefixlist

\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.

```
 \protected@edef\@gls@thislabel{#1}%
 \@for\@glsxtr@prefix:=\@glsxtr@labelprefixes\do
 {%
   \protected@edef\@gls@thislabel{\@glsxtr@prefix#1}%
   \ifglsentryexists{\@gls@thislabel}{\@endfortrue}{}%
 }%
 \edef\@glo@tmp{\endgroup\noexpand\def\noexpand\@gls@thislabel{\@gls@thislabel}}\@glo@tmp
}
```

\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}%
}
```

Multi (compound/combined) entry commands used by bib2gls.

tryadjustedname

\glsxtrmultientryadjustedname{⟨*list1*⟩}{⟨*name*⟩}{⟨*list2*⟩}{⟨*label*⟩}

This command is used by bib2gls when it adjusts the name field of an entry that's been identified as a main entry in the multi-entry set ⟨*label*⟩.

The final argument ⟨*label*⟩ is the multi-entry label from which the set was obtained. The first argument ⟨*list1*⟩ is the list of other labels that come before the main label. The third argument ⟨*list2*⟩ is the remaining list of other labels. The ⟨*name*⟩ argument is the previous name before adjustment.

```
\newrobustcmd*{\glsxtrmultientryadjustedname}[4]{%
  \bgroup
    \let\@glsxtrmultientryadjustednamesep\glsxtrmultientryadjustednamesep
    \let\@glsxtrmultientryadjustednamepresep\glsxtrmultientryadjustednamepresep
    \let\@glsxtrmultientryadjustednamepostsep\glsxtrmultientryadjustednamepostsep
    \let\@glsxtrmultientryadjustednameother\glsxtrmultientryadjustednameother
    \let\@glsxtrmultientryadjustednamefmt\glsxtrmultientryadjustednamefmt
    \let\@glsxtrmultientryadjustednamefirstother\glsxtrmultientryadjustednameother
    \let\@glsxtrmultientryadjustednamefirstfmt\glsxtrmultientryadjustednamefmt
    \@glsxtrmultientryadjustedname{#1}{#2}{#3}{#4}%
  \egroup
}
```

tryadjustedname First letter upper case

```
\newrobustcmd*{\Glsxtrmultientryadjustedname}[4]{%
  \bgroup
    \let\@glsxtrmultientryadjustednamesep\glsxtrmultientryadjustednamesep
    \let\@glsxtrmultientryadjustednamepresep\glsxtrmultientryadjustednamepresep
    \let\@glsxtrmultientryadjustednamepostsep\glsxtrmultientryadjustednamepostsep
    \let\@glsxtrmultientryadjustednameother\glsxtrmultientryadjustednameother
    \let\@glsxtrmultientryadjustednamefmt\glsxtrmultientryadjustednamefmt
    \let\@glsxtrmultientryadjustednamefirstother\Glsxtrmultientryadjustednameother
    \let\@glsxtrmultientryadjustednamefirstfmt\Glsxtrmultientryadjustednamefmt
    \@glsxtrmultientryadjustedname{#1}{#2}{#3}{#4}%
  \egroup
}
```

tryadjustedname Title case

```
\newrobustcmd*{\GlsXtrmultientryadjustedname}[4]{%
  \bgroup
    \let\@glsxtrmultientryadjustednamesep\glsxtrmultientryadjustednamesep
    \let\@glsxtrmultientryadjustednamepresep\glsxtrmultientryadjustednamepresep
    \let\@glsxtrmultientryadjustednamepostsep\glsxtrmultientryadjustednamepostsep
    \let\@glsxtrmultientryadjustednameother\GlsXtrmultientryadjustednameother
    \let\@glsxtrmultientryadjustednamefmt\GlsXtrmultientryadjustednamefmt
    \let\@glsxtrmultientryadjustednamefirstother\GlsXtrmultientryadjustednameother
    \let\@glsxtrmultientryadjustednamefirstfmt\GlsXtrmultientryadjustednamefmt
    \@glsxtrmultientryadjustedname{#1}{#2}{#3}{#4}%
```

```
  \egroup
}
```

tryadjustedname All caps.

```
\newrobustcmd*{\GLSxtrmultientryadjustedname}[4]{%
  \bgroup
    \let\@glsxtrmultientryadjustednamesep\glsxtrmultientryadjustednamesep
    \let\@glsxtrmultientryadjustednamepresep\glsxtrmultientryadjustednamepresep
    \let\@glsxtrmultientryadjustednamepostsep\glsxtrmultientryadjustednamepostsep
    \let\@glsxtrmultientryadjustednameother\GLSxtrmultientryadjustednameother
    \let\@glsxtrmultientryadjustednamefmt\GLSxtrmultientryadjustednamefmt
    \let\@glsxtrmultientryadjustednamefirstother\GLSxtrmultientryadjustednameother
    \let\@glsxtrmultientryadjustednamefirstfmt\GLSxtrmultientryadjustednamefmt
    \@glsxtrmultientryadjustedname{#1}{#2}{#3}{#4}%
  \egroup
}
```

tryadjustedname

```
\newcommand*{\@glsxtrmultientryadjustedname}[4]{%
  \letcs\mglscurrentmainlabel{@gls@combined@#4@main}%
  \letcs\mglscurrentmainlist{@gls@combined@#4@list}%
  \letcs\mglscurrentmainoptions{@gls@combined@#4@options}%
  \ifblank{#1}%
  {%
    \@glsxtrmultientryadjustednamefirstfmt{#2}%
  }%
  {%
    \def\@mgls@previouslabel{}%
    \let\@gls@xtradjustedother\@glsxtrmultientryadjustednamefirstother
    \@for\mglscurrentlabel:=#1\do{%
       \ifx\@mgls@previouslabel\empty
       \else
         \@glsxtrmultientryadjustednamesep{\@mgls@previouslabel}{\mglscurrentlabel}%
       \fi
       \@gls@xtradjustedother{\mglscurrentlabel}%
       \let\@mgls@previouslabel\mglscurrentlabel
       \let\@gls@xtradjustedother\@glsxtrmultientryadjustednameother
    }%
    \@glsxtrmultientryadjustednamepresep{\@mgls@previouslabel}{\mglscurrentmainlabel}%
    \@glsxtrmultientryadjustednamefmt{#2}%
  }%
  \ifblank{#3}%
  {}%
  {%
    \let\@mgls@previouslabel\mglscurrentmainlabel
    \let\@gls@xtrmultientryadjustednamesep\@glsxtrmultientryadjustednamepostsep
    \@for\mglscurrentlabel:=#3\do{%
       \@gls@xtrmultientryadjustednamesep{\@mgls@previouslabel}{\mglscurrentlabel}%
       \@glsxtrmultientryadjustednameother{\mglscurrentlabel}%
```

```
        \let\@mgls@previouslabel\mglscurrentlabel
        \let\@gls@xtrmultientryadjustednamesep\@glsxtrmultientryadjustednamesep
    }%
  }%
}
```

adjustednamesep

```
\newcommand*{\glsxtrmultientryadjustednamesep}{\glscombinedfirstsepfirst}
```

ustednamepresep Separator before main name.

```
\newcommand*{\glsxtrmultientryadjustednamepresep}{\glsxtrmultientryadjustednamesep}
```

stednamepostsep Separator after main name.

```
\newcommand*{\glsxtrmultientryadjustednamepostsep}{\glsxtrmultientryadjustednamesep}
```

adjustednamefmt

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

justednameother

```
\newcommand*{\glsxtrmultientryadjustednameother}[1]{\glsentryname{#1}}
```

adjustednamefmt

```
\newcommand*{\Glsxtrmultientryadjustednamefmt}[1]{\makefirstuc{#1}}
```

justednameother

```
\newcommand*{\Glsxtrmultientryadjustednameother}[1]{\Glsentryname{#1}}
```

justednameother

```
\newcommand*{\GlsXtrmultientryadjustednameother}[1]{%
 \glsentrytitlecase{#1}{name}}
```

adjustednamefmt

```
\ifdef\glscapitalisewords
{%
  \newcommand*{\GlsXtrmultientryadjustednamefmt}[1]{\glscapitalisewords{#1}}
}
{
  \newcommand*{\GlsXtrmultientryadjustednamefmt}[1]{\capitalisewords{#1}}
}
```

justednameother

```
\newcommand*{\GLSxtrmultientryadjustednameother}[1]{%
 \mfirstucMakeUppercase{\glsentryname{#1}}}
```

adjustednamefmt

```
\newcommand*{\GLSxtrmultientryadjustednamefmt}[1]{\mfirstucMakeUppercase{#1}}
```

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
15574 \providecommand*{\Alpha}{\mathrm{A}}
   \Beta
15575 \providecommand*{\Beta}{\mathrm{B}}
\Epsilon
15576 \providecommand*{\Epsilon}{\mathrm{E}}
   \Zeta
15577 \providecommand*{\Zeta}{\mathrm{Z}}
    \Eta
15578 \providecommand*{\Eta}{\mathrm{H}}
   \Iota
15579 \providecommand*{\Iota}{\mathrm{I}}
  \Kappa
15580 \providecommand*{\Kappa}{\mathrm{K}}
     \Mu
15581 \providecommand*{\Mu}{\mathrm{M}}
     \Nu
15582 \providecommand*{\Nu}{\mathrm{N}}
\Omicron
15583 \providecommand*{\Omicron}{\mathrm{O}}
    \Rho
15584 \providecommand*{\Rho}{\mathrm{P}}
    \Tau
15585 \providecommand*{\Tau}{\mathrm{T}}
    \Chi
15586 \providecommand*{\Chi}{\mathrm{X}}
```

```
  \Digamma
15587 \providecommand*{\Digamma}{\mathrm{F}}

  \omicron
15588 \providecommand*{\omicron}{\mathit{o}}
```

Provide corresponding upright characters if upgreek has been loaded. (The upper case characters are the same as above.)

```
15589 \@ifpackageloaded{upgreek}%
15590 {

  \Upalpha
15591   \providecommand*{\Upalpha}{\mathrm{A}}

   \Upbeta
15592   \providecommand*{\Upbeta}{\mathrm{B}}

\Upepsilon
15593   \providecommand*{\Upepsilon}{\mathrm{E}}

   \Upzeta
15594   \providecommand*{\Upzeta}{\mathrm{Z}}

    \Upeta
15595   \providecommand*{\Upeta}{\mathrm{H}}

   \Upiota
15596   \providecommand*{\Upiota}{\mathrm{I}}

  \Upkappa
15597   \providecommand*{\Upkappa}{\mathrm{K}}

     \Upmu
15598   \providecommand*{\Upmu}{\mathrm{M}}

     \Upnu
15599   \providecommand*{\Upnu}{\mathrm{N}}

\Upomicron
15600   \providecommand*{\Upomicron}{\mathrm{O}}

    \Uprho
15601   \providecommand*{\Uprho}{\mathrm{P}}

    \Uptau
15602   \providecommand*{\Uptau}{\mathrm{T}}
```

```
\Upchi
15603   \providecommand*{\Upchi}{\mathrm{X}}
\upomicron
15604   \providecommand*{\upomicron}{\mathrm{o}}
15605 }%
15606 {}% 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.

```
15607 \newcommand*{\glsxtrcontrolrules}{%
15608  \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
15609  \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
15610  \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
15611  \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
15612  \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
15613  \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
15614 0010\string'\string=\glshex 0011
15615  \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
15616  \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
```

```
16108 }

\glsxtrLatinI
16109 \newcommand*{\glsxtrLatinI}{%
16110   i\string=\glshex 2071,I
16111 }

\glsxtrLatinK
16112 \newcommand*{\glsxtrLatinK}{%
16113   k\string=\glshex 2096,K
16114 }

\glsxtrLatinL
16115 \newcommand*{\glsxtrLatinL}{%
16116   l\string=\glshex 2097,L
16117 }

\glsxtrLatinM
16118 \newcommand*{\glsxtrLatinM}{%
16119   m\string=\glshex 2098,M
16120 }

\glsxtrLatinN
16121 \newcommand*{\glsxtrLatinN}{%
16122   n\string=\glshex 207F\string=\glshex 2099,N
16123 }

\glsxtrLatinO
16124 \newcommand*{\glsxtrLatinO}{%
16125   o\string=\glshex 00BA\string=\glshex 2092,O
16126 }

\glsxtrLatinP
16127 \newcommand*{\glsxtrLatinP}{%
16128   p\string=\glshex 209A,P
16129 }

\glsxtrLatinS
16130 \newcommand*{\glsxtrLatinS}{%
16131   s\string=\glshex 209B,S
16132 }

\glsxtrLatinT
16133 \newcommand*{\glsxtrLatinT}{%
16134   t\string=\glshex 209C,T
16135 }
```

\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
```

```
16523 }
```

glsxtrUpEpsilon

```
16524 \newcommand*{\glsxtrUpEpsilon}{%
16525  \glshex 03B5% lower case epsilon
16526  \string=\glshex 03F5,% lower case epsilon variant
16527  \glshex 0395% upper case epsilon
16528 }
```

glsxtrUpDigamma

```
16529 \newcommand*{\glsxtrUpDigamma}{%
16530  \glshex 03DD,% lower case digamma
16531  \glshex 03DC% upper case digamma
16532 }
```

\glsxtrUpZeta

```
16533 \newcommand*{\glsxtrUpZeta}{%
16534  \glshex 03B6,% lower case zeta
16535  \glshex 0396% upper case zeta
16536 }
```

\glsxtrUpEta

```
16537 \newcommand*{\glsxtrUpEta}{%
16538  \glshex 03B7,% lower case eta
16539  \glshex 0397% upper case eta
16540 }
```

\glsxtrUpTheta

```
16541 \newcommand*{\glsxtrUpTheta}{%
16542  \glshex 03B8% lower case theta
16543  \string=\glshex 03D1,% lower case theta variant
16544  \glshex 0398% upper case theta
16545 }
```

\glsxtrUpIota

```
16546 \newcommand*{\glsxtrUpIota}{%
16547  \glshex 03B9,% lower case iota
16548  \glshex 0399% upper case iota
16549 }
```

\glsxtrUpKappa

```
16550 \newcommand*{\glsxtrUpKappa}{%
16551  \glshex 03BA% lower case kappa
16552  \string=\glshex 03F0,% lower case kappa variant
16553  \glshex 039A% upper case kappa
16554 }
```

```
\glsxtrUpLambda
16555 \newcommand*{\glsxtrUpLambda}{%
16556  \glshex 03BB,% lower lambda
16557  \glshex 039B% upper case lambda
16558 }

    \glsxtrUpMu
16559 \newcommand*{\glsxtrUpMu}{%
16560  \glshex 03BC,% lower case mu
16561  \glshex 039C% upper case mu
16562 }

    \glsxtrUpNu
16563 \newcommand*{\glsxtrUpNu}{%
16564  \glshex 03BD,% lower case nu
16565  \glshex 039D% upper case nu
16566 }

    \glsxtrUpXi
16567 \newcommand*{\glsxtrUpXi}{%
16568  \glshex 03BE,% lower case xi
16569  \glshex 039E% upper case xi
16570 }

glsxtrUpOmicron
16571 \newcommand*{\glsxtrUpOmicron}{%
16572  \glshex 03BF,% lower case omicron
16573  \glshex 039F% upper case omicron
16574 }

    \glsxtrUpPi
16575 \newcommand*{\glsxtrUpPi}{%
16576  \glshex 03C0% lower case pi
16577  \string=\glshex 03D6,% lower case pi variant
16578  \glshex 03A0% upper case pi
16579 }

   \glsxtrUpRho
16580 \newcommand*{\glsxtrUpRho}{%
16581  \glshex 03C1% lower case rho
16582  \string=\glshex 03F1,% lower case rho variant
16583  \glshex 03A1% upper case rho
16584 }

 \glsxtrUpSigma
16585 \newcommand*{\glsxtrUpSigma}{%
16586  \glshex 03C2% lower case sigma
16587  \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
16654 \newcommand*{\glsxtrMathItalicKappa}{%
16655  \glshex 1D705% lower case kappa (maths italic)
16656  \string=\glshex 1D718,% lower case kappa variant (maths italic)
16657  \glshex 1D6EB% upper case kappa (maths italic)
16658 }

athItalicLambda
16659 \newcommand*{\glsxtrMathItalicLambda}{%
16660  \glshex 1D706,% lower case lambda (maths italic)
16661  \glshex 1D6EC% upper case lambda (maths italic)
16662 }

xtrMathItalicMu
16663 \newcommand*{\glsxtrMathItalicMu}{%
16664  \glshex 1D707,% lower case mu (maths italic)
16665  \glshex 1D6ED% upper case mu (maths italic)
16666 }

xtrMathItalicNu
16667 \newcommand*{\glsxtrMathItalicNu}{%
16668  \glshex 1D708,% lower case nu (maths italic)
16669  \glshex 1D6EE% upper case nu (maths italic)
16670 }

xtrMathItalicXi
16671 \newcommand*{\glsxtrMathItalicXi}{%
16672  \glshex 1D709,% lower case xi (maths italic)
16673  \glshex 1D6EF% upper case xi (maths italic)
16674 }

thItalicOmicron
16675 \newcommand*{\glsxtrMathItalicOmicron}{%
16676  \glshex 1D70A,% lower case omicron (maths italic)
16677  \glshex 1D6F0% upper case omicron (maths italic)
16678 }

xtrMathItalicPi
16679 \newcommand*{\glsxtrMathItalicPi}{%
16680  \glshex 1D70B% lower case pi (maths italic)
16681  \string=\glshex 1D71B,% lower case pi variant (maths italic)
16682  \glshex 1D6F1% upper case pi (maths italic)
16683 }

trMathItalicRho
16684 \newcommand*{\glsxtrMathItalicRho}{%
16685  \glshex 1D70C% lower case rho (maths italic)
16686  \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}[2021/11/22 v1.48 (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}%
    \RequirePackage{glossary-topic}%
  }
}

\DeclareOption*{%
  \IfFileExists{glossary-\CurrentOption.sty}
   {\eappto\@glsxtr@loadstyles{%
      \noexpand\RequirePackage{glossary-\CurrentOption}}%
   }%
```

```
16847    {%
16848      \PackageError{glossaries-extra-styles}%
16849      {Unknown option `\CurrentOption'}{}%
16850    }%
16851 }
```

Process the package options:

```
16852 \ProcessOptions
```

Load the required packages:

```
16853 \@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.

```
16854 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

ewglossarystyle

```
16855 \providecommand{\renewglossarystyle}[2]{%
16856   \ifcsundef{@glsstyle@#1}%
16857   {%
16858     \PackageError{glossaries-extra}{Glossary style `#1' isn't already defined}{}%
16859   }%
16860   {%
16861     \csdef{@glsstyle@#1}{#2}%
16862   }%
16863 }
```

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.

```
16864 \ifdef{\@glsstyle@listdotted}
16865 {%
16866  \renewglossarystyle{listdotted}{%
16867    \setglossarystyle{list}%
16868    \renewcommand*{\glossentry}[2]{%
16869     \item[]\makebox[\glslistdottedwidth][l]{%
16870       \glsentryitem{##1}%
16871       \glstarget{##1}{\glossentryname{##1}}%
16872       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
16873       \glossentrydesc{##1}\glspostdescription}%
16874    \renewcommand*{\subglossentry}[3]{%
16875     \item[]\makebox[\glslistdottedwidth][l]{%
16876     \glssubentryitem{##2}%
16877     \glstarget{##2}{\glossentryname{##2}}%
16878     \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}
```

`lslistgroupskip`

```
  \newcommand{\glslistgroupskip}{\nobreak\indexspace\nobreak}
```

`\glslistitem`

```
  \newcommand{\glslistitem}[1]{%
    \item[\glsentryitem{#1}%
          \glstarget{#1}{\glossentryname{#1}}]%
  }
```

`\glslistinit` This command was only added to glossary-list v4.48 so provide it if it hasn't been defined:

```
  \providecommand{\glslistinit}{%
    \ifdef\GetTitleStringDisableCommands
    {%
      \GetTitleStringSetup{expand}%
      \GetTitleStringDisableCommands{%
        \let\glsentryitem\@gobble
        \let\glstarget\@secondoftwo
        \let\glossentryname\glslistexpandedname
        \let\glslistgroupheaderfmt\@firstofone
        \let\glsgetgrouptitle\@firstofone
```

Technically this has an optional argument but it's not used in the list styles.

```
        \let\glsnavhypertarget\@secondoftwo
        \let\glsnavigation\relax
      }%
    }%
    {}%
  }
```

istexpandedname This command was only added to glossary-list v4.48 so provide it if it hasn't been defined. The original definition uses \glsunexpandedfieldvalue which was added to glossaries v4.48 (so if \glslistexpandedname hasn't been defined then neither will \glsunexpandedfieldvalue).

```
\providecommand{\glslistexpandedname}[1]{%
  \ifcsname glo@\glsdetoklabel{#1}@name\endcsname
   \expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\expandafter\endcsname
  \fi
}
```

Redefine list to use these commands.

```
\renewglossarystyle{list}{%
  \renewenvironment{theglossary}%
    {\glslistinit\begin{description}}{\end{description}}%
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{}%
  \renewcommand*{\glossentry}[2]{%
    \glslistitem{##1}\glslistdesc{##1}\glslistprelocation ##2}%
  \renewcommand*{\subglossentry}[3]{%
    \glssubentryitem{##2}%
    \glstarget{##2}{\strut}\space
    \glslistdesc{##2}%
    \glslistchildprelocation ##3\glslistchildpostlocation}%
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glslistgroupskip\fi}%
}
}
{}
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
\ifdef{\@glsstyle@altlist}
{%
```

\glsaltlistitem

```
\newcommand{\glsaltlistitem}[1]{%
  \glslistitem{#1}%
  \mbox{}\par\nobreak\@afterheading
}

\renewglossarystyle{altlist}{%
  \setglossarystyle{list}%
  \renewcommand*{\glossentry}[2]{%
    \glsaltlistitem{##1}%
```

```
      \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}
{%
```

roupafterheader

```
  \newcommand{\glslistgroupheaderitem}[2]{\item[{#2}]}
  \newcommand{\glslistgroupafterheader}{%
      \mbox{}\par\nobreak\@afterheading
  }

  \renewglossarystyle{listgroup}{%
    \setglossarystyle{list}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glslistgroupheaderitem{##1}{\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}}%
      \glslistgroupafterheader
    }%
  }
}
{}
```

Similarly for listhypergroup.

```
\ifdef{\@glsstyle@listhypergroup}
{%
  \renewglossarystyle{listhypergroup}{%
    \setglossarystyle{list}%
    \renewcommand*{\glossaryheader}{%
      \glslistnavigationitem{\glsnavigation}}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glslistgroupheaderitem{##1}{\glslistgroupheaderfmt
         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}}%
      \glslistgroupafterheader
    }%
  }
}
{}
```

Similarly for altlistgroup.

```
\ifdef{\@glsstyle@altlistgroup}
{%
  \renewglossarystyle{altlistgroup}{%
```

```
    \setglossarystyle{altlist}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glslistgroupheaderitem{##1}%
        {\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}}%
      \glslistgroupafterheader
    }%
  }
}
{}
```

Similarly for altlisthypergroup.

```
\ifdef{\@glsstyle@altlisthypergroup}
{%
  \renewglossarystyle{altlisthypergroup}{%
    \setglossarystyle{altlist}%
    \renewcommand*{\glossaryheader}{%
      \glslistnavigationitem{\glsnavigation}}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glslistgroupheaderitem{##1}{\glslistgroupheaderfmt
        {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}}%
      \glslistgroupafterheader
    }%
  }
}
{}
```

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}}
```

lstreePreHeader Takes the label as the first argument and title as the second argument so this can be modified to add a bookmark.

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

```
}
{}
```

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}
```

Don't prohibit a page break at the start of a new group if there's no header.

lstreegroupskip

```
\newcommand{\glstreegroupskip}{\indexspace}
```

groupheaderskip This doesn't include `\@afterheading` as it can cause interference with some styles.

```
\newcommand{\glstreegroupheaderskip}{\nopagebreak\glstreegroupskip\nobreak}
```

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}%
      \glstreeDescLoc{##1}{##2}%
    }%
    \renewcommand{\subglossentry}[3]{%
      \ifcase##1\relax
        \item
      \or
        \subitem
        \glssubentryitem{##2}%
      \else
        \subsubitem
      \fi
      \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
      \glstreechildsymbol{##2}%
      \glstreeChildDescLoc{##2}{##3}%
    }%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
  }
}
{}
```

The indexgroup style is redefined to discourage a page break after the heading.

```
\ifdef{\@glsstyle@indexgroup}
{%
  \renewglossarystyle{indexgroup}{%
    \setglossarystyle{index}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \item\glstreegroupheaderfmt{\glsxtr@grptitle}%
      \glstreegroupheaderskip\@afterheading
    }%
  }
}
{}
```

Similarly for indexhypergroup.

```
\ifdef{\@glsstyle@indexhypergroup}
{%
  \renewglossarystyle{indexhypergroup}{%
    \setglossarystyle{index}%
    \renewcommand*{\glossaryheader}{%
      \item\glstreenavigationfmt{\glsnavigation}%
       \glstreegroupheaderskip\@afterheading}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \item\glstreegroupheaderfmt
        {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
```

```
      \glstreegroupheaderskip\@afterheading}%
  }%
}
{}
```

Adjust tree style to remove hard coded space before number list.

```
\ifdef{\@glsstyle@tree}
{%
```

The original alttree style doesn't use \glstreepredesc but since v1.42 the modified style (below) has switched to using \glstreeDescLoc so provide an alternative that can be used with alttree.

sxtrtreepredesc

```
\newcommand{\glsxtrtreepredesc}{\glstreepredesc}
```

reechildpredesc

```
\newcommand{\glsxtrtreechildpredesc}{\glstreechildpredesc}
```

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]{%
  \glsxtrtreepredesc\glossentrydesc{#1}\glspostdescription
}
```

\glstreeDescLoc

\glstreeDescLoc{⟨*label*⟩}{⟨*location*⟩}

This checks for the description and symbol. If both are missing, a different separator may be required. For example, a comma and space if there's no description or symbol but just a space if either of those fields are present.

```
\newcommand{\glstreeDescLoc}[2]{%
  \ifglshasdesc{#1}%
  {\glstreedesc{#1}\glstreeprelocation}%
  {\ifglshassymbol{#1}{\glstreeprelocation}{\glstreeNoDescSymbolPreLocation}}%
  #2%
}
```

mbolPreLocation

\glstreeNoDescSymbolPreLocation

```
\newcommand{\glstreeNoDescSymbolPreLocation}{\space}
```

Similarly for the symbol.

\glstreesymbol

```
\newcommand{\glstreesymbol}[1]{%
  \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
}%
```

And for the child entries:

lstreechilddesc

```
\newcommand{\glstreechilddesc}[1]{%
  \glsxtrtreechildpredesc\glossentrydesc{#1}\glspostdescription
}%
```

reeChildDescLoc

```
\newcommand{\glstreeChildDescLoc}[2]{%
  \ifglshasdesc{#1}%
  {\glstreechilddesc{#1}\glstreechildprelocation}%
  {\ifglshassymbol{#1}{\glstreechildprelocation}%
    {\glstreeNoDescSymbolPreLocation}%
  }%
  #2%
}%
```

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}%
    \glstreeDescLoc{##1}{##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}%
      \glstreeChildDescLoc{##2}{##3}\par
    }%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
  }%
}
{}
```

The treegroup style is redefined to discourage a page break after the heading.

```
\ifdef{\@glsstyle@treegroup}
{%
  \renewglossarystyle{treegroup}{%
    \setglossarystyle{tree}%
    \renewcommand{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
      \glstreegroupheaderskip\@afterheading}%
  }
}
{}
```

Similarly for treehypergroup

```
\ifdef{\@glsstyle@treehypergroup}
{%
  \renewglossarystyle{treehypergroup}{%
    \setglossarystyle{tree}%
    \renewcommand*{\glossaryheader}{%
      \par\noindent\glstreenavigationfmt{\glsnavigation}%
      \glstreegroupheaderskip\@afterheading}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent
      \glstreegroupheaderfmt
        {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip\@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})}{}%
}%
```

eenonameDescLoc

```
\newcommand{\glstreenonameDescLoc}[2]{%
  \glstreenonamedesc{#1}\glstreeprelocation#2%
}
```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```
\newcommand{\glstreenonamechilddesc}[1]{%
  \glossentrydesc{#1}\glspostdescription
}%
```

ameChildDescLoc

```
\newcommand{\glstreenonameChildDescLoc}[2]{%
  \glstreenonamechilddesc{#1}\glstreechildprelocation#2%
}

\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}%

    \glstreenonameDescLoc{##1}{##2}\par
  }%
  \renewcommand{\subglossentry}[3]{%
    \hangindent##1\glstreeindent\relax
    \parindent##1\glstreeindent\relax
    \ifnum##1=1\relax
      \glssubentryitem{##2}%
    \fi
    \glstarget{##2}{\strut}%
    \glstreenonameChildDescLoc{##2}{##3}\par
  }%
  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
}
```

```
}
{}
```

The treenonamegroup style is redefined to discourage a page break after the heading.

```
\ifdef{\@glsstyle@treenonamegroup}
{%
  \renewglossarystyle{treenonamegroup}{%
    \setglossarystyle{treenoname}%
    \renewcommand{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
      \glstreegroupheaderskip\@afterheading
    }%
  }
}
{}
```

Similarly for treenonamehypergroup

```
\ifdef{\@glsstyle@treenonamehypergroup}
{%
  \renewglossarystyle{treenonamehypergroup}{%
    \setglossarystyle{treenoname}%
    \renewcommand*{\glossaryheader}{%
      \par\noindent\glstreenavigationfmt{\glsnavigation}%
      \glstreegroupheaderskip\@afterheading}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip\@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.

salttreepredesc

```
\newcommand{\glsalttreepredesc}{}
```

reechildpredesc

```
\newcommand{\glsalttreechildpredesc}{\glsalttreepredesc}
```

bolDescLocation

\glsxtralttreeSymbolDescLocation{⟨*label*⟩}{⟨*location list*⟩}

Layout the symbol, description and location for top-level entries.

```
\newcommand{\glsxtralttreeSymbolDescLocation}[2]{%
  {%
    \let\par\glsxtrAltTreePar
    \let\glsxtrtreepredesc\glsalttreepredesc
    \let\glsxtrtreechildpredesc\glsalttreechildpredesc
    \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
    \glstreeDescLoc{#1}{#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}%
}
```

bolDescLocation

\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}%
        {%
          \protected@edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
          \ifglshasparent{\@glo@parent}%
          {%
            \protected@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}%
      {%
        \protected@edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
        \ifglshasparent{\@glo@parent}%
        {%
          \protected@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
}
```

teTreeSubIndent

\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\glstreegroupskip\fi}%
  }
}%
{%
}
```

Redefine alttreegroup so that it discourages a break after group headings.

```
\ifdef{\@glsstyle@alttreegroup}
{%
  \renewglossarystyle{alttreegroup}{%
    \setglossarystyle{alttree}%
    \renewcommand{\glsgroupheading}[1]{\par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \glstreegroupheaderfmt{\glsxtr@grptitle}%
```

Can't use `\@afterheading` here as it messes with the first item of the group.

```
      \glstreegroupheaderskip
    }%
  }%
}%
{%
}
```

Similarly for alttreehypergroup.

```
\ifdef{\@glsstyle@alttreehypergroup}
{%
  \renewglossarystyle{alttreehypergroup}{%
    \setglossarystyle{alttree}%
    \renewcommand*{\glossaryheader}{%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreenavigationfmt{\glsnavigation}%
```

Can't use `\@afterheading` here as it messes with the first item of the group.

```
      \glstreegroupheaderskip
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
```

Can't use `\@afterheading` here as it messes with the first item of the group.

```
      \glstreegroupheaderskip
    }%
  }
}%
{%
}
```

2.9 Multicolumn Styles

Adjust mcolindexgroup to discourage page breaks after the group headings.

```
\ifdef{\@glsstyle@mcolindexgroup}
{%
  \renewglossarystyle{mcolindexgroup}{%
    \setglossarystyle{mcolindex}%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \item\glstreegroupheaderfmt{\glsxtr@grptitle}%
      \glstreegroupheaderskip\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcolindexhypergroup.

```
\ifdef{\@glsstyle@mcolindexhypergroup}
{%
  \renewglossarystyle{mcolindexhypergroup}{%
    \setglossarystyle{mcolindex}%
    \renewcommand*{\glossaryheader}{%
      \item\glstreenavigationfmt{\glsnavigation}%
      \glstreegroupheaderskip\@afterheading
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \item\glstreegroupheaderfmt
        {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip\@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]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \item\glstreegroupheaderfmt
        {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreegroup.

```
\ifdef{\@glsstyle@mcoltreegroup}
{%
  \renewglossarystyle{mcoltreegroup}{%
```

```
    \setglossarystyle{mcoltree}%
    \renewcommand{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
      \glstreegroupheaderskip\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreehypergroup.

```
\ifdef{\@glsstyle@mcoltreehypergroup}
{%
  \renewglossarystyle{mcoltreehypergroup}{%
    \setglossarystyle{mcoltree}%
    \renewcommand*{\glossaryheader}{%
      \par\noindent\glstreenavigationfmt{\glsnavigation}%
      \glstreegroupheaderskip
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip\@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]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent
```

```
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreenonamegroup.

```
\ifdef{\@glsstyle@mcoltreenonamegroup}
{%
  \renewglossarystyle{mcoltreenonamegroup}{%
    \setglossarystyle{mcoltreenoname}%
    \renewcommand{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
      \glstreegroupheaderskip\@afterheading
    }%
  }
}%
{%
}
```

Similarly for mcoltreenonamehypergroup.

```
\ifdef{\@glsstyle@mcoltreenonamehypergroup}
{%
  \renewglossarystyle{mcoltreenonamehypergroup}{%
    \setglossarystyle{mcoltreenoname}%
    \renewcommand*{\glossaryheader}{%
      \par\noindent\glstreenavigationfmt{\glsnavigation}%
      \glstreegroupheaderskip
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip\@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]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par\noindent
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip\@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]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsxtr@grptitle}%
      \glstreegroupheaderskip
    }%
```

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

Similarly for mcolalttreehypergroup.

```
\ifdef{\@glsstyle@mcolalttreehypergroup}
{%
  \renewglossarystyle{mcolalttreehypergroup}{%
    \setglossarystyle{mcolalttree}%
    \renewcommand*{\glossaryheader}{%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreenavigationfmt{\glsnavigation}%
      \glstreegroupheaderskip
    }%
    \renewcommand*{\glsgroupheading}[1]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip
    }%
  }
}%
{%
}
```

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]{%
      \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
      \glstreePreHeader{##1}{\glsxtr@grptitle}%
      \par
```

```
      \def\@gls@prevlevel{-1}%
      \hangindent0pt\relax
      \parindent0pt\relax
      \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
      \glstreegroupheaderskip
    }%
  }
}%
{%
}
```

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}[2021/11/22 v1.48 (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}%
}
```

`okindexlocation`

\glsxtrbookindexlocation{⟨*label*⟩}{⟨*location*⟩}

Displays the location.

```
\newcommand*{\glsxtrbookindexlocation}[2]{#2}
```

ndexsublocation

\glsxtrbookindexlocation{⟨*label*⟩}{⟨*location*⟩}

Displays the location for sub-entries.

```
\newcommand*{\glsxtrbookindexsublocation}{\glsxtrbookindexlocation}
```

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]{}
}
```

xbookmarkprefix Make the bookmark label prefix used for letter groups depend on the glossary label (instead of original hardcoded "index.").

```
\newcommand*{\glsxtrbookindexbookmarkprefix}{\currentglossary.}
```

kindexcolspread

```
\newcommand*{\glsxtrbookindexcolspread}{}
```

dexmulticolsenv

```
\newcommand*{\glsxtrbookindexmulticolsenv}{multicols}
```

Define the style.

```
\newglossarystyle{bookindex}{%
  \setglossarystyle{index}%
  \renewenvironment{theglossary}%
  {%
    \ifnum\glsxtrbookindexcols>1\relax
      \ifdefempty\glsxtrbookindexcolspread
      {%
        \edef\glsxtr@beginbookindex{%
          \noexpand\begin{\glsxtrbookindexmulticolsenv}
            {\glsxtrbookindexcols}%
        }%
      }%
      {%
        \edef\glsxtr@beginbookindex{%
          \noexpand\begin{\glsxtrbookindexmulticolsenv}%
            {\glsxtrbookindexcols}[\glsxtrbookindexcolspread]%
        }%
      }%
    \else
      \def\glsxtr@beginbookindex{}%
    \fi
    \glsxtr@beginbookindex
    \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.

```
    \ifnum\glsxtrbookindexcols>1\relax
      \edef\glsxtr@endbookindex{%
        \noexpand\end{\glsxtrbookindexmulticolsenv}%
      }%
    \else
      \def\glsxtr@endbookindex{}%
    \fi
    \glsxtr@endbookindex
  }%
```

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}%
    \glsxtrbookindexlocation{##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}%
    \glsxtrbookindexsublocation{##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}{\glsxtrcurrentgrptitle}%
```

Do the PDF bookmark if supported.

```
    \glsxtrbookindexbookmark{\glsxtrcurrentgrptitle}{\glsxtrbookindexbookmarkprefix##1}%
```

Format the group title.

```
    \glsxtrbookindexformatheader{\glsxtrcurrentgrptitle}%
    \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}[2021/11/22 v1.48 (NLCT)]
```

Load required packages.

```
\RequirePackage{glossary-longbooktabs}
```

ongextraNameFmt

\glslongextraNameFmt{⟨*label*⟩}

Governs the way the name is displayed.

```
\newcommand{\glslongextraNameFmt}[1]{%
 \glsentryitem{#1}\glstarget{#1}{\glossentryname{#1}}%
}
```

ongextraDescFmt

\glslongextraDescFmt{⟨*label*⟩}

Governs the way the description is displayed.

```
\newcommand{\glslongextraDescFmt}[1]{%
  \glossentrydesc{#1}\glspostdescription
}
```

gextraSymbolFmt

\glslongextraSymbolFmt{⟨*label*⟩}

Governs the way the symbol is displayed.

```
\newcommand{\glslongextraSymbolFmt}[1]{\glossentrysymbol{#1}}
```

xtraLocationFmt

\glslongextraLocationFmt{⟨*label*⟩}{⟨*location list*⟩}

Governs the way the location is displayed.

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

extraSubNameFmt

\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}%
}
```

extraSubDescFmt

\glslongextraSubDescFmt{⟨*level*⟩}{⟨*label*⟩}

Governs the way the child description is displayed.

```
\newcommand{\glslongextraSubDescFmt}[2]{%
  \glslongextraDescFmt{#2}%
}
```

traSubSymbolFmt

\glslongextraSubSymbolFmt{⟨*level*⟩}{⟨*label*⟩}

Governs the way the child symbol is displayed.

```
\newcommand{\glslongextraSubSymbolFmt}[2]{%
  \glslongextraSymbolFmt{#2}%
}
```

aSubLocationFmt

\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
  }%
}
```

dateWidestChild

\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
  }%
```

```
18732   \renewcommand{\subglossentry}[3]{%
18733      \glslongextraSubNameFmt{##1}{##2}&
18734      \glslongextraSubDescFmt{##1}{##2}&
18735      \glslongextraSubLocationFmt{##1}{##2}{##3}%
18736      \tabularnewline
18737   }%
18738   \ifglsnogroupskip
18739     \renewcommand*{\glsgroupskip}{}%
18740   \else
18741     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
18742   \fi
18743 }
```

aDescNameHeader

```
18744 \newcommand{\glslongextraDescNameHeader}{%
18745  \glslongextraDescNameTabularHeader\endhead
18746  \glslongextraDescNameTabularFooter\endfoot
18747 }
```

meTabularHeader

```
18748 \newcommand{\glslongextraDescNameTabularHeader}{%
18749  \toprule
18750  \glslongextraHeaderFmt\descriptionname&
18751  \glslongextraHeaderFmt\entryname \tabularnewline
18752  \midrule
18753 }
```

meTabularFooter

```
18754 \newcommand{\glslongextraDescNameTabularFooter}{%
18755  \bottomrule
18756 }
```

long-desc-name Like name-desc but swaps the columns.

```
18757 \newglossarystyle{long-desc-name}%
18758 {%
18759   \ifGlsLongExtraUseTabular
18760    \renewenvironment{theglossary}%
18761     {%
18762       \glslongextraSetDescWidth
18763       \edef\@glslongextra@begintab{%
18764         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
18765           \expandonce\glslongextraDescAlign
18766           \expandonce\glslongextraNameAlign}}%
18767       \@glslongextra@begintab
18768     }%
18769     {%
18770       \glslongextraDescNameTabularFooter
18771       \end{tabular}%
18772     }%
```

```
    \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
}
```

5 topic styles (glossary-topic.sty)

5.1 Package Initialisation and Options

Provides "topic" styles where top-level entries are considered a topic.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossary-topic}[2021/11/22 v1.48 (NLCT)]
```

Load required package.

```
\RequirePackage{multicol}
```

The top-level entries act like headers. If the top-level entry has a description it's placed below the name.

topic

```
\newglossarystyle{topic}{%
  \renewenvironment{theglossary}%
  {%
    \glstopicInit
    \def\glstopic@prechildren{}%
    \def\glstopic@prevlevel{-1}%
  }%
  {\par}%
  \renewcommand*{\glossaryheader}{}%
  \renewcommand*{\glsgroupheading}[1]{%
    \def\glstopic@prevlevel{-1}%
    \glstopicGroupHeading{##1}%
  }%
  \renewcommand{\glossentry}[2]{%
    \hangindent0pt\relax
    \parindent\glstopicParIndent\relax
    \glstopicItem{##1}{##2}%
```

If there isn't a description, penalise a page break.

```
     \ifglshasdesc{##1}%
     {%
       \def\glstopic@prechildren{}%
     }%
     {%
       \def\glstopic@prechildren{\nopagebreak}%
     }%
  }%
  \renewcommand{\subglossentry}[3]{%
    \ifnum\glstopic@prevlevel=0\relax\glstopic@prechildren\fi
    \def\glstopic@prevlevel{##1}%
```

Grouping is added to scope the effect of \everypar.

```
    \begingroup
    \glstopicAssignSubIndent{##1}%
    \glstopicSubItem{##1}{##2}{##3}%
    \par
    \endgroup
  }%
  \renewcommand*{\glsgroupskip}{}%
}
```

picGroupHeading

\glstopicGroupHeading{⟨*group label*⟩}

May be redefined if letter group headings are required. For example:

```
\renewcommand*{\glstopicGroupHeading}[1]{%
  \glsxtrgetgrouptitle{#1}{\thisgrptitle}%
  \section*{\thisgrptitle}%
}
```

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

\glstopicItem

\glstopicItem{⟨*label*⟩}{⟨*location list*⟩}

```
\newcommand*{\glstopicItem}[2]{%
  \glspar\glstopicPreSkip\glspar\noindent
  \glstopicMarker{#1}%
  \glstopicTitleFont
  {%
    \glsentryitem{#1}\glstarget{#1}{\glstopicTitle{#1}}%
  }%
  \ifglshasdesc{#1}%
  {\glspar\nobreak\glstopicMidSkip\glspar\nobreak
   \@afterheading\glstopicDesc{#1}\glspar\glstopicPostSkip}%
  {\glspar\nobreak\glstopicPostSkip}%
  \glstopicLoc{#1}{#2}%
}
```

\glstopicMarker May be used to insert a bookmark etc if required.

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

\glstopicName

```
\newcommand*{\glstopicTitle}[1]{\Glossentryname{#1}%
  \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
}
```

stopicTitleFont

```
19473 \newcommand*{\glstopicTitleFont}[1]{\textbf{\large #1}}
```

\glstopicDesc

```
19474 \newcommand*{\glstopicDesc}[1]{\Glossentrydesc{#1}\glspostdescription}
```

\glstopicLoc

```
19475 \newcommand*{\glstopicLoc}[2]{}
```

stopicParIndent

```
19476 \newlength\glstopicParIndent
19477 \setlength\glstopicParIndent{20pt}
```

stopicSubIndent

```
19478 \newlength\glstopicSubIndent
19479 \setlength\glstopicSubIndent{20pt}
```

\glstopicInit

```
19480 \newcommand{\glstopicInit}{}
```

AssignSubIndent

\glstopicAssignSubIndent{⟨*level*⟩}

Used to set the indentation for sub-levels.

```
19481 \newcommand*{\glstopicAssignSubIndent}[1]{%
19482   \par
19483   \parindent\dimexpr#1\glstopicSubIndent-\glstopicSubIndent\relax
19484   \glstopicAssignWidest{#1}%
19485   \glstopicsubitemhangindent=\dimexpr\parindent+\glstopicwidest\relax
19486   \hangindent\glstopicsubitemhangindent\relax
19487   \everypar{\hangindent\glstopicsubitemhangindent\relax
19488     \parindent\dimexpr\glstopicSubItemParIndent+\glstopicsubitemhangindent\relax}%
19489 }
```

bitemhangindent

```
19490 \newlength\glstopicsubitemhangindent
```

ubItemParIndent

```
19491 \newlength\glstopicSubItemParIndent
19492 \glstopicSubItemParIndent\parindent
```

\glstopicwidest

```
19493 \newlength\glstopicwidest
```

picAssignWidest

\glstopicAssignWidest{⟨*level*⟩}

Used in the definition of \glstopicAssignSubIndent to set the indentation from the widest name for the given level. This will require glossary-tree to set the values.

```
\newcommand*{\glstopicAssignWidest}[1]{%
  \ifcsundef{@glswidestlength\romannumeral#1}%
  {%
    \ifcsdef{@glswidestname\romannumeral#1}%
    {%
      \settowidth{\glstopicwidest}{%
       \glstopicSubNameFont{\csuse{@glswidestname\romannumeral#1}}%
       \glstopicSubItemSep
      }%
    }%
    {\setlength{\glstopicwidest}{0pt}}%
```

Save the value so that it doesn't have to keep being recalculated.

```
    \csedef{@glswidestlength\romannumeral#1}{\the\glstopicwidest}%
  }%
  {\setlength{\glstopicwidest}{\csuse{@glswidestlength\romannumeral#1}}}%
}
```

glstopicPreSkip

```
\newcommand*{\glstopicPreSkip}{\medskip}
```

glstopicMidSkip

```
\newcommand*{\glstopicMidSkip}{\smallskip}
```

lstopicPostSkip

```
\newcommand*{\glstopicPostSkip}{\smallskip}
```

glstopicSubItem

\glstopicSubItem{⟨*level*⟩}{⟨*label*⟩}{⟨*location list*⟩}

```
\newcommand*{\glstopicSubItem}[3]{%
  \glstopicSubItemBox{#1}{\glstopicSubNameFont{\glsentryitem{#2}%
    \glstarget{#2}{\glossentryname{#2}}}%
    \glstopicSubItemSep
  }%
  \ifglshassymbol{#2}{(\glossentrysymbol{#2})\space}{}%
  \ifglshasdesc{#2}%
   {\glossentrydesc{#2}\glspostdescription\glstopicSubPreLocSep}{}%
  \glstopicSubLoc{#2}{#3}%
}
```

topicSubItemSep

```
19522 \newcommand*{\glstopicSubItemSep}{\quad}
```

topicSubItemBox

\glstopicSubItemBox{⟨*level*⟩}{⟨*text*⟩}

```
19523 \newcommand*{\glstopicSubItemBox}[2]{%
19524   \ifdim\glstopicwidest>0pt\relax\makebox[\glstopicwidest][l]{#2}\else#2\fi
19525 }
```

opicSubNameFont

```
19526 \newcommand*{\glstopicSubNameFont}[1]{\textbf{#1}}
```

picSubPreLocSep

```
19527 \newcommand*{\glstopicSubPreLocSep}{\space}
```

\glstopicSubLoc

```
19528 \newcommand*{\glstopicSubLoc}[2]{#2}
```

\glstopicCols

```
19529 \newcommand*{\glstopicCols}{2}
```

glstopicColsEnv

```
19530 \newcommand*{\glstopicColsEnv}{multicols}
```

topicmcols

```
19531 \newglossarystyle{topicmcols}{%
19532   \renewenvironment{theglossary}%
19533   {%
19534     \glstopicInit
19535     \def\glstopic@prechildren{}%
19536     \def\glstopic@postchildren{}%
19537     \def\glstopic@prevlevel{-1}%
19538   }%
19539   {%
19540     \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
19541     \par
19542   }%
19543   \renewcommand*{\glossaryheader}{}%
19544   \renewcommand*{\glsgroupheading}[1]{%
19545     \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
19546     \def\glstopic@prevlevel{-1}%
19547     \glstopicGroupHeading{##1}%
19548   }%
19549   \renewcommand{\glossentry}[2]{%
```

```
    \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
    \def\glstopic@prevlevel{0}%
    \hangindent0pt\relax
    \parindent\glstopicParIndent\relax
    \glstopicItem{##1}{##2}%
    \ifnum\glstopicCols>1\relax
```

If there isn't a description, penalise a page break.

```
      \ifglshasdesc{##1}%
      {%
        \edef\glstopic@prechildren{%
          \noexpand\begin{\glstopicColsEnv}{\glstopicCols}%
        }%
      }%
      {%
        \edef\glstopic@prechildren{%
          \noexpand\nopagebreak
          \noexpand\begin{\glstopicColsEnv}{\glstopicCols}%
         }%
      }%
      \edef\glstopic@postchildren{\noexpand\end{\glstopicColsEnv}}%
    \fi
  }%
  \renewcommand{\subglossentry}[3]{%
    \ifnum\glstopic@prevlevel=0\relax\glstopic@prechildren\fi
    \def\glstopic@prevlevel{##1}%
    \glstopicAssignSubIndent{##1}%
    \glstopicSubItem{##1}{##2}{##3}%
  }%
  \renewcommand*{\glsgroupskip}{}%
}
```

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 4
0.2 (2015-11-30)
\Glsfmtshort: new 379
\glsfmtshort: new 378
\Glsfmtshortpl: new 379
\glsfmtshortpl: new 379
short: switched inline full form to short (long) 265
0.3 (2015-12-02)
\@ACRlong: added redefinition 97
\@ACRlongpl: added redefinition 98
\@ACRshort: added redefinition 95
\@ACRshortpl: added redefinition 96
\@Acrlong: added redefinition 96
\@Acrlongpl: added redefinition 98
\@Acrshort: added redefinition 94
\@Acrshortpl: added redefinition 95
\@GLSdesc@: added redefinition 90
\@GLSdescplural@: added redefinition . 91
\@GLSfirst@: added redefinition 88
\@GLSfirstplural@: added redefinition 89
\@GLSname@: added redefinition 90
\@GLSplural@: added redefinition 88
\@GLSsymbol@: added redefinition 91
\@GLSsymbolplural@: added redefinition 92
\@GLStext@: added redefinition 87
\@GLSuseri@: added redefinition 92
\@GLSuserii@: added redefinition 92
\@GLSuseriii@: added redefinition 93
\@GLSuseriv@: added redefinition 93
\@GLSuserv@: added redefinition 93
\@GLSuservi@: added redefinition 93
\@Glsdesc@: added redefinition 90
\@Glsdescplural@: added redefinition . 90
\@Glsfirst@: added redefinition 87
\@Glsfirstplural@: added redefinition 89
\@Glsname@: added redefinition 89
\@Glsplural@: added redefinition 88
\@Glssymbol@: added redefinition 91
\@Glssymbolplural@: added redefinition 91
\@Glstext@: added redefinition 87
\@Glsuseri@: added redefinition 92
\@Glsuserii@: added redefinition 92
\@Glsuseriii@: added redefinition 92
\@Glsuseriv@: added redefinition 93
\@Glsuserv@: added redefinition 93
\@Glsuservi@: added redefinition 93
\@acrlong: added redefinition 96
\@acrlongpl: added redefinition 97
\@acrshort: added redefinition 94
\@acrshortpl: added redefinition 95
\@gls@field@link: added optional argument 77
\@glsdescplural@: added redefinition . 90
\@glsfirst@: added redefinition 87
\@glsfirstplural@: added redefinition 88
\@glsplural@: added redefinition 88
\@glssymbolplural@: added redefinition 91
\@glsxtr@defaultnoglossarywarning: new 157
\@glsxtr@field@linkdefs: new 86
\@glsxtr@insertdots: new 229
\@print@glossary: added redefinition 154
\glsabbrvdefaultfont: renamed from \abbrvdefaultfont 235
\glsaccessdesc: new 187
\glsaccessdescplural: new 188
\glsaccessfirst: new 185
\glsaccessfirstplural: new 186
\Glsaccesslong: new 190
\glsaccesslong: new 190
\glsaccessname: new 183
\glsaccessplural: new 184
\Glsaccessshort: new 189
\glsaccessshort: new 189
\Glsaccessshortpl: new 189

\glsaccessshortpl: new 189
\glsaccesssymbol: new 186
\glsaccesssymbolplural: new 187
\glsaccesstext: new 184
\glsentryfmt: added check for short .. 77
\glslongpltok: new 229
\glsshortpltok: new 229
\glsxtr@newabbreviation: fixed family name in \setkeys 231
\glsxtrdiscardperiod: added check for plural 226
\GLSxtrlongpl: new 246
\Glsxtrlongpl: new 245
\glsxtrlongpl: new 245
\glsxtrNoGlossaryWarning: new 24
\glsxtrpostlinkAddDescOnFirstUse: new 226
\glsxtrpostlinkAddSymbolOnFirstUse: new 226
\glsxtrpostlinkendsentence: new . 225
\GLSxtrshortpl: new 244
\Glsxtrshortpl: new 243
\glsxtrshortpl: new 243
short-long-desc: fixed name to use \glslabeltok 257
long-short-desc: fixed name to use \glslabeltok 255
0.4 (2015-12-03)
\@glsxtr@doabbreviationsdef: added redefinition of \acronymtype 20
\Glsfmtshort: changed to use \Glsxtrshort 379
\glsfmtshort: changed to use \glsxtrshort 378
\Glsfmtshortpl: changed to use \glsxtrshortpl 379
\glsfmtshortpl: changed to use \glsxtrshortpl 379
\glsxtrifemptyglossary: new 34
\glsxtrnewnumber: added extra argument 206
\glsxtrnewsymbol: added extra argument 206
\MakeAcronymsAbbreviations: set the default type to \acronymtype 136
\newterm: fixed name argument 205
0.5 (2015-12-07)
\@cGLS: new 126
\@cGLS@: new 126
\@cGLSpl: new 127
\@cGLSpl@: new 127
\@glsxtr@setentrycountunsetattr: new 122
\cGLS: new 126
\cGLSformat: new 126
\cGLSpl: new 126
\cGLSplformat: new 127
\GlossariesExtraWarningNoLine: new 18
\glsenableentrycount: new 122
\glsfirstabbrvdefaultfont: new .. 235
\glsfirstlongdefaultfont: new ... 235
\Glsfmtfirst: new 382
\glsfmtfirst: new 382
\Glsfmtfirstpl: new 383
\glsfmtfirstpl: new 383
\Glsfmtplural: new 381
\glsfmtplural: new 381
\Glsfmtshort: changed to use \Glsxtrtitleshort 379
renamed from \Glsentryfmtshort . 379
\glsfmtshort: changed to use \glsxtrtitleshort 378
renamed from \glsentryfmtshort . 378
\Glsfmtshortpl: changed to use \Glsxtrtitleshortpl 379
renamed from \Glsentryfmtshortpl 379
\glsfmtshortpl: changed to use \glsxtrtitleshortpl 379
renamed from \glsentryfmtshortpl 379
\Glsfmttext: new 381
\glsfmttext: new 380
\glshasattribute: new 203
\glshascategoryattribute: new ... 202
\glsxtremsuffix: new 306
\GlsXtrEnableEntryCounting: new . 121
\glsxtrifcounttrigger: new 124
\glsxtrscfont: new 272
\glsxtrscsuffix: new 273
\glsxtrsmfont: new 289
\glsxtrsmsuffix: new 289
short-em: new 314
short-em-desc: new 315
short-em-footnote: new 325
short-em-long: new 310
short-em-long-desc: new 311

short-em-postfootnote: new 328
short-sc-footnote: new 284
short-sc-postfootnote: new 286
short-sm: new 293
short-sm-desc: new 295
short-sm-footnote: new 300
short-sm-long: new 291
short-sm-long-desc: new 293
short-sm-postfootnote: new 303
long-noshort-em: new 318
long-noshort-em-desc: new 322
long-noshort-sm: new 297
long-noshort-sm-desc: new 299
long-short-em: new 306
long-short-em-desc: new 307
long-short-sm: new 290
long-short-sm-desc: new 291
0.5.1 (2015-12-02)
\Glsaccesstext: new 184
0.5.1 (2015-12-07)
\@glsxtr@doaccsupp: new 23
General: removed \ifglsxtruseuchead 368
\Glsaccessdesc: new 188
\Glsaccessdescplural: new 188
\Glsaccessfirst: new 185
\Glsaccessfirstplural: new 186
\Glsaccessname: new 184
\Glsaccessplural: new 185
\Glsaccesssymbol: new 186
\Glsaccesssymbolplural: new 187
\Glsxtrheadfirst: now uses headuc attribute 373
\glsxtrheadfirst: now uses headuc attribute 373
\Glsxtrheadfirstplural: now uses headuc attribute 374
\glsxtrheadfirstplural: now uses headuc attribute 373
\Glsxtrheadplural: now uses headuc attribute 372
\glsxtrheadplural: now uses headuc attribute 372
\Glsxtrheadshort: now uses headuc attribute 369
\glsxtrheadshort: now uses headuc attribute 368
\Glsxtrheadshortpl: now uses headuc attribute 369
\glsxtrheadshortpl: now uses headuc attribute 368
\Glsxtrheadtext: now uses headuc attribute 371
\glsxtrheadtext: now uses headuc attribute 371
short-em-footnote: switch off regular attribute if set 326
short-em-footnote-desc: switch off regular attribute if set 327
short-long: switch off regular attribute if set 256
short-long-desc: switch off regular attribute if set 257
short-postfootnote-desc: switch off regular attribute if set 263
short-sc-footnote: switch off regular attribute if set 284
short-sc-footnote-desc: switch off regular attribute if set 286
short-sm-footnote: switch off regular attribute if set 301
short-sm-footnote-desc: switch off regular attribute if set 303
long-short: switch off regular attribute if set 253
long-short-desc: switch off regular attribute if set 255
long-short-sc-desc: switch off regular attribute if set 274
footnote: switch off regular attribute if set 258
postfootnote: switch off regular attribute if set 261
0.5.2 (2015-12-08)
\@GLSdesc@: added accessibility support 90
\@GLSdescplural@: added accessibility support 91
\@GLSfirst@: added accessibility support 88
\@GLSfirstplural@: added accessibility support 89
\@GLSname@: added accessibility support 90
\@GLSplural@: added accessibility support 88
\@GLSsymbol@: added accessibility support 91
\@GLSsymbolplural@: added accessibility support 92

\@GLStext@: added accessibility support 87
\@Glsdesc@: added accessibility support 90
\@Glsdescplural@: added accessibility support 90
\@Glsfirst@: added accessibility support 87
\@Glsfirstplural@: added accessibility support 89
\@Glsname@: add accessibility support . . 89
\@Glsplural@: added accessibility support 88
\@Glssymbol@: added accessibility support 91
\@Glssymbolplural@: added accessibility support 91
\@Glstext@: added accessibility support 87
\@glsdesc@: added accessibility support 90
\@glsdescplural@: added accessibility support 90
\@glsfirst@: added accessibility support 87
\@glsfirstplural@: added accessibility support 88
\@glsname@: added accessibility support 89
\@glsplural@: added accessibility support 88
\@glssymbol@: added accessibility support 91
\@glssymbolplural@: added accessibility support 91
\@glstext@: added accessibility support 86
\@glsxtr@activate@initialtagging: new 223
\@glsxtr@do@titlecaps@warn: new . 223
\@glsxtr@tag: new 223
General: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded 183
removed \glsxtrabbrvfmt 247
\glossaryentrynumbers: added 74
\Glossentrydesc: added 220
\Glossentryname: added 212
\Glossentrysymbol: added 221
\glossentrysymbol: added 220
\GLSaccessdesc: new 188, 199
\GLSaccessdescplural: new . . . 188, 199
\GLSaccessfirst: new 185, 198
\GLSaccessfirstplural: new . . 186, 198
\GLSaccesslong: new 190, 200
\GLSaccesslongpl: new 190, 200
\Glsaccesslongpl: new 190
\glsaccesslongpl: new 190
\GLSaccessname: new 184, 197
\GLSaccessplural: new 185, 197
\GLSaccessshort: new 189, 199
\GLSaccessshortpl: new 190, 199
\GLSaccesssymbol: new 187, 198
\GLSaccesssymbolplural: new . 187, 198
\GLSaccesstext: new 184, 197
\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here 77
\GlsXtrEnableInitialTagging: new 221
\glsxtrfieldtitlecase: new 207
\GlsXtrFormatLocationList: new . . . 75
\glsxtrnewabbrevpresetkeyhook: new 233
\glsxtrtagfont: new 223
\KV@printgloss@nonumberlist: added 76
\mfu@checkword@do: added 222
\setabbreviationstyle: added check for post-definition style switch 250
0.5.3 (2015-12-09)
\@glsxtr@autoindex@at: new 218
\@glsxtr@autoindex@encap: new . . . 218
\@glsxtr@autoindex@esc: new 219
\@glsxtr@autoindex@level: new . . . 219
\@glsxtr@autoindex@setname: new . 217
\@glsxtr@doabbreviationsdef: new . 20
General: removed \GlsXtrNoGlsWarningNoAutoMakeMain 156
\glsdescwidth: added 74
\glspagelistwidth: added 74
\glsxtrdoautoindexname: new 216
\glsxtrpostnamehook: new 213
\if@glsxtr@format@override: new . 215
\ProvidesGlossariesExtraLang: new 437
\RequireGlossariesExtraLang: new 437
0.5.4 (2015-12-15)
\@@newglossaryentry@defunitcounters: new 127
\@GLSxtr@p@acrlong@: new 113
\@GLSxtr@p@acrlongpl@: new 113
\@GLSxtr@p@acrshort@: new 112
\@GLSxtr@p@acrshortpl@: new 113
\@GLSxtr@p@long@: new 112
\@GLSxtr@p@longpl@: new 112

\@GLSxtr@p@plural@: new 111
\@GLSxtr@p@short@: new 111
\@GLSxtr@p@shortpl@: new 112
\@GLSxtr@p@text@: new 111
\@GlsXtrEnableOnTheFly: new 70
\@Glsxtr: new 71
\@Glsxtr@p@acrlong@: new 113
\@Glsxtr@p@acrlongpl@: new 113
\@Glsxtr@p@acrshort@: new 112
\@Glsxtr@p@acrshortpl@: new 112
\@Glsxtr@p@long@: new 112
\@Glsxtr@p@longpl@: new 112
\@Glsxtr@p@plural@: new 111
\@Glsxtr@p@short@: new 111
\@Glsxtr@p@shortpl@: new 111
\@Glsxtr@p@text@: new 110
\@Glsxtrpl: new 72
\@alt@gls@hyp@opt: new 105
\@gls@alt@hyp@opt: new 105
\@gls@alt@hyp@opt@char: new 105
\@gls@alt@hyp@opt@keys: new 105
\@gls@increment@currunitcount: new 128
\@gls@local@increment@currunitcount: new 129
\@gls@setdefault@glslink@opts: new 102
\@glsxtr: new 70
\@glsxtr@addunitcounter: new 128
\@glsxtr@currunitcount: new 129
\@glsxtr@ifunitcounter: new 128
\@glsxtr@p@acrlong@: new 113
\@glsxtr@p@acrlongpl@: new 113
\@glsxtr@p@acrshort@: new 112
\@glsxtr@p@acrshortpl@: new 112
\@glsxtr@p@long@: new 112
\@glsxtr@p@longpl@: new 112
\@glsxtr@p@plural@: new 111
\@glsxtr@p@short@: new 111
\@glsxtr@p@shortpl@: new 111
\@glsxtr@p@text@: new 110
\@glsxtr@prevunitcount: new 129
\@glsxtr@setentryunitcountunsetattr: new 133
\@glsxtr@unitcountlist: new 128
\@glsxtrpl: new 71
\@newglossaryentryposthook: added empty see value if not set and added 'see' to field key map 57
\@sGlsXtrEnableOnTheFly: new 70
\cGlsformat: added 127
\cglsformat: added 127
\cGlsplformat: added 127
\cglsplformat: added 127
\glsdisablehyper: added 109
\glsdonohyperlink: added 109
\glsenableentryunitcount: new ... 130
\glshasattribute: added check for entry's existence 203
\glsifattribute: added check for entry's existence 203
\glspostlinkhook: added existence check 224
\Glsxtr: new 71
\glsxtr: new 70
\glsxtrcat: new 70
\glsxtrdohyperlink: added 108
\glsxtrdowrglossaryhook: new 105
\GlsXtrEnableEntryUnitCounting: new 133
\GlsXtrEnableOnTheFly: new 69
\Glsxtrpl: new 71
\glsxtrpl: new 71
\glsxtrpostlocalreset: new 121
\glsxtrpostlocalunset: new 120
\glsxtrpostreset: new 120
\glsxtrpostunset: new 119
\glsxtrprotectlinks: new 110
\GlsXtrSetAltModifier: new 105
\GlsXtrSetDefaultGlsOpts: new ... 104
\glsxtrstarflywarn: new 70
\GlsXtrWarning: new 72
\MakeAcronymsAbbreviations: now disables \setacronymstyle 136

1.0 (2016-01-24)

\@glsxtr@autoindexcrossrefs: new . 17
\@glsxtr@idx@displaynumberlist: new 147
\@glsxtr@idx@entrynumberlist: new 149
\@glsxtr@noidx@displaynumberlist: new 147
\@glsxtr@noidx@entrynumberlist: new 148
\@glsxtr@noidx@numberlistloop: new 148
\@glsxtr@reg@glosslist: new 138
\makeglossaries: new 139

1.01 (2016-02-02)
\glsxtrdiscardperiod: added check for first use 226
short-desc: fixed typo in \glsxtrinlinefullformat and added missing second argument . . . 266
1.02 (2016-04-25)
\@glsxtr@current@style: new 73
\Glsfmtfull: new 386
\glsfmtfull: new 386
\Glsfmtfullpl: new 387
\glsfmtfullpl: new 386
\Glsfmtlong: new 384
\glsfmtlong: new 384
\Glsfmtlongpl: new 385
\glsfmtlongpl: new 385
\Glsxtrheadfull: new 377
\glsxtrheadfull: new 376
\Glsxtrheadfullpl: new 378
\glsxtrheadfullpl: new 377
\Glsxtrheadlong: new 375
\glsxtrheadlong: new 374
\Glsxtrheadlongpl: new 376
\glsxtrheadlongpl: new 375
\Glsxtrtitlefull: new 377
\glsxtrtitlefull: new 377
\Glsxtrtitlefullpl: new 378
\glsxtrtitlefullpl: new 377
\Glsxtrtitlelong: new 375
\glsxtrtitlelong: new 375
\Glsxtrtitlelongpl: new 376
\glsxtrtitlelongpl: new 375
short-postfootnote-desc: added redef of \glsxtrsetupfulldefs . . 264
\ifglsxtrinsertinside: new 253
postfootnote: added redef of \glsxtrsetupfulldefs 262
stylemods: new 24
1.03 (2016-04-27)
\@GLSfirstplural@: bug fix: misspelt cs name 89
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural 88
\@Glsfirstplural@: bug fix: misspelt cs name 89
\@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural 88
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural 88
\glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl . . 375
\glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl 369
1.04 (2015-04-30)
short-em-footnote: renamed from "footnote-em" 325
1.04 (2016-05-02)
\@@glsxtrpostloctag: new 76
\@GLSdesc@: set abbreviation and regular format 90
\@GLSdescplural@: set abbreviation and regular format 91
\@GLSfirst@: set abbreviation format . . 88
\@GLSfirstplural@: set abbreviation and regular format 89
\@GLSname@: set abbreviation and regular format 90
\@GLSplural@: set abbreviation and regular format 88
\@GLSsymbol@: set regular format 91
\@GLSsymbolplural@: set regular format 92
\@GLStext@: set abbreviation and regular format 87
\@GLSuseri@: set regular format 92
\@GLSuserii@: set regular format 92
\@GLSuseriii@: set regular format 93
\@GLSuseriv@: set regular format 93
\@GLSuserv@: set regular format 93
\@GLSuservi@: set regular format 93
\@Glsdesc@: set abbreviation and regular format 90
\@Glsdescplural@: set abbreviation and regular format 90
\@Glsfirst@: set abbreviation and regular format 87
\@Glsfirstplural@: set abbreviation and regular format 89
\@Glsname@: set abbreviation and regular format 89
\@Glsplural@: set abbreviation and regular format 88
\@Glssymbol@: set regular format 91
\@Glssymbolplural@: set regular format 91
\@Glstext@: set abbreviation and regular format 87
\@Glsuseri@: set regular format 92
\@Glsuserii@: set regular format 92
\@Glsuseriii@: set regular format 92

\@Glsuseriv@: set regular format 93
\@Glsuserv@: set regular format 93
\@Glsuservi@: set regular format 93
\@gls@preglossaryhook: added check for entry's existence 223
\@glsdesc@: set abbreviation and regular format 90
\@glsdescplural@: set abbreviation and regular format 90
\@glsfirst@: set abbreviation and regular format 87
\@glsfirstplural@: set abbreviation and regular format 88
\@glsname@: set abbreviation and regular format 89
\@glsplural@: set abbreviation and regular format 88
\@glssymbol@: set regular format 91
\@glssymbolplural@: set regular format 91
\@glstext@: set abbreviation and regular format 86
\@glsxtr@deprecated@abbrstyle: new 252
\@glsxtr@do@style: new 25
\@glsxtr@doloctag: new 76
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand 149
\@glsxtr@pagestag: new 76
\@glsxtr@pagetag: new 76
\@glsxtr@preloctag: new 76
\@glsxtrpostloctag: new 76
\@glsxtrpreloctag: new 75, 76
\glossentrydesc: added glossdescfont attribute check 208
\Glossentryname: added glossnamefont attribute check 212
\glossentryname: added glossnamefont attribute check 209
moved post name hook inside condition 212
\glsabbrvemfont: new 306
\glsabbrvuserfont: new 331
\glsfirstabbrvemfont: new 306
\glsfirstabbrvuserfont: new 331
\glsfirstlongemfont: new 306
\glsfirstlonguserfont: new 331
\glsifnotregularcategory: new . . . 204
\glslongdefaultfont: new 235
\glslongemfont: new 306
\glslongfont: new 235
\glslonguserfont: new 331
\glsxtrassignfieldfont: new 86
\GlsXtrEnablePreLocationTag: new . 75
\glsxtrfirstscfont: new 273
\glsxtrfirstsmfont: new 289
\glsxtrlongshortdescsort: new . . . 254
\glsxtrpostnamehook: added category check 213
\glsxtrregularfont: new 77
\glsxtruserfield: new 331
\glsxtruserparen: new 331
\glsxtrusersuffix: new 331
\GlsXtrWarnDeprecatedAbbrStyle: new 252
short-em-long-em: new 312
short-em-long-em-desc: new 313
short-em-nolong: new 315
short-em-nolong-desc: new 317
short-em-postfootnote: renamed from "postfootnote-em" 328
short-footnote: new 260
short-long-user: new 342
short-long-user-desc: new 343
short-nolong: new 266
short-nolong-desc: new 267
short-postfootnote: new 263
short-sc-footnote: renamed from "footnote-sc" 284
short-sc-nolong: new 278
short-sc-nolong-desc: new 280
short-sc-postfootnote: renamed from "postfootnote-sc" 286
short-sm-footnote: renamed from "footnote-sm" 300
short-sm-nolong: new 295
short-sm-nolong-desc: new 296
short-sm-postfootnote: renamed from "postfootnote-sm" 303
\letabbreviationstyle: new 251
\newabbreviationstyle: bug fix: corrected test for existence 250
long-em-noshort-em: new 319
long-em-noshort-em-desc: new 323
long-em-short-em: new 308
long-em-short-em-desc: new 309
long-noshort: new 272
long-noshort-desc: new 271

long-noshort-em: renamed from "long-em" 318
long-noshort-em-desc: renamed from "long-desc-em" 322
long-noshort-sc: renamed from "long-sc" 280
long-noshort-sc-desc: renamed from "long-desc-sc" 282
long-noshort-sm: renamed from "long-sm" 297
long-noshort-sm-desc: renamed from \long-desc-sm 299
long-short-user: new 332
long-short-user-desc: new 341
\renewabbreviationstyle: new 251
style: new 25
1.05 (2016-06-10)
\eglssetwidest: new 509
\glsFindWidestAnyName: new 511
\glsFindWidestAnyNameLocation: new 517
\glsFindWidestAnyNameSymbol: new 514
\glsFindWidestAnyNameSymbolLocation: new 515
\glsFindWidestLevelTwo: new 513
\glsFindWidestUsedAnyName: new . . 511
\glsFindWidestUsedAnyNameLocation: new 516
\glsFindWidestUsedAnyNameSymbol: new 514
\glsFindWidestUsedAnyNameSymbolLocation: new 515
\glsFindWidestUsedLevelTwo: new . 512
\glsFindWidestUsedTopLevelName: new 510
\glsfirstlongfootnotefont: new . . 258
\glsgetwidestname: new 510
\glsgetwidestsubname: new 510
\glslongfootnotefont: new 258
\glsxtrAltTreeIndent: new 508
\glsxtralttreeInit: new 508
\glsxtrAltTreePar: new 508
\glsxtrAltTreeSetHangIndent: new 518
\glsxtrAltTreeSetSubHangIndent: new 518
\glsxtralttreeSubSymbolDescLocation: new 508
\glsxtralttreeSymbolDescLocation: new 507
\glsxtrComputeTreeIndent: new . . . 517
\glsxtrComputeTreeSubIndent: new 517
\glsxtrtreetopindent: new 508
short-em-long: fixed incorrect font used by long form 311
\xglssetwidest: new 509
1.06 (2016-06-18)
\@glsdoifexistsorwarn: new 17
\@glsxtr@docdefval: new 16
\@glsxtr@usesee: new 58
General: disabled docdef key at the start of the document 33
docdef option changed to choice 16
\glsxtr@usesee: new 58
\glsxtrusesee: new 57
\glsxtruseseeformat: new 58
\if@glsxtrdocdefrestricted: new . . 17
1.07 (2016-08-15)
\@@glsxtrp: new 113
\@GLSfirst@: added check for nohyperfirst attribute 88
\@GLSfirstplural@: added check for nohyperfirst attribute 89
\@GLSxtrp: new 114
\@Glsfirst@: added check for nohyperfirst attribute 87
\@Glsfirstplural@: added check for nohyperfirst attribute 89
\@Glsxtrp: new 114
\@gls@preglossaryhook: added \glossxtrsetpopts 224
\@glsfirst@: added check for nohyperfirst attribute 87
\@glsfirstplural@: added check for nohyperfirst attribute 89
\@glsxtrinmark: new 366
\@glsxtrnotinmark: new 366
\@glsxtrp: new 114
\@glsxtrp@opt: new 113
\glossxtrsetpopts: new 113
\glsps: new 116
\glspt: new 116
\glsxtr@entry@p: new 115
\glsxtrabbrvfootnote: new 258
\glsxtrchecknohyperfirst: new 87
\glsxtrfieldtitlecasecs: new 207
\glsxtrifinmark: new 366
\GLSxtrp: new 117
\Glsxtrp: new 116

\glsxtrp: new 115
\glsxtrsetpopts: new 113
short-long-desc: added text key 257
fixed misspelling of \glsabbrvfont in plural key 257
long-short-desc: added missing text key 255
fixed misspelling of \glsabbrvfont . 255
footnote: changed first forms to use \glsfirstlongfootnotefont . . . 258
postfootnote: removed \footnote from first keys 261
switched from \glsfirstlongfont to \glsfirstlongfootnotefont . . . 262
\RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glsxtrifwasfirstuse 137
1.08 (2016-12-13)
\@@glsxtr@record: new 7
\@GLS@: added \@glsxtr@record 79
\@GLSpl@: added \@glsxtr@record . . . 79
\@Gls@: added \@glsxtr@record 78
\@Glspl@: added \@glsxtr@record . . . 79
\@gls@: added \@glsxtr@record 78
\@gls@@link@: added \@glsxtr@record 79
\@gls@field@link: added \@glsxtr@record 78
\@gls@saveentrycounter: new 32
\@glsdisp: added \@glsxtr@record . . 79
\@glspl@: added \@glsxtr@record . . . 78
\@glsxtr@dorecord: new 9
\@glsxtr@err@undefaction: new 5
\@glsxtr@record: new 6
\@glsxtr@warn@onexistsordo: new . . . 5
\@glsxtr@warn@undefaction: new 5
\@print@unsrt@glossary: new 166
record: added record package option . . . 15
\glsadd: added \@glsxtr@record 85
\glsdoifexists: now defines \glslabel 56
\glsxtr@@do@wrglossary: new 31
\glsxtr@addloclistfield: new 12
\glsxtr@indexonly@saveentrycounter: new 12
\glsxtr@record: new 162
\glsxtr@resource: new 160
\glsxtr@saveentrycounter: new 32
\glsxtr@setup@record: new 12
\glsxtrassignfieldfont: added check for existence 86
\glsxtrresourcefile: new 158
\printunsrtglossaries: new 165
\printunsrtglossary: new 165
1.09 (2016-12-16)
\@glsxtr@gettype: new 147
\@glsxtr@mixed@assign@sortkey: new 147
\@printglossary: redefined to save options 145
\glsxtr@makeglossaries: new 147
1.10 (2016-12-17)
\@GLSpl@: fixed bug caused by typo in command name 79
1.11 (2017-01-19)
\@glsxtr@do@redef@forglsentries: new 5
\@glsxtr@noidx@do: new 172
\@glsxtr@redef@forglsentries: new . 5
\@glsxtr@shortcutsval: new 22
\@glsxtr@unsrt@getgrouptitle: new 171
\@print@noidx@glossary: added redefinition 150
\glsxtr@addloclistfield: added group key 13
added location key 13
\glsxtr@fields: new 160
\glsxtr@linkprefix: new 160
\glsxtr@org@newignoredglossary: new 53
\glsxtr@s@newignoredglossary: new 53
\glsxtr@shortcutsval: new 160
\glsxtr@texencoding: new 160
\glsxtr@writefields: new 160
\GlsXtrLoadResources: new 160
\glsxtrpageref: new 49
\glsxtrresourcefile: changed extension to .glstex 158
\newignoredglossary: added starred version 52
1.12 (2017-02-03)
\@@glsxtr@recordcounter: new 12
\@gls@preglossaryhook: check for definition 223
\@glsxtr@counterrecordhook: new . 162
\@glsxtr@display@loc: new 151
\@glsxtr@docounterrecord: new . . . 163

\@glsxtr@longnewglossaryentry: new 52
\@glsxtr@noop@recordcounter: new . 12
\@glsxtr@op@recordcounter: new ... 12
\@glsxtr@provide@storagekey: new . 34
\@glsxtr@s@longnewglossaryentry: new 51
\@glsxtrentryfmt: new 37
\@glsxtrindexaliased: new 103
\@glsxtrsetaliasnoindex: new 103
\@newglossaryentryposthook: added check for alias key 65
\@no@glsxtrindexaliased: new 103
\@printunsrtglossary: new 165
General: added target key to printgloss family 145
\apptoglossarypreamble: new 50
\csGlsXtrLetField: new 45
\eGlsXtrSetField: new 45
\gGlsXtrSetField: new 45
\glsnoidxdisplayloc: added redefinition 151
\glssettoctitle: added patch 54
\glsxtr@counterrecord: new 162
\glsxtr@langtag: new 160
\glsxtr@newabbreviation: new 230
\glsxtr@org@newignoredglossary: Added check for existence 53
\glsxtr@pluralsuffixes: new 160
\glsxtr@provideignoredglossary: new 54
\glsxtr@s@newignoredglossary: Added check for existence 53
\glsxtr@s@provideignoredglossary: new 55
\glsxtrabbrvpluralsuffix: new ... 235
\glsxtralias: new 65
\glsxtrcopytoglossary: new 55
\glsxtrdeffield: new 44
\glsxtrdisplayendloc: new 152
\glsxtrdisplayendlochook: new ... 152
\glsxtrdisplaysingleloc: new 152
\glsxtrdisplaystartloc: new 152
\glsxtrdohyperlink: added check for alias field 108
\glsxtredeffield: new 44
\glsxtrentryfmt: new 36
\glsxtrfielddolistloop: new 38
\glsxtrfieldforlistloop: new 38
\glsxtrfieldifinlist: new 38
\glsxtrfieldlistadd: new 37
\glsxtrfieldlisteadd: new 37
\glsxtrfieldlistgadd: new 37
\glsxtrfieldlistxadd: new 37
\glsxtrfieldxifinlist: new 38
\glsxtrfmt: new 35
\GlsXtrFmtDefaultOptions: new 35
\GlsXtrFmtField: new 35
\glsxtrifkeydefined: new 34
\glsxtrindexaliased: new 104
\GlsXtrLetField: new 45
\GlsXtrLetFieldToField: new 45
\GlsXtrLoadResources: removed restriction on only one per document 160
\glsxtrlocrangefmt: new 152
\glsxtrpostlongdescription: new .. 52
\glsxtrprovidestoragekey: new 34
\GlsXtrRecordCounter: new 162
\glsxtrresourcecount: new 160
\glsxtrresourcefile: added catcode change for @ 159
\glsxtrsetaliasnoindex: new 103
\GlsXtrSetField: new 45
\glsxtrsetfieldifexists: new 45
\glsxtrunsrtdo: new 171
\Glsxtrusefield: new 43
\glsxtrusefield: new 43
short-postlong-user: new 338
short-postlong-user-desc: new ... 340
\longnewglossaryentry: added starred version 51
long-postshort-user: new 333
long-postshort-user-desc: new ... 337
postdot: new 18
\pretoglossarypreamble: new 50
\print@noop@unsrtglossaryunit: new 171
\print@op@unsrtglossaryunit: new 170
\printunsrtglossary: added starred form 165
\printunsrtglossaryhandler: new . 170
\printunsrtglossaryunit: new 12
\printunsrtglossaryunitsetup: new 171
\provideignoredglossary: new 54
\s@glsxtr@provide@storagekey: new 35
\s@printunsrtglossary: new 165
\xGlsXtrSetField: new 45

1.13 (2017-02-07)
\@glsdisp: removed \@glsxtr@org@glsdisp 79
\glsxtrsetaliasnoindex: switched to \providecommand 103
1.14 (2017-04-18)
\@gls@link: added redefinition 82
\@gls@noidx@getgrouptitle: new .. 149
\@gls@removespaces: new 153
\@glsxtr@do@automake@err: new ... 162
\@glsxtr@org@gloautosee: new 30
\@glsxtr@record: added third arg 6
\@glsxtr@recordsee: new 12
General: added \glsadd option theHvalue 85
added \glsadd option thevalue 85
\glsdisablehyper: added redefinition 109
\glsenableentrycount: fixed assignment of \@cGls@ 123
\glsenableentryunitcount: fixed assignment of \@cGls@ 131
\glsnavigation: new 150
\glsxtr@org@getgrouptitle: new .. 149
\glsxtr@recordsee: new 6
\glsxtr@writefields: added check for automake 161
\glsxtrdisplayendloc: added check for empty format 152
\glsxtrgetgrouptitle: new 149
\glsxtrinitwrgloss: new 80
\glsxtrlocationhyperlink: new ... 153
\glsxtrsetgrouptitle: new 150
\glsxtrsupphypernumber: new 153
\ifglsxtrwrglossbefore: new 80
1.15 (2017-05-10)
\@glsxtr@dorecord: corrected premature expansion of \@glslocref 9
short-em-long-em: fixed spelling of \glsabbrvfont 312
short-long: fixed spelling of \glsabbrvfont 256
short-long-user: fixed spelling of \glsabbrvfont 342
short-postfootnote-desc: fixed spelling of \glsabbrvfont 263
short-postlong-user: fixed spelling of \glsabbrvfont 339
short-postlong-user-desc: fixed spelling of \glsabbrvfont 340
long-em-short-em: fixed spelling of \glsabbrvfont 309
long-postshort-user: fixed spelling of \glsabbrvfont 333
long-postshort-user-desc: fixed spelling of \glsabbrvfont 337
long-short: fixed spelling of \glsabbrvfont 253
long-short-user: fixed spelling of \glsabbrvfont 332
footnote: fixed spelling of \glsabbrvfont 258
postfootnote: fixed spelling of \glsabbrvfont 261
1.16 (2017-06-15)
\@glo@autosee: added redefinition 30
\@gls@noidx@getgrouptitle: fixed bug 149
\@glsxtr@addunusedxrefs: added check for seealso field 66
\@glsxtr@checkgroup: use \csuse instead of \csname 172
\@glsxtr@dorecordnodefer: new 10
\@glsxtr@record@only@setup: added check for \@gls@setupsort@none .. 15
\@print@unsrt@glossary: corrected misspelt command 166
\@printunsrt@glossary@handler: new 170
\gls@checkseeallowed: added redefinition 31
\glsxtr@writefields: added \providecommand lines 160
\glsxtrautoindex: new 217
\glsxtrautoindexassignsort: new . 217
\glsxtrautoindexentry: new 217
\glsxtrindexseealso: new 62
\glsxtrseealsolabels: new 65
\glsxtrseelist: new 61
\glsxtruseseealso: new 60
\glsxtruseseealsoformat: new 61
\seealsoname: new 62
autoseeindex: new 17
1.17 (2017-08-09)
\@glsxtr@mark@wordseps: new 230
\@glsxtr@markwordseps: new 230
\@glsxtr@noidx@displaynumberlist: replace hard-coded ?? with \glsxtrundeftag 148

\@glsxtr@noidx@entrynumberlist: replace hard-coded ?? with \glsxtrundeftag 149
\@glsxtr@noidx@numberlistloop: replace hard-coded ?? with \glsxtrundeftag 148
\@glsxtrifhyphenstart: new 344
General: removed some inconsistencies in the abbreviation styles 253
\glsabbrvhyphenfont: new 344
\glsabbrvonlyfont: new 359
\glsabbrvscfont: new 272
\glsabbrvsmfont: new 289
\glsabbrvuserfont: initialised to default font 331
\glsfirstabbrvhyphenfont: new . . . 345
\glsfirstabbrvonlyfont: new 359
\glsfirstabbrvscfont: new 273
\glsfirstabbrvsmfont: new 289
\glsfirstlonghyphenfont: new 345
\glsfirstlongonlyfont: new 359
\glslonghyphenfont: new 345
\glslongonlyfont: new 359
\glslonguserfont: initialised to default font 331
\glsxtr@newabbreviation: added \glsxtrorgshort and \glsxtrorglong 231
\GlsXtrDefineAcShortcuts: new 21
\glsxtrgenabbrvfmt: added check for \ifglsxtrinsertinside 247
\glsxtrhyphensuffix: new 345
\glsxtrifhyphenstart: new 344
\glsxtrlonghyphen: new 349
\glsxtrlonghyphennoshort: new . . . 347
\glsxtrlonghyphenshort: new 344
\glsxtrlongshortdescname: new . . . 254
\glsxtronlydescname: new 361
\glsxtronlydescsort: new 361
\glsxtronlysuffix: new 359
\glsxtrparen: new 234
\glsxtrposthyphenlong: new 355
\glsxtrposthyphenshort: new 350
\glsxtrposthyphensubsequent: new 350
\glsxtrshortdescname: new 266
\glsxtrshorthyphen: new 355
\glsxtrshorthyphenlong: new 353
\glsxtrshortlongdescname: new . . . 257
\glsxtrshortlongdescsort: new . . . 257
\Glsxtrsubsequentfmt: new 249
\glsxtrsubsequentfmt: new 249
\Glsxtrsubsequentplfmt: new 249
\glsxtrsubsequentplfmt: new 249
\glsxtrword: new 230
\glsxtrwordsep: new 230
short-hyphen-long-hyphen: new . . . 353
short-hyphen-long-hyphen-desc: new 354
short-hyphen-postlong-hyphen: new 356
short-hyphen-postlong-hyphen-desc: new 358
short-long-user-desc: corrected first forms 343
short-nolong-desc-noreg: new 268
short-nolong-noreg: new 266
long-em-noshort-em-desc-noreg: new 325
long-em-noshort-em-noreg: new . . . 321
long-hyphen-noshort-desc-noreg: new 347
long-hyphen-noshort-noreg: new . . 349
long-hyphen-postshort-hyphen: new 350
long-hyphen-postshort-hyphen-desc: new 352
long-hyphen-short-hyphen: new . . . 345
long-hyphen-short-hyphen-desc: new 346
long-noshort-desc-noreg: new 271
long-noshort-noreg: new 272
long-only-short-only: new 359
long-only-short-only-desc: new . . 361
long-short-user-desc: corrected first forms 341
1.18 (2017-08-10)
stylemods: changed default value to "default" 24
1.19 (2017-09-09)
\@glsxtr@defaultnumberformat: new . 6
\@glsxtr@dorecord: Use \@glsrecordlocref instead of \@glslocref 9
\@glsxtr@dorecordnodefer: Use \theglsentrycounter for the location rather than \@glslocref . . 10
\@glsxtr@record@setting: new 13
\@glsxtr@record@setting@alsoindex: new 13
\@glsxtrifhasfield: new 41

General: added \glslink option theHvalue 80
added \glslink option thevalue . . . 80
\glsxtr@writefields: removed double-quotes around \jobname . . 161
\glsxtrdoautoindexname: changed format test 216
\glsxtrhyperlink: new 109
\glsxtrifhasfield: new 41
\GlsXtrSetDefaultNumberFormat: new 6
\s@glsxtrifhasfield: new 41
1.20 (2017-09-11)
\@glsxtrhypernameprefix: new 145
\glsdohypertarget: added redefinition 146
\printunsrtglossaryunitsetup: switched from redefining \glolinkprefix to \@glsxtrhypernameprefix 171
1.21 (2017-11-03)
\@@glsxtr@record: added check for default options 8
\@@glsxtrwrglossmark: new 26
\@glslink: changed \let to \def 109
\@glsxtr@checkgroup: new 172
\@glsxtr@defpostpunc: new 18
\@glsxtr@do@record@wrglossary: new 7
\@glsxtr@dosee@alsoindex@glossary: new 30
\@glsxtr@doseeglossary: new 30
\@glsxtr@noidx@do: removed code dealing with the group 173
\@glsxtr@record@setting@off: new . 14
\@glsxtr@record@setting@only: new 13
\@glsxtr@rglstrigger@record: new 177
\@glsxtrglossentry: new 163
\@glsxtrnewgls: new 174
\@glsxtrsetaliasnoindex: changed to use \glsxtrifhasfield instead of \ifglshasfield 103
\@glsxtrwrglossmark: new 26
\@rGLS: new 180
\@rGLS@: new 180
\@rGLSpl: new 180
\@rGLSpl@: new 180
\@rGls: new 179
\@rGls@: new 179
\@rGlspl: new 180
\@rGlspl@: new 180
\@rgls: new 178
\@rgls@: new 178
\@rglspl: new 179
\@rglspl@: new 179
General: adjusted mcolalttree 524
modified index to remove hard coded \space 501
modified list to remove hard coded \space 489
moved conditional outside of \glsgroupskip 493–500
new 527
redefined altlistgroup to discourage breaks after group headings 491
redefined altlisthypergroup to discourage breaks after group headings 492
redefined alttreegroup to discourage breaks after group headings 519
redefined alttreehypergroup to discourage breaks after group headings 519
redefined indexgroup to discourage breaks after group headings 502
redefined indexhypergroup to discourage breaks after group headings 502
redefined listgroup to discourage breaks after group headings 491
redefined listhypergroup to discourage breaks after group headings 491
redefined mcolalttreegroup to discourage breaks after group headings 524
redefined mcolalttreehypergroup to discourage breaks after group headings 525
redefined mcolalttreespannav to discourage breaks after group headings 525
redefined mcolindexgroup to discourage breaks after group headings 520
redefined mcolindexhypergroup to discourage breaks after group headings 521

redefined mcolindexspannav to discourage breaks after group headings 521
redefined mcoltreegroup to discourage breaks after group headings 521
redefined mcoltreehypergroup to discourage breaks after group headings 522
redefined mcoltreenonamegroup to discourage breaks after group headings 523
redefined mcoltreenonamehypergroup to discourage breaks after group headings 523
redefined mcoltreenonamespannav to discourage breaks after group headings 523
redefined mcoltreespannav to discourage breaks after group headings 522
redefined treegroup to discourage breaks after group headings 505
redefined treehypergroup to discourage breaks after group headings 505
redefined treenonamegroup to discourage breaks after group headings 507
redefined treenonamehypergroup to discourage breaks after group headings 507
debug: new 27
\gglssetwidest: new 509
\glsdisablehyper: added check for existence 109
changed to use \def rather than \let 109
\glsenablehyper: changed to use \def rather than \let 109
\Glsfmtname: new 380
\glsfmtname: new 379
\glshex: new 439
\glslistchildpostlocation: new . . 489
\glslistchildprelocation: new . . . 489
\glslistprelocation: new 489
\glsnavhyperlink: patched 106
\glsseeitemformat: new 58
\glsshowtarget: new 29
\glstreechildprelocation: new . . . 501
\glstreeprelocation: new 501
\glstriggerrecordformat: new 178
\glsuseabbrvfont: new 247
\glsuselongfont: new 247
\glsxtr@do@alsoindex@wrglossary: new 7
\glsxtr@org@@do@wrglossary: new . . 31
\glsxtr@org@dohyperlink: new 106
\glsxtr@setbookindexmark: new . . . 532
\glsxtrbookindexatendgroup: new . 528
\glsxtrbookindexbetween: new 528
\glsxtrbookindexbookmark: new . . . 528
\glsxtrbookindexcols: new 527
\glsxtrbookindexcolspread: new . . 529
\glsxtrbookindexfirstmark: new . . 533
\glsxtrbookindexfirstmarkfmt: new 533
\glsxtrbookindexformatheader: new 528
\glsxtrbookindexgroupskip: new . . 528
\glsxtrbookindexlastmark: new . . . 533
\glsxtrbookindexlastmarkfmt: new 533
\glsxtrbookindexmarkentry: new . . 532
\glsxtrbookindexname: new 527
\glsxtrbookindexparentchildsep: new 528
\glsxtrbookindexparentsubchildsep: new 528
\glsxtrbookindexprelocation: new 527
\glsxtrbookindexsubatendgroup: new 528
\glsxtrbookindexsubbetween: new . 528
\glsxtrbookindexsubname: new 527
\glsxtrbookindexsubprelocation: new 527
\glsxtrbookindexsubsubatendgroup: new 528
\glsxtrbookindexsubsubbetween: new 528
\glsxtrbookindexthepage: new 532
\glsxtrdetoklocation: new 176
\glsxtrenablerecordcount: new . . . 176
\glsxtrglossentry: new 163
\glsxtrgroupfield: new 171
\Glsxtrheadname: new 370
\glsxtrheadname: new 370
\GlsXtrIfFieldEqStr: new 46
\glsxtriflabelinlist: new 170
\glsxtrifrecordtrigger: new 177

\glsxtrindexseealso: added check that the entry exists 63
\glsxtrinithyperoutside: new 81
\GlsXtrLocationRecordCount: new . 176
\glsxtrnewgls: new 173, 175
\glsxtrnewGLSlike: new 175
\glsxtrnewglslike: new 175
\glsxtrnewrgls: new 175
\glsxtrnewrGLSlike: new 176
\glsxtrnewrglslike: new 176
\glsxtrprelocation: new 488, 527
\GlsXtrRecordCount: new 176
\glsxtrrecordtriggervalue: new . . 177
\glsxtrresourcefile: now disables record key 159
\glsxtrresourceinit: new 159
\GlsXtrSetRecordCountAttribute: new 177
\Glsxtrtitlename: new 370
\glsxtrtitlename: new 370
\glsxtrtitleorpdforheading: new . 366
\GlsXtrTotalRecordCount: new 176
\glsxtrwrglossmark: new 26
short-em: new 315
short-sc: corrected first letter uppercasing 277
short-sm: corrected first letter uppercasing 294
shortcuts: ac 23
\ifglsxtr@hyperoutside: new 80
all: new 487
nolong-short: new 268
nolong-short-em: new 317
nolong-short-noreg: new 269
nolong-short-sc: new 280
nolong-short-sm: new 296
nopostdot: new 19
postpunc: new 19
\printunsrtglossaryentryprocesshook: new 170
\printunsrtglossarypredoglossary: new 170
\printunsrtglossaryskipentry: new 170
\rGLS: new 180
\rGls: new 179
\rgls: new 178
\rGLSformat: new 181
\rGlsformat: new 181
\rglsformat: new 181
\rGLSpl: new 180
\rGlspl: new 179
\rglspl: new 179
\rGLSplformat: new 181
\rGlsplformat: new 181
\rglsplformat: new 181
\s@glsxtrifhasfield: switched from \ifdef to \ifundef 41
1.22 (2017-11-08)
\@glsxtr@nopostpunc: new 144
\@glsxtr@orgprintglossary: changed explicit \let for \nopostdesc to \glsxtractivatenopost 143
\@glsxtrglossentryother: new 164
\glossentrynameother: new 214
\glsseeitemformat: switched check from regular to short 58
\glsxtr@setaccessdisplay: new . . . 213
\glsxtr@writefields: provide \glsxtr@record in aux file 161
\glsxtractivatenopost: new 144
\glsxtrbookindexprelocation: removed check for no post dot 527
\glsxtrglossentryother: new 164
\glsxtrnopostpunc: new 144
1.23 (2017-11-12)
\@@glsxtrfmt: added check for indexing 36
added grouping 36
new 36
\@glsxtr@nopostpunc@postdesc: new 144
\@glsxtr@restore@postpunc: new . . 145
\@glsxtrentryfmt: fixed missing label argument 37
\@glsxtrfmt: new 35
\eglsupdatewidest: new 509
\gglsupdatewidest: new 509
\glsupdatewidest: new 509
\GlsXtrDefineAbbreviationShortcuts: changed \newabbr definition to use \providecommand 21
\GlsXtrDefineAcShortcuts: changed \newabbr definition to use \providecommand 21
\glsxtrfmtdisplay: new 36
\glsxtrifcustomdiscardperiod: new 225
\GlsXtrIfFieldUndef: new 43
\glsxtrrestorepostpunc: new 145
\s@@glsxtrfmt: new 36
\s@glsxtrfmt: new 36

\xglsupdatewidest: new 510
1.24 (2017-11-14)
\glsadd: added \@gls@setsort 85
\glsxtrforcsvfield: new 38
\glsxtrlocalsetgrouptitle: new .. 150
1.25 (2017-11-14)
\glsxtrbookindexmulticolsenv: new 529
1.25 (2017-11-24)
\glsextrapostnamehook: new 213
\glsxtrfootnotename: new 258
\glsxtrlongnoshortdescname: new . 269
\glsxtrlongnoshortname: new 271
\glsxtrlongshortname: new 253
\glsxtrlongshortuserdescname: new 336
\glsxtronlyname: new 359
\glsxtrpostlinkAddDescOnFirstUse: changed to use \glsxtrparen 226
\glsxtrpostlinkAddSymbolOnFirstUse: changed to use \glsxtrparen 226
\glsxtrshortlongname: new 255
\glsxtrshortlonguserdescname: new 340
\glsxtrshortnolongname: new 264
1.26 (2018-01-05)
\@glsxtr@do@inc@linkcount: new .. 182
\glslinkpresetkeys: new 81
\glsxtr@inc@linkcount: new 81
\GlsXtrEnableLinkCounting: new .. 183
\GlsXtrIfLinkCounterDef: new 183
\glsxtrinclinkcounter: new 182
\GlsXtrLinkCounterName: new 183
\GlsXtrLinkCounterValue: new 182
\GlsXtrTheLinkCounter: new 182
1.27 (2018-02-26)
\@glsxtrdialecthook: new 33
General: added glossaries-extra-bib2gls.sty 437
\Alpha: new 454
\Beta: new 454
\Chi: new 454
\Digamma: new 455
\Epsilon: new 454
\Eta: new 454
\glsxtr@loaddialect: new 437
\glsxtrBasicDigitrules: new 484
\glsxtrcombiningdiacriticIIIrules: new 458
\glsxtrcombiningdiacriticIIrules: new 458
\glsxtrcombiningdiacriticIrules: new 457
\glsxtrcombiningdiacriticIVrules: new 459
\glsxtrcombiningdiacriticrules: new 457
\glsxtrcontrolrules: new 456
\glsxtrcurrencyrules: new 461
\glsxtrdigitrules: new 484
\glsxtrfractionrules: new 485
\glsxtrGeneralLatinIIIrules: new 463
\glsxtrGeneralLatinIIrules: new . 462
\glsxtrGeneralLatinIrules: new .. 462
\glsxtrGeneralLatinIVrules: new . 464
\glsxtrGeneralLatinVIIIrules: new 467
\glsxtrGeneralLatinVIIrules: new 466
\glsxtrGeneralLatinVIrules: new . 465
\glsxtrGeneralLatinVrules: new .. 464
\glsxtrgeneralpuncIIrules: new .. 461
\glsxtrgeneralpuncIrules: new ... 460
\glsxtrgeneralpuncrules: new 460
\glsxtrhyphenrules: new 460
\glsxtrLatinA: new 467
\glsxtrLatinAA: new 469
\glsxtrLatinAELigature: new 469
\glsxtrLatinE: new 467
\glsxtrLatinEszettSs: new 469
\glsxtrLatinEszettSz: new 469
\glsxtrLatinEth: new 469
\glsxtrLatinH: new 467
\glsxtrLatinI: new 468
\glsxtrLatinInsularG: new 470
\glsxtrLatinK: new 468
\glsxtrLatinL: new 468
\glsxtrLatinLslash: new 470
\glsxtrLatinM: new 468
\glsxtrLatinN: new 468
\glsxtrLatinO: new 468
\glsxtrLatinOELigature: new 469
\glsxtrLatinOslash: new 470
\glsxtrLatinP: new 468
\glsxtrLatinS: new 468
\glsxtrLatinSchwa: new 469
\glsxtrLatinT: new 468
\glsxtrLatinThorn: new 469
\glsxtrLatinWynn: new 470
\glsxtrLatinX: new 469
\glsxtrMathGreekIIrules: new 476
\glsxtrMathGreekIrules: new 475

\glsxtrMathItalicAlpha: new 480
\glsxtrMathItalicBeta: new 480
\glsxtrMathItalicChi: new 483
\glsxtrMathItalicDelta: new 481
\glsxtrMathItalicEpsilon: new ... 481
\glsxtrMathItalicEta: new 481
\glsxtrMathItalicGamma: new 481
\glsxtrMathItalicGreekIIrules: new 472
\glsxtrMathItalicGreekIrules: new 471
\glsxtrMathItalicIota: new 481
\glsxtrMathItalicKappa: new 482
\glsxtrMathItalicLambda: new 482
\glsxtrMathItalicLowerGreekIIrules: new 474
\glsxtrMathItalicLowerGreekIrules: new 473
\glsxtrMathItalicMu: new 482
\glsxtrMathItalicNabla: new 484
\glsxtrMathItalicNu: new 482
\glsxtrMathItalicOmega: new 483
\glsxtrMathItalicOmicron: new ... 482
\glsxtrMathItalicPartial: new ... 484
\glsxtrMathItalicPhi: new 483
\glsxtrMathItalicPi: new 482
\glsxtrMathItalicPsi: new 483
\glsxtrMathItalicRho: new 482
\glsxtrMathItalicSigma: new 483
\glsxtrMathItalicTau: new 483
\glsxtrMathItalicTheta: new 481
\glsxtrMathItalicUpperGreekIIrules: new 473
\glsxtrMathItalicUpperGreekIrules: new 472
\glsxtrMathItalicUpsilon: new ... 483
\glsxtrMathItalicXi: new 482
\glsxtrMathItalicZeta: new 481
\glsxtrMathUpGreekIIrules: new .. 471
\glsxtrMathUpGreekIrules: new ... 470
\glsxtrnonprintablerules: new ... 457
\glsxtrprovidecommand: new 439
\glsxtrspacerules: new 457
\glsxtrSubScriptDigitrules: new . 484
\glsxtrSuperScriptDigitrules: new 484
\glsxtrUpAlpha: new 477
\glsxtrUpBeta: new 477
\glsxtrUpChi: new 480
\glsxtrUpDelta: new 477
\glsxtrUpDigamma: new 478
\glsxtrUpEpsilon: new 478
\glsxtrUpEta: new 478
\glsxtrUpGamma: new 477
\glsxtrUpIota: new 478
\glsxtrUpKappa: new 478
\glsxtrUpLambda: new 479
\glsxtrUpMu: new 479
\glsxtrUpNu: new 479
\glsxtrUpOmega: new 480
\glsxtrUpOmicron: new 479
\glsxtrUpPhi: new 480
\glsxtrUpPi: new 479
\glsxtrUpPsi: new 480
\glsxtrUpRho: new 479
\glsxtrUpSigma: new 479
\glsxtrUpTau: new 480
\glsxtrUpTheta: new 478
\glsxtrUpUpsilon: new 480
\glsxtrUpXi: new 479
\glsxtrUpZeta: new 478
\Iota: new 454
\Kappa: new 454
\Mu: new 454
\Nu: new 454
\Omicron: new 454
\omicron: new 455
\Rho: new 454
\Tau: new 454
\Upalpha: new 455
\Upbeta: new 455
\Upchi: new 456
\Upepsilon: new 455
\Upeta: new 455
\Upiota: new 455
\Upkappa: new 455
\Upmu: new 455
\Upnu: new 455
\Upomicron: new 455
\upomicron: new 456
\Uprho: new 455
\Uptau: new 455
\Upzeta: new 455
\Zeta: new 454

1.28 (2018-03-06)

\@glsxtr@docdefval: changed from count register to macro 16
\@glsxtrdialecthook: save and restore \TrackLangRequireDialectPrefix 485

\glsxtredeffield: changed \csedef to \protected@csedef 44
\glsxtrlocalsetgrouptitle: changed \csedef \protected@csedef 150
\glsxtrsetgrouptitle: changed \csxdef \protected@csxdef 150
1.29 (2018-04-09)
\@gls@removespaces: added expansion 153
\@glsxtr@dorecord: don't suppress expansion of \@glsrecordlocref if counter isn't page 10
\@glsxtr@wrglossary@locationhyperlink: new 25
\glsxtr@inc@wrglossaryctr: new . . . 25
\glsxtr@wrglossarylocation: new . 440
\GlsXtrBibTeXEntryAliases: new . . 441
\glsxtrfieldforlistloop: corrected argument order in \forlistcsloop 38
\GlsXtrIndexCounterLink: new 440
\GlsXtrInternalLocationHyperlink: new 25
\GlsXtrProvideBibTeXFields: new . 441
indexcounter: new 26
\setentrycounter: new 152
1.30 (2018-04-25)
\@@glsxtr@record: added check for post-key hook 8
added check for pre-key hook 8
\@GLSxtr@fullpl: added \@glsxtr@record 239
\@GlsXtrStopUnsetBuffering: new . 120
\@Glsxtr@fullpl: added \@glsxtr@record 238
\@glsxtr@dorecord: don't suppress expansion of \@glsrecordlocref . . 10
\@glsxtr@full: added \@glsxtr@record 236
\@glsxtr@fullpl: added \@glsxtr@record 238
\@glsxtr@glossadd@postkeys: new . . . 9
\@glsxtr@glossadd@prekeys: new 9
\@glsxtr@glslink@postkeys: new 9
\@glsxtr@glslink@prekeys: new 9
\@glsxtr@local@textformat: new . . . 80
\@glsxtr@unset: new 118
\@glsxtrbuffer@unset: new 119
\glsadd: added \glsaddpostsetkeys . 85
added \glsaddpresetkeys 85
\glsaddpostsetkeys: new 85
\glsaddpresetkeys: new 85
\glsuserdescription: new 331
\glsxtrabbreviationfont: new 77
\GlsXtrDualBackLink: new 440
\GlsXtrDualField: new 440
\GlsXtrExpandedFmt: new 81
\GLSxtrlong: added \@glsxtr@record 242
\Glsxtrlong: added \@glsxtr@record 242
\glsxtrlong: added \@glsxtr@record 241
\GLSxtrlongpl: added \@glsxtr@record 246
\Glsxtrlongpl: added \@glsxtr@record 245
\glsxtrlongpl: added \@glsxtr@record 245
\GLSxtrshort: added \@glsxtr@record 240
\Glsxtrshort: added \@glsxtr@record 240
\glsxtrshort: added \@glsxtr@record 239
\GLSxtrshortpl: added \@glsxtr@record 244
\Glsxtrshortpl: added \@glsxtr@record 244
\glsxtrshortpl: added \@glsxtr@record 243
\GlsXtrStartUnsetBuffering: new . 119
\GlsXtrStopUnsetBuffering: new . . 119
indexcounter: added check for wrglossary counter 26
\s@GlsXtrStopUnsetBuffering: new 120
1.31 (2018-05-09)
\@GlsXtrStartUnsetBuffering: new 119
\@gls@ifaccessattribute@set: new 192
\@gls@initaccesskeys: new . . . 191, 200
\@gls@setup@default@short@access: new 193
\@glsxtr@record@noglossarywarning: new 158
\@glsxtrbuffer@nodup@unset: new . 119
General: added prefix key for glslink . 81
added prefix key for printgloss . . 145
changed \let to \def 145
\glsaddeach: new 86
\glscapturedgroup: new 439
\glsdefpostdesc: new 224
\glsdefpostlink: new 225
\glsdefpostname: new 213

\glsdohypertarget: bug fix: ensure that new version is picked up 146
\glslistdesc: new 489
\glslocalreseteach: new 121
\glslocalunseteach: new 121
\glstreechilddesc: new 504
\glstreechildsymbol: new 504
\glstreedefaultnamefmt: new 500
\glstreedesc: new 503
\glstreegroupheaderfmt: added redefinition 501
\glstreenamefmt: added redefinition . 501
\glstreenavigationfmt: added redefinition 501
\glstreenonamechilddesc: new 506
\glstreenonamedesc: new 505
\glstreenonamesymbol: new 506
\glstreesymbol: new 504
\glsxtr@newabbreviation: added \ExtraCustomAbbreviationFields 231
\GlsXtrForUnsetBufferedList: new 120
\GlsXtrIfFieldCmpNum: new 42
\GlsXtrIfFieldEqNum: new 42
\GlsXtrIfFieldEqXpStr: new 46
\GlsXtrIfFieldNonZero: new 42
\GlsXtrIfHasNonZeroChildCount: new 439
\GlsXtrIfXpFieldEqXpStr: new 47
\glsxtrpostlinkAddSymbolDescOnFirstUse: new 226
\GlsXtrRecordWarning: new 157
\glsxtrRevertTocMarks: new 365
\GlsXtrStandaloneGlossaryType: new 163
\GlsXtrStandaloneSubEntryItem: new 164
\s@GlsXtrStartUnsetBuffering: new 119
1.32 (2018-05-24)
\GlsXtrForeignText: new 47
\GlsXtrForeignTextField: new 49
\GlsXtrUnknownDialectWarning: new 49
1.33 (2018-07-26)
\ifglsused: added redefinition 51
1.34 (2018-07-29)
\gls@begindocdefs: atom 67
\GlsXtrIfUnusedOrUndefined: new . . 33
\glsxtrNoGlossaryWarning: added package warning 24
\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 79
1.36 (2018-08-18)
\glsxtrautoindexesc: new 217
\glsxtrdisplaysupploc: new 442
\glsxtrmultisupplocation: new . . . 441
1.37 (2018-11-30)
\@@glsxtr@record: added check for auto-add 8
\@dGLS: new 450
\@dGLSpl: new 450
\@dGls: new 450
\@dGlspl: new 450
\@dgls: new 449
\@dglspl: new 449
\@gls@getcounterprefix: new 32
\@glslongextrawidestname: new . . . 536
\@glsxtr@bibgls@removespaces: new 444
\@glsxtr@check@bibgls@nameref: new 159
\@glsxtr@do@nameref@record: new . . 11
\@glsxtr@get@prefixedlabel: new . 449
\@glsxtr@if@record@only: new 14
\@glsxtr@ifnum@mmode: new 11
\@glsxtr@labelprefixes: new 448
\@glsxtr@prefixlabellist: new . . . 448
\@glsxtr@providenewgls: new 174
\@glsxtr@record@only@setup: new . . 14
\@glsxtr@record@setting@nameref: new 14
\@glsxtr@use@equation@counter@or: new 81
General: new 534
page: nameref 10
\dGLS: new 450
\dgls: new 449
\dglsdisp: new 450
\dglslink: new 450
\dGLSpl: new 450
\glsadd: added grouping 85
ensure that \glsadd performs indexing 86
\glslongextraDescAlign: new 536
\glslongextraDescFmt: new 534
\glslongextraDescNameHeader: new 541

\glslongextraDescNameTabularFooter: new 541
\glslongextraDescNameTabularHeader: new 541
\glslongextraDescSymNameHeader: new 553
\glslongextraDescSymNameTabularFooter: new 553
\glslongextraDescSymNameTabularHeader: new 553
\glslongextraGroupHeading: new . . 536
\glslongextraHeaderFormat: new . . 536
\glslongextraLocationAlign: new . 536
\glslongextraLocationDescNameHeader: new 542
\glslongextraLocationDescNameTabularFooter: new 542
\glslongextraLocationDescNameTabularHeader: new 542
\glslongextraLocationDescSymNameHeader: new 554
\glslongextraLocationDescSymNameTabularFooter: new 555
\glslongextraLocationDescSymNameTabularHeader: new 555
\glslongextraLocationFmt: new . . . 535
\glslongextraLocationSymDescNameHeader: new 551
\glslongextraLocationSymDescNameTabularFooter: new 552
\glslongextraLocationSymDescNameTabularHeader: new 551
\glslongextraLocSetDescWidth: new 538
\glslongextraNameAlign: new 536
\glslongextraNameDescHeader: new 536
\glslongextraNameDescLocationHeader: new 539
\glslongextraNameDescLocationTabularFooter: new 540
\glslongextraNameDescLocationTabularHeader: new 539
\glslongextraNameDescSymHeader: new 544
\glslongextraNameDescSymLocationHeader: new 545
\glslongextraNameDescSymLocationTabularFooter: new 545
\glslongextraNameDescSymLocationTabularHeader: new 545
\glslongextraNameDescSymTabularFooter: new 544
\glslongextraNameDescSymTabularHeader: new 544
\glslongextraNameDescTabularFooter: new 536
\glslongextraNameDescTabularHeader: new 536
\glslongextraNameFmt: new 534
\glslongextraNameSymDescHeader: new 547
\glslongextraNameSymDescLocationHeader: new 548
\glslongextraNameSymDescLocationTabularFooter: new 549
\glslongextraNameSymDescLocationTabularHeader: new 548
\glslongextraNameSymDescTabularFooter: new 547
\glslongextraNameSymDescTabularHeader: new 547
\glslongextraSetDescWidth: new . . 537
\glslongextraSetWidest: new 536
\glslongextraSubDescFmt: new 535
\glslongextraSubLocationFmt: new 535
\glslongextraSubNameFmt: new 535
\glslongextraSubSymbolFmt: new . . 535
\glslongextraSymbolAlign: new . . . 536
\glslongextraSymbolFmt: new 534
\glslongextraSymDescNameHeader: new 550
\glslongextraSymDescNameTabularFooter: new 550
\glslongextraSymDescNameTabularHeader: new 550
\glslongextraSymLocSetDescWidth: new 538
\glslongextraSymSetDescWidth: new 537
\glslongextraTabularVAlign: new . 538
\glslongextraUpdateWidest: new . . 537
\glslongextraUpdateWidestChild: new 537
\glsrenewcommand: new 439
\glsseeitemformat: removed reference to \glslabel 58
\glsxtr@dblfloat: new 18
\glsxtr@do@autoadd: new 81
\glsxtr@float: new 18
\glsxtr@record@nameref: new 162

\glsxtr@renewcommand: new 439
\glsxtr@writefields: provide \glsxtr@record@nameref in aux file 161
\glsxtraddlabelprefix: new 448
\GlsXtrAutoAddOnFormat: new 81
\glsxtrclearlabelprefixes: new .. 448
\glsxtrdisplaylocnameref: new ... 442
\glsxtrfmtexternalnameref: new .. 444
\glsxtrfmtinternalnameref: new .. 444
\GLSXTRhiername: new 60
\GLSxtrhiername: new 59
\GlsXtrhiername: new 59
\Glsxtrhiername: new 59
\glsxtrhiername: new 58
\glsxtrhiernamesep: new 60
\glsxtridentifyglslike: new 174
\glsxtrifinlabelprefixlist: new . 448
\GlsXtrLocationField: new 172
\glsxtrnameloclink: new 443
\glsxtrnamereflink: new 443
\glsxtrprependlabelprefix: new .. 448
\GlsXtrSetAltModifier: write modifier to aux 106
\glsxtrSetWidest: new 445
\glsxtrSetWidestFallback: new ... 447
\GlsXtrStandaloneEntryName: new . 163
\GlsXtrStandaloneEntryOther: new 165
\GLSxtrusefield: new 44
\Glsxtrusefield: fixed internal command and added check for \texorpdfstring 43
\ifGlsLongExtraUseTabular: new .. 538
floats: new 18
long-desc-name: new 541
long-desc-sym-name: new 553
long-loc-desc-name: new 543
long-loc-desc-sym-name: new 555
long-loc-sym-desc-name: new 552
long-name-desc: new 538
long-name-desc-loc: new 540
long-name-desc-sym: new 544
long-name-desc-sym-loc: new 546
long-name-sym-desc: new 547
long-name-sym-desc-loc: new 549
long-sym-desc-name: new 550
equations: new 18
1.38 (2018-12-01)
\glslongextraNameFmt: bug fix: removed double param 534
all: added glossary-longextra 487
1.39 (2019-03-22)
\@GlsXtrIfFieldCmpNum: new 43
\@GlsXtrIfFieldEqNum: new 42
\@GlsXtrIfFieldEqStr: new 46
\@GlsXtrIfFieldEqXpStr: new 46
\@GlsXtrIfFieldNonZero: new 42
\@GlsXtrIfXpFieldEqXpStr: new 47
\@gls@removespaces: changed \x to \@glo@tmp 153
\@glsxtr@dorecord: added protection for fragile commands 10
General: added label key for printgloss 145
\glsxtrbookindexlocation: new ... 527
\glsxtrbookindexsublocation: new 528
\glsxtrentryparentname: new 44
\GlsXtrIfFieldCmpNum: added starred version 42
\GlsXtrIfFieldEqNum: added starred version 42
\GlsXtrIfFieldEqStr: added starred form 46
\GlsXtrIfFieldEqXpStr: added starred form 46
\GlsXtrIfFieldNonZero: added starred version 42
\GlsXtrIfXpFieldEqXpStr: added starred form 47
\glsxtrsetglossarylabel: new 146
\glsxtrshortdescname: corrected to show long form as advertised in the manual 266
short-desc: corrected to omit description key as advertised in the manual 266
short-em-desc: bug fix: omit description key as advertised in the manual ... 315
short-sc-desc: bug fix: omit description key as advertised in the manual ... 278
short-sm-desc: corrected to omit description key as advertised in the manual 295
\s@GlsXtrIfFieldCmpNum: new 43
\s@GlsXtrIfFieldEqNum: new 42
\s@GlsXtrIfFieldEqStr: new 46

\s@GlsXtrIfFieldEqXpStr: new 46
\s@GlsXtrIfXpFieldEqXpStr: new ... 47
1.4.2 (??)
\@glossentrysymbol: new 221
\glsentrypdfsymbol: new 221
1.40 (2019-03-22)
General: new 557
\glstopicAssignSubIndent: new ... 559
\glstopicAssignWidest: new 560
\glstopicCols: new 561
\glstopicColsEnv: new 561
\glstopicDesc: new 559
\glstopicGroupHeading: new 558
\glstopicInit: new 559
\glstopicItem: new 558
\glstopicLoc: new 559
\glstopicMarker: new 558
\glstopicMidSkip: new 560
\glstopicName: new 558
\glstopicParIndent: new 559
\glstopicPostSkip: new 560
\glstopicPreSkip: new 560
\glstopicSubIndent: new 559
\glstopicSubItem: new 560
\glstopicSubItemBox: new 561
\glstopicSubItemSep: new 561
\glstopicSubLoc: new 561
\glstopicSubNameFont: new 561
\glstopicTitleFont: new 559
\glstopicwidest: new 559
all: added glossary-topic 487
topic: new 557
topicmcols: new 561
1.40 (2019-03-31)
\glsfirstabbrvdefaultfont: changed definition from \glsabbrvfont to \glsabbrvdefaultfont for consistency 235
\GlsXtrDefaultResourceOptions: new 158
long-hyphen-noshort-noreg: corrected formatting commands ... 349
\printunsrtabbreviations: new ... 438
\printunsrtacronyms: new 438
\printunsrtindex: new 438
\printunsrtnumbers: new 438
\printunsrtsymbols: new 438
1.41 (2019-04-09)
General: changed \thisgrptitle to \glsxtrcurrentgrptitle 532
\glslistgroupskip: new 489
\glstopicAssignSubIndent: moved \par from \glstopicSubItem 559
\glstopicSubItem: added check for description 560
moved \par to \glstopicAssignSubIndent 560
\glstopicSubLoc: moved \space to \glstopicSubPreLocSep 561
\glstopicSubPreLocSep: new 561
\glstreeChildDescLoc: new 504
\glstreeDescLoc: new 503
\glstreegroupskip: new 501
\glstreePreHeader: new 501
\glsxtralttreeSymbolDescLocation: added check for description 508
topic: added penalty if no description . 557
topicmcols: added penalty if no description 562
1.42 (2020-02-03)
\@@glsxtr@record: moved label definition outside of conditional 8
\@ACRfull: added redefinition 99
\@ACRfullpl: added redefinition 99
\@Acrfull: added redefinition 99
\@Acrfullpl: added redefinition 99
\@GlsXtrIfFieldValueInCsvList: new 39
\@acrfull: added redefinition 98
\@acrfullpl: added redefinition 99
\@domakeglossaries: provided definition for \@domakeglossaries 138
\@gls@assign@actual: new 192
\@gls@entry@field: redefined 51
\@gls@setup@default@access: added \glsdefaultshortaccess 193
\@gls@setup@default@short@access: renamed to \@gls@setup@default@access .. 193
\@glslink: switched from \glsdohyperlink to \glsxtrdohyperlink 109
\@glsxtr@abbrlists: new 135
\@glsxtr@acronymlists: new 135
\@glsxtr@addabbreviationlist: new 136
\@glsxtr@base@acrcmd: new 94

\@glsxtr@doloadprefix: new 23
\@glsxtr@org@addtoacronynlists: new 135
\@glsxtr@org@setacronymlists: new 135
\@glsxtrentryfmt: added \glslabel and scope 37
General: added \@afterheading 521
debug: showaccsupp 27
\forallabbreviationlists: new ... 135
\forallacronyms: new 136
\glsdefaultshortaccess: new 192
\glsdisplaynumberlist: added 438
\glsenablehyper: switched from \glsdohyperlink to \glsxtrdohyperlink 109
\glsentrynumberlist: added 439
\GLSfmtfirst: new 383
\GLSfmtfirstpl: new 383
\GLSfmtfull: new 386
\Glsfmtfull: switched pdf case to use \glspdffmtfull 386
\glsfmtfull: switched pdf case to use \glspdffmtfull 386
\GLSfmtfullpl: new 387
\Glsfmtfullpl: switched pdf case to use \glspdffmtfullpl 387
\glsfmtfullpl: switched pdf case to use \glspdffmtfullpl 386
\GLSfmtlong: new 384
\GLSfmtlongpl: new 385
\GLSfmtname: new 380
\GLSfmtplural: new 382
\GLSfmttext: new 381
\glspdffmtfull: new 385
\glspdffmtfullpl: new 386
\glsseeitemformat: switched to using \glsfmttext and \glsfmtname ... 58
\glsshowtarget: added check for \glsshowtargetouter 29
\glstreeChildDescLoc: added \glstreeNoDescSymbolPreLocation 504
\glstreegroupheaderskip: new 501
\glstreeNoDescSymbolPreLocation: new 503
\glsxtr@newabbreviation: moved apply abbreviation style to after category key has been obtained ... 231
removed \relax and updated \@gls@short instead of \glsshorttok 232
replaced explicit \spacefactor with \@ 232
\glsxtr@writefields: added check for order=letter 162
\glsxtrAccSuppAbbrSetFirstLongAttrs: new 196, 200
\glsxtrAccSuppAbbrSetNameLongAttrs: new 197, 200
\glsxtrAccSuppAbbrSetNameShortAttrs: new 196, 200
\glsxtrAccSuppAbbrSetNoLongAttrs: new 196, 200
\glsxtrAccSuppAbbrSetTextShortAttrs: new 196, 200
\glsxtralttreeSymbolDescLocation: switched to using \glstreeDescLoc 508
\glsxtrassignactualsetup: new ... 192
\glsxtrbookindexbookmarkprefix: new 529
\GlsXtrDiscardUnsetBuffering: new 120
\glsxtrdohyperlink: new (was former redefinition of \glsdohyperlink) . 108
\glsxtrequationlocfmt: new 443
\glsxtrfieldformatcsvlist: new ... 39
\glsxtrfieldformatlist: new 38
\glsxtrfootnotedescname: new 260
\glsxtrfootnotedescsort: new 260
\GLSXTRhiername: switched to using \GLSfmttext and \GLSfmtname ... 60
\GLSxtrhiername: switched to using \glsfmttext, \glsfmtname, \GLSfmttext and \GLSfmtname ... 59
\GlsXtrhiername: switched to using \Glsfmttext and \Glsfmtname ... 59
\Glsxtrhiername: switched to using \glsfmttext and \glsfmtname ... 59
\glsxtrhiername: switched to using \glsfmttext and \glsfmtname ... 58
\GlsXtrIfFieldValueInCsvList: new 39
\glsxtrpdfentryfmt: new 37
\glsxtrprovideaccsuppcmd: new ... 196
\glsxtrscsuffix: added \protect .. 273
\GlsXtrSetAltModifier: added check 106
\GLSxtrtitlefirst: new 373
\GLSxtrtitlefirstplural: new 374
\GLSxtrtitlefull: new 378

\GLSxtrtitlefullpl: new 378
\GLSxtrtitlelong: new 376
\GLSxtrtitlelongpl: new 376
\GLSxtrtitlename: new 371
\GLSxtrtitleplural: new 372
\GLSxtrtitleshort: new 369
\GLSxtrtitleshortpl: new 370
\GLSxtrtitletext: new 371
\glsxtrusealias: new 61
short-em: removed \protect from \glsxtremsuffix 314
short-em-desc: removed \protect from \glsxtremsuffix 316
short-em-footnote: added missing text key 325
removed \protect from \glsxtremsuffix 326
short-em-footnote-desc: new 327
short-em-long: added missing text key 310
removed \protect from \glsxtremsuffix 311
short-em-long-em: added missing text key 312
removed \protect from \glsxtremsuffix 313
short-em-postfootnote: added missing text key 328
removed \protect from \glsxtremsuffix 329
short-em-postfootnote-desc: new . 330
short-footnote-desc: new 260
short-hyphen-long-hyphen: added missing text key 353
short-hyphen-postlong-hyphen: added missing text key 356
short-long: added missing text key ... 256
short-long-user: added missing text key 342
short-postfootnote-desc: added missing text key 263
new 263
short-postlong-user: added missing text key 338
short-sc: moved \protect inside \glsxtrscsuffix 277
short-sc-desc: moved \protect inside \glsxtrscsuffix 279
short-sc-footnote: added missing text key 284
moved \protect inside \glsxtrscsuffix 284
short-sc-footnote-desc: new 285
short-sc-long: added missing text key 275
moved \protect inside \glsxtrscsuffix 275
short-sc-postfootnote: added missing text key 286
moved \protect inside \glsxtrscsuffix 287
short-sc-postfootnote-desc: new . 288
short-sm: removed \protect from \glsxtrsmsuffix 294
short-sm-desc: removed \protect from \glsxtrsmsuffix 295
short-sm-footnote: added missing text key 300
removed \protect from \glsxtrsmsuffix 301
short-sm-footnote-desc: new 302
short-sm-long: added missing text key 291
removed \protect from \glsxtrsmsuffix 292
short-sm-postfootnote: added missing text key 303
removed \protect from \glsxtrsmsuffix 304
short-sm-postfootnote-desc: new . 305
\makeglossaries: added \@domakeglossaries 139
let \@makeglossary to \@gobble instead of \relax 140
removed redefinition of \makeglossary 140
\makenoidxglossaries: added \@domakeglossaries 68
long-em-noshort-em: removed \protect from \glsxtremsuffix . 320
long-em-noshort-em-desc: removed \protect from \glsxtremsuffix . 323
long-em-short-em: added missing text key 308
removed \protect from \glsxtremsuffix 309
long-hyphen-noshort-desc-noreg: added missing text key 347
long-hyphen-postshort-hyphen: added missing text key 350

long-hyphen-short-hyphen: added missing text key 345
long-noshort-em: removed \protect from \glsxtremsuffix 318
long-noshort-em-desc: removed \protect from \glsxtremsuffix . 322
long-noshort-sc: moved \protect inside \glsxtrscsuffix 281
long-noshort-sc-desc: moved \protect inside \glsxtrscsuffix 282
long-noshort-sm: removed \protect from \glsxtrsmsuffix 297
long-noshort-sm-desc: removed \protect from \glsxtrsmsuffix . 299
long-only-short-only: added missing text key 359
removed \protect from \glsxtronlysuffix 360
long-postshort-user: added missing text key 333
long-short: added missing text key . . . 253
long-short-em: added missing text key 306
removed \protect from \glsxtremsuffix 307
long-short-sc: added missing text key 273
moved \protect inside \glsxtrscsuffix 273
long-short-sm: added missing text key 290
removed \protect from \glsxtrsmsuffix 290
long-short-user: added missing text key 332
footnote: added missing text key 258
footnote-desc: new 261
postfootnote: added missing text key . 261
prefix: new 23
\RestoreAcronyms: added display style 137
\s@GlsXtrIfFieldValueInCsvList: new 39
\seealsoname: add check for \alsoname 62
1.42 (?)
postfootnote-desc: new 264
1.43 (2020-02-28)
\@glsxtrentryfmt: changed \def to \edef to avoid infinite recursion . . . 37
1.44 (2020-03-23)
\@glsxtr@assign@leveloffset: new 146
\@glsxtr@leveloffset: new 146
\@glsxtr@noidx@do: replaced \ifglshasparent with \@glsxtr@ifischild 172
\@print@unsrt@innerglossary: new 168
General: added groups key 146
added leveloffset key 146
\doifglossarynoexistsordo: switched to starred form of \ifglossaryexists 57
\glswriteentry: replaced \ifglsused with \GlsXtrIfUnusedOrUndefined . . 104
\glsxtr@printgloss@checkexists: new 143
\glsxtralttreeSymbolDescLocation: removed duplicate description 508
\ifglossaryexists: added check for starred form 33
\np@glsxtr@assign@leveloffset: new 146
\p@glsxtr@assign@leveloffset: new 146
\pp@glsxtr@assign@leveloffset: new 146
\printunsrtglossary: added check for \@printgloss@checkexists 165
\printunsrtinnerglossary: new . . . 167
printunsrtglossarywrap: new 167
1.45 (2020-04-01)
General: removed duplicate description 506
\glstreenonameChildDescLoc: new . 506
\glstreenonameDescLoc: new 506
1.46 (2021-09-18)
\@glsxtrsetaliasnoindex: changed to use starred version of \glsxtrifhasfield 103
1.46 (2021-09-20)
\@@glsxtr@record: changed \edef to \protected@edef 8
\@@newglossaryentry@defunitcounters: changed \edef to \protected@edef 127
\@glossentrysymbol: changed \edef to \protected@edef 221
\@gls@increment@currunitcount: changed \edef to \protected@edef 128
\@gls@link: changed \edef to \protected@edef 82, 83
\@gls@link@checkfirsthyper: changed \edef to \protected@edef 101

\@gls@local@increment@currunitcount: changed \edef to \protected@edef 129
\@gls@setup@default@access: changed \edef to \protected@edef 193, 194
\@glsxtr@addabbreviationlist: changed \eappto to \protected@eappto 136
changed \edef to \protected@edef 136
\@glsxtr@bibgls@removespaces: changed \x to \@glo@tmp 444
\@glsxtr@do@inc@linkcount: changed \x to \@glo@tmp 182
\@glsxtr@do@record@wrglossary: changed \edef to \protected@edef . 7
\@glsxtr@do@redef@forglsentries: changed \edef to \protected@edef . 5
\@glsxtr@get@prefixedlabel: changed \edef to \protected@edef 449
changed \x to \@glo@tmp 449
\@glsxtr@mixed@assign@sortkey: changed \edef to \protected@edef 147
\@glsxtr@op@recordcounter: changed \eappto to \protected@eappto . . . 12
\@glsxtr@orgprintglossary: changed \xdef to \protected@xdef 143
\@glsxtr@rglstrigger@record: changed \edef to \protected@edef 177
\@glsxtr@warn@hybrid@noprintgloss: new 14
\@glsxtrentryfmt: changed \edef to \protected@edef 37
\@glsxtrglossentry: changed \edef to \protected@edef 163
\@glsxtrglossentryother: changed \edef to \protected@edef 164
\@glsxtrindexaliased: changed \edef to \protected@edef 103
\@makeglossaries@warn@noprintglossary: new 138
\@newglossaryentryposthook: changed \edef to \protected@edef 65
\@print@unsrt@glossary: changed \eappto to \protected@eappto . . 167
\@print@unsrt@innerglossary: changed \eappto to \protected@eappto 169
\@printunsrt@glossary@handler: changed \xdef to \protected@xdef 170
General: changed \edef to \protected@edef 64, 229
record: added hybrid 15
\glossentrydesc: changed \edef to \protected@edef 208, 209
\Glossentryname: changed \edef to \protected@edef 212, 213
\glossentryname: changed \edef to \protected@edef 209, 211
\glossentrynameother: changed \edef to \protected@edef 214
\glsadd: changed \edef to \protected@edef 85
\glsalttreechildpredesc: new 507
\glsalttreepredesc: new 507
\glsdisablehyper: changed \edef to \protected@edef 109
\glsdoifexists: changed \edef to \protected@edef 56
\glsenableentryunitcount: changed \edef to \protected@edef 131
\glsFindWidestLevelTwo: changed \edef to \protected@edef 513
\glsFindWidestUsedLevelTwo: changed \edef to \protected@edef 512
\glsnavhyperlink: changed \edef to \protected@edef 106
\glstopicAssignSubIndent: bug 182 maintain hangindent for multiple paragraphs 559
\glstopicsubitemhangindent: new . 559
\glstopicSubItemParIndent: new . . 559
\glsxtr@org@newignoredglossary: changed \eappto to \protected@eappto 53
changed \edef to \protected@edef . 53
\glsxtr@provideignoredglossary: changed \eappto to \protected@eappto 55
changed \edef to \protected@edef . 54
\glsxtr@s@newignoredglossary: changed \edef to \protected@edef 53
\glsxtr@s@provideignoredglossary: changed \edef to \protected@edef 55
\glsxtr@setaccessdisplay: changed \edef to \protected@edef 214
\glsxtralttreeSymbolDescLocation: switch to using \glsalttreepredesc and \glsalttreechildpredesc . . 508

\glsxtrdisplayendloc: changed \edef to \protected@edef 152
\glsxtrdisplaystartloc: changed \edef to \protected@edef 152
\glsxtrdoautoindexname: changed \eappto to \protected@eappto . . 216
\glsxtrseelist: changed \edef to \protected@edef 61
\glsxtrtreechildpredesc: new 503
\glsxtrtreepredesc: new 503
\makeglossaries: adjust warning on missing glossary for “alsoindex” . . . 139
changed \edef to \protected@edef 139–141
\makenoidxglossaries: changed \edef to \protected@edef 68
topic: added \par (bug 176) 558
grouping added to scope \everypar (bug 182) 558
printunsrtglossarywrap: changed \xdef to \protected@xdef 168
\setabbreviationstyle: changed \edef to \protected@edef 250
1.47 ()
\@GlsXtrIfValueInFieldCsvList: new 40
\@xGlsXtrIfValueInFieldCsvList: new 41
\s@GlsXtrIfValueInFieldCsvList: new 40
\s@xGlsXtrIfValueInFieldCsvList: new 41
1.47 (2017-11-14)
\@glsxtrforcsvfield: new 38
\s@glsxtrforcsvfield: new 39
1.47 (2021-11-04)
\@GlsXtrIfHasNonZeroChildCount: new 439
\@glsxtrsetaliasnoindex: changed to use \ifcsvoid 103
\glsaltlistitem: new 490
\glslistexpandedname: new 490
\glslistgroupafterheader: new . . . 491
\glslistinit: new 489
\glslistitem: new 489
\glsseefirstitem: new 62
\glsseelastoxfordsep: new 62
\glsseelist: redefined 61
\glsunsetcategoryattribute: new . 202
\glsxtrapptocsvfield: new 44
\glsxtrcopytoglossary: replaced \cseappto with \protected@cseappto 55
\glsxtrfieldtitlecasecs: added check for \glscapitalisewords . . 207
\GlsXtrIfHasNonZeroChildCount: added starred version 439
\GlsXtrIfValueInFieldCsvList: new 40
\s@GlsXtrIfFieldNonZero: new 42
\s@GlsXtrIfHasNonZeroChildCount: new 439
\xGlsXtrIfValueInFieldCsvList: new 40
1.48 (2017-02-03)
long-postshort-sc-user: new 335
1.48 (2021-11-22)
\@@gls@navhypertarget: new 107
\@@mgls@hyperlink: new 402
\@GlsXtrMglsOrGls: new 427
\@Glsfieldorgls: new 433
\@Glsfullorfirst@: new 430
\@Glslongortext@: new 430
\@Glsshortortext@: new 430
\@PGLSorgls: new 434
\@PGLSorglspl: new 434
\@Pglsorgls: new 434
\@Pglsorglspl: new 434
\@alt@GlsXtrMglsOrGls: new 427
\@def@multi@glossaryentry: new . . 393
\@defmultiglossaryentry: new 392
\@firstofthree: new 408
\@gls@combined@category: new 389
\@gls@combined@encapmain: new . . . 389
\@gls@combined@encapothers: new . 389
\@gls@combined@firstprefix: new . 390
\@gls@combined@firstskipmain: new 390
\@gls@combined@firstskipothers: new 390
\@gls@combined@firstsuffix: new . 390
\@gls@combined@hyper: new 388
\@gls@combined@indexmain: new . . . 388
\@gls@combined@indexothers: new . 388
\@gls@combined@mglsopts: new 389
\@gls@combined@mglsopts@do: new 389, 390
\@gls@combined@mpostlink: new . . . 391
\@gls@combined@mpostlinkelement: new 391

\@gls@combined@postlinks: new ... 391
\@gls@combined@textformat: new .. 389
\@gls@combined@usedprefix: new .. 390
\@gls@combined@usedskipmain: new 391
\@gls@combined@usedskipothers: new ... 391
\@gls@combined@usedsuffix: new .. 390
\@gls@do@glsunset: new ... 102
\@gls@restore@glslocal: new ... 102
\@gls@save@glslocal: new ... 102
\@glsfieldorgls: new ... 433
\@glsfullorfirst@: new ... 430
\@glslongortext@: new ... 430
\@glsnavhypertarget: added patch .. 107
\@glsshortortext@: new ... 430
\@glsshowtarget: new ... 29
\@glsshowtargetmarkfmt: new ... 30
\@glssymbolorGls: new ... 432
\@glssymbolorgls: new ... 432
\@glsxtr@addunused: added check for multientry labels ... 66
\@glsxtr@do@org@target: new ... 147
\@glsxtr@doshowtarget: new ... 26
\@glsxtr@mglslike: new ... 426
\@glsxtr@mglsrefs: new ... 425
\@glsxtr@mglswrite: new ... 424
\@glsxtr@multientry: new ... 396
\@glsxtr@seefirstitem: new ... 62
\@glsxtr@seeitem: new ... 61
\@glsxtrmultientryadjustedname: new ... 452
\@glsxtrshowtargetleft: new ... 27
\@glsxtrshowtargetmark: new ... 27
\@glsxtrshowtargetright: new ... 27
\@mgls@all: new ... 400
\@mgls@disable@writeseparateref@cond: new ... 425
\@mgls@hyper: new ... 401
\@mgls@hyperlink: new ... 402
\@mgls@main: new ... 401
\@mgls@others: new ... 401
\@mgls@resetall: new ... 403
\@mgls@resetmain: new ... 403
\@mgls@resetothers: new ... 403
\@mgls@setup: new ... 401
\@mgls@setup@do: new ... 401
\@mgls@setup@do@not: new ... 401
\@mgls@unsetaction: new ... 401
\@mgls@unsetall: new ... 404
\@mgls@unsetmain: new ... 404
\@mgls@unsetothers: new ... 404
\@mglslocalreset: new ... 398
\@mglslocalunset: new ... 398
\@mglsreset: new ... 398
\@mglsunset: new ... 397
\@multi@glossary@doifexists: new 393
\@multi@glossary@entry: new ... 394
\@multi@glossaryentry: new ... 392
\@multi@glossaryentry@list: new . 394
\@multiglossaryentry: new ... 392
\@pglsorgls: new ... 434
\@pglsorglspl: new ... 434
\@provide@multi@glossaryentry@noop: new ... 394
\@secondofthree: new ... 408
\@thirdofthree: new ... 408
\alt@GlsXtrMglsOrGls: new ... 427
\glsabbrvsconlyfont: new ... 361
\glsabbrvscuserfont: new ... 334
\glscombinedfirstsep: new ... 421
\glscombinedfirstsepfirst: new .. 421
\glscombinedsep: new ... 421
\glscombinedsepfirst: new ... 422
\glsdoshowtarget: new ... 29
\glsfirstabbrvsconlyfont: new ... 361
\glsfirstabbrvscuserfont: new ... 335
\glslinkwrcontent: new ... 82
\glsnavhypertarget: new ... 107
\glssetcategoriesattribute: new . 201
\glssetcategoriesattributes: new 201
\glssetcombinedsepabbrvnbsp: new 422
\glssetcombinedsepabbrvnone: new 422
\glssetcombinedsepnarrow: new ... 423
\glsshowtarget: removed check for \glsshowtargetouter ... 29
\glsshowtargetfont: new ... 29
\glsshowtargetinner: new ... 29
\glsshowtargetinnercontentsymleft: new ... 29
\glsshowtargetinnercontentsymright: new ... 29
\glsshowtargetouter: new ... 29
\glsxtr@mgls@applyopts: new ... 407
\glsxtr@mgls@checklastelement: new ... 406
\glsxtr@mgls@inner: new ... 408
\glsxtr@newmgls: new ... 426
\glsxtr@setup@docurrent: new ... 405

\glsxtrdohyperlink: added check for multi-entry 108
\glsxtrifmulti: new 391
\glsxtrlongshortscuserdescname: new 337
\GlsXtrMglsOrGls: new 426
\glsxtrmglsWarnAllSkipped: new . . 406
\GLSxtrmultientryadjustedname: new 452
\GlsXtrmultientryadjustedname: new 451
\Glsxtrmultientryadjustedname: new 451
\glsxtrmultientryadjustedname: new 451
\GLSxtrmultientryadjustednamefmt: new 453
\GlsXtrmultientryadjustednamefmt: new 453
\Glsxtrmultientryadjustednamefmt: new 453
\glsxtrmultientryadjustednamefmt: new 453
\GLSxtrmultientryadjustednameother: new 453
\GlsXtrmultientryadjustednameother: new 453
\Glsxtrmultientryadjustednameother: new 453
\glsxtrmultientryadjustednameother: new 453
\glsxtrmultientryadjustednamepostsep: new 453
\glsxtrmultientryadjustednamepresep: new 453
\glsxtrmultientryadjustednamesep: new 453
\glsxtrmultilastotherindex: new . 391
\glsxtrmultilist: new 391
\glsxtrmultimain: new 391
\glsxtrmultimainindex: new 391
\glsxtrmultitotalelements: new . . 391
\glsxtrsconlydescname: new 363
\glsxtrsconlydescsort: new 363
\glsxtrsconlyname: new 362
\glsxtrsconlysuffix: new 362
\glsxtrscusername: new 335
\glsxtrscusersuffix: new 335
\glsxtrshowtargetinner: new 27
\glsxtrshowtargetouter: new 27
\glsxtrshowtargetsymbolleft: new . 29
\glsxtrshowtargetsymbolright: new 29
showtargets: new 27
\if@mgls@writeseparaterefs: new . 425
\ifKV@mgls@presetlocal: new 401
\ifmglsused: new 397
\ifmultiglossaryentryglobal: new 392
\MGLS: new 429
\MGls: new 429
\Mgls: new 428
\mgls: new 427
\mgls@disable@mglsopts: new 390
\mgls@disable@setup: new 401
\mgls@enable@mglsopts: new 390
\mgls@enable@setup: new 401
\mglsAddOptions: new 400
\mglscustompostlinkhook: new 419
\mglsdefcategoryprefix: new 420
\mglsdefcategorysuffix: new 420
\mglselementindex: new 392
\mglselementposthook: new 421
\mglselementprehook: new 421
\mglselementreset: new 403
\mglselementunset: new 403
\mglsfield: new 433
\mglsforelements: new 402
\mglsforotherelements: new 402
\Mglsfull: new 431
\mglsfull: new 431
\mglshascategoryprefix: new 420
\mglshascategorysuffix: new 421
\mglslastelementpostlinkhook: new 420
\mglslastmainpostlinkhook: new . . 420
\mglslocalreset: new 398
\mglslocalunset: new 398
\mglslocalunsetothers: new 402
\Mglslong: new 431
\mglslong: new 430
\MGLSmainpl: new 429
\MGlsmainpl: new 429
\Mglsmainpl: new 428
\mglsmainpl: new 428
\MGlsname: new 432
\Mglsname: new 432
\mglsname: new 431
\MGLSpl: new 429
\MGlspl: new 429
\Mglspl: new 428

\mglspl: new 428
\mglsprefix: new 420
\mglsreset: new 397
\mglsresetall: new 399
\mglsseefirstitem: new 62
\mglsseeitem: new 62
\mglsSetMain: new 399
\mglsSetOptions: new 400
\Mglsshort: new 431
\mglsshort: new 430
\mglssuffix: new 421
\MGlssymbol: new 433
\Mglssymbol: new 432
\mglssymbol: new 432
\mglsunset: new 397
\mglsunsetall: new 398
\mglsunsetothers: new 402
\mglsusecategoryprefix: new 420
\mglsusecategorysuffix: new 421
\MGlsusefield: new 433
\Mglsusefield: new 433
\mglsusefield: new 433
\mglsWriteSeparateRefsFalse: new 425
\mglsWriteSeparateRefsTrue: new . 425
\MPGLS: new 436
\MPGls: new 436
\Mpgls: new 435
\mpgls: new 435
\MPGLSmainpl: new 436
\MPGlsmainpl: new 436
\Mpglsmainpl: new 435
\mpglsmainpl: new 435
\MPGLSpl: new 436
\MPGlspl: new 436
\Mpglspl: new 435
\mpglspl: new 435
\mpglsWarning: new 434
\multiglossaryentry: new 392
\multiglossaryentrysetup: new ... 388
long-only-short-sc-only: new 362
long-only-short-sc-only-desc: new 363
long-postshort-sc-user-desc: new 338
\p@GlsXtrMglsOrGls: new 427
\providemultiglossaryentry: new . 393
\s@GlsXtrMglsOrGls: new 427
\writemultiglossentry: new 397

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

\$ 439
\, ... 60
\. 18, 19, 225, 500
\@ 67, 159, 192, 232
\@@cGLS@ 123, 131
\@@cGLSpl@ 123, 131
\@@cGls@ 123, 131
\@@cGlspl@ 123, 131
\@@cgls@ 123, 131
\@@cglspl@ 123, 131
\@@do@@wrglossary 7, 9, 140
\@@do@wrglossary 12, 14–16, 31, 86, 103
\@@glo@assign@sortkey 147
\@@glo@list 5
\@@glo@type 165
\@@glossarysec 528
\@@glossaryseclabel 146
\@@gls@expand@field 35
\@@gls@navhypertarget 106, 107
\@@glslocalreset 120
\@@glslocalunset 120
\@@glsreset 120
\@@glsunset 119
\@@glsxtr@autoindex@escspch ... 218–220
\@@glsxtr@base@acrcmd@warn 94, 136
\@@glsxtr@checkspch 217, 218, 220
\@@glsxtr@disabledflycommand 72, 73
\@@glsxtr@org@postdescription 144
\@@glsxtr@record 14, 16
\@@glsxtr@recordcounter 14–16, 162
\@@glsxtrfmt 35, 36
\@@glsxtrp 114
\@@glsxtrpostloctag 75
\@@glsxtrpreloctag 75
\@@glsxtrwrglossmark .....
..... 7, 8, 12, 30, 31, 63, 68, 140
\@@mgls@hyperlink 402
\@@newglossaryentry@defcounters ... 122
\@@newglossaryentry@defunitcounters 130
\@@par 508
\@ACRlong 110
\@ACRlongpl 110
\@ACRshort 110
\@ACRshortpl 110
\@Acrlong 110
\@Acrlongpl 110
\@Acrshort 110
\@Acrshortpl 110
\@GLS@ ... 110, 125, 126, 180, 429, 434, 436, 450
\@GLSdesc@ 91
\@GLSpl@ . 110, 126, 127, 181, 429, 434, 436, 450
\@GLSplural@ 111
\@GLSsymbol@ 92
\@GLStext@ 111
\@GLSxtr@full 237
\@GLSxtr@fullpl 238
\@GLSxtr@p@acrlong@ 110
\@GLSxtr@p@acrlongpl@ 110
\@GLSxtr@p@acrshort@ 110
\@GLSxtr@p@acrshortpl@ 110
\@GLSxtr@p@long@ 110
\@GLSxtr@p@longpl@ 110
\@GLSxtr@p@plural@ 110
\@GLSxtr@p@short@ 110
\@GLSxtr@p@shortpl@ 110
\@GLSxtr@p@text@ 110
\@GLSxtrlong 110, 242
\@GLSxtrlongpl 110, 246
\@GLSxtrp 117, 118
\@GLSxtrshort 110, 240
\@GLSxtrshortpl 110, 244
\@Gls@ 110,
125, 126, 179, 428, 429, 432–434, 436, 450

\@Gls@acrentryname 134
\@Gls@entry@field 44, 99, 116, 117, 215
\@Gls@entryname 134
\@GlsXtrEnableOnTheFly 69, 70
\@GlsXtrIfFieldCmpNum 42
\@GlsXtrIfFieldEqNum 42
\@GlsXtrIfFieldEqStr 46
\@GlsXtrIfFieldEqXpStr 46
\@GlsXtrIfFieldNonZero 42, 439
\@GlsXtrIfFieldValueInCsvList 39
\@GlsXtrIfHasNonZeroChildCount 439
\@GlsXtrIfValueInFieldCsvList 40
\@GlsXtrIfXpFieldEqXpStr 47
\@GlsXtrMglsOrGls 427
\@GlsXtrStartUnsetBuffering 119
\@GlsXtrStopUnsetBuffering 119
\@Glsfieldorgls 433, 434
\@Glsfirst@ 430
\@Glsfullorfirst 430, 431
\@Glslongortext 430, 431
\@Glsname@ 432
\@Glspl@ .....
110, 125, 126, 180, 428, 429, 434, 436, 450
\@Glsplural@ 111
\@Glsshortortext 430, 431
\@Glstext@ 110, 430
\@Glsxtr 71, 72
\@Glsxtr@full 236, 430
\@Glsxtr@fullpl 238
\@Glsxtr@p@acrlong@ 110
\@Glsxtr@p@acrlongpl@ 110
\@Glsxtr@p@acrshort@ 110
\@Glsxtr@p@acrshortpl@ 110
\@Glsxtr@p@long@ 110
\@Glsxtr@p@longpl@ 110
\@Glsxtr@p@plural@ 110
\@Glsxtr@p@short@ 110
\@Glsxtr@p@shortpl@ 110
\@Glsxtr@p@text@ 110
\@Glsxtrlong 110, 242, 430
\@Glsxtrlongpl 110, 245
\@Glsxtrp 116, 117
\@Glsxtrpl 72
\@Glsxtrshort 110, 240, 430
\@Glsxtrshortpl 110, 243
\@PGLS@ 434
\@PGLSorgls@ 436
\@PGLSorglspl@ 436
\@PGLSpl@ 434
\@Pgls@ 434
\@Pglsorgls@ 435, 436
\@Pglsorglspl@ 435, 436
\@Pglspl@ 434
\@acrlong 110
\@acrlongpl 110
\@acrshort 110
\@acrshortpl 110
\@addtoacronymlists 134–137
\@addtoreset 182
\@afterheading 490,
491, 502, 503, 505, 507, 520–524, 532, 558
\@alt@GlsXtrMglsOrGls 427
\@alt@gls@hyp@opt 105
\@auxout 10–12, 68, 76,
106, 107, 123, 124, 132, 139, 140, 154,
159–161, 163, 174, 397, 425, 426, 449, 532
\@bibgls@restoreat 159
\@cGLS 126
\@cGLS@ 123, 126, 131
\@cGLSpl 126
\@cGLSpl@ 123, 127, 131
\@cGls@ 123, 131
\@cGlspl@ 123, 131
\@cgls@ 123, 131
\@cglspl@ 123, 131
\@currentlabelname 146
\@dGLS 450
\@dGLSpl 450
\@dGls 449
\@dGlspl 450
\@dblfloat 18
\@def@multi@glossaryentry 392, 393
\@def@multi@glossaryentry@do 393
\@defmultiglossaryentry 392
\@dgls 449
\@dglspl 449
\@disable@onlypremakeg 140
\@do@auxoutstuff 154
\@do@gls@getcounterprefix 10, 11
\@do@glssee 64, 65
\@do@newglossaryentry 134, 135, 233
\@do@seeglossary 14–16, 30, 68, 140
\@do@wrglossary 84, 178
\@domakeglossaries 68, 139
\@dtl@formatlist@handler 38
\@dtl@formatlist@itemsep 38
\@dtl@formatlist@lastitem 38
\@dtl@formatlist@prelastitem 38

\@dtl@formatlist@prelastitemsep 38
\@dtlformatlist 39
\@empty .. 32, 86, 94–98, 136, 145, 152, 217, 218, 236–246, 395, 407–409, 411, 413, 418
\@end@glsxtr@addunused 66
\@end@glsxtr@gettype 142, 147
\@end@glsxtr@usesee 58
\@end@glsxtrifhyphenstart 344
\@endfortrue 39, 214, 250, 449
\@firstofone 86, 166, 169, 192, 208, 209, 216, 221, 222, 390, 401, 405, 406, 410, 443, 489
\@firstofthree .. 79, 86, 94–97, 105, 236, 238, 239, 241, 243, 245, 408, 411, 418, 419
\@firstoftwo 87–92, 95–98, 101, 105, 137, 172, 214, 227, 228, 236, 238, 239, 243–246, 366, 367, 406, 408, 409, 411, 413, 415, 418, 419, 427
\@float 18
\@for 5, 24, 39, 61, 66, 86, 121, 122, 133, 136, 137, 139, 142, 150, 166, 169, 177, 183, 201, 207, 214, 222, 394, 398, 399, 402–405, 413, 449, 452
\@glo@alias 63, 64
\@glo@assign@sortkey 142
\@glo@autosee 30
\@glo@autoseehook 64
\@glo@category 128
\@glo@check@sortallowed 142
\@glo@counterprefix 10, 11, 32, 152, 153, 444
\@glo@countunit 128
\@glo@default@sorttype 142
\@glo@desc 52
\@glo@descplural 52
\@glo@group 13
\@glo@label 13, 34, 57, 64–66, 99, 109, 510–517
\@glo@location 13
\@glo@loclist 13
\@glo@name 216, 217
\@glo@no@assign@sortkey 147
\@glo@parent 512, 513
\@glo@see 57, 58, 60, 61, 64–66
\@glo@seealso 63, 64
\@glo@sort 217
\@glo@sorttype 142, 151
\@glo@text 79
\@glo@thislettergrp 172
\@glo@thisvalue 331
\@glo@tmp ... 32, 34, 61, 99, 153, 182, 444, 449
\@glo@type 66, 106, 107, 134, 136, 139, 143, 144, 147, 150, 151, 154, 157, 158, 166–169
\@glo@types 14, 138, 205, 510–517
\@glossary@default@style 73, 143, 167, 526
\@glossarystyle 143, 167, 168
\@glossentrysymbol 220, 221
\@gls@ ... 110, 124, 126, 179, 428, 432–435, 449
\@gls@@automake@immediate 139
\@gls@@link 79
\@gls@ReturnAfterFi 153, 444
\@gls@actualchar 217
\@gls@actuallong 193, 194
\@gls@actuallongpl 193, 194
\@gls@actualshort 193, 194
\@gls@actualshortpl 193, 194
\@gls@adjustmode 85
\@gls@alt@hyp@opt 105
\@gls@alt@hyp@opt@char 105, 106, 427
\@gls@alt@hyp@opt@keys 105, 106
\@gls@assign@actual 193
\@gls@automake 142
\@gls@between 150
\@gls@checkedmkidx 217, 218, 220
\@gls@checkmkidxchars 63, 217
\@gls@codepage 154
\@gls@combined@category 409, 410
\@gls@combined@current@label .. 392–397
\@gls@combined@current@main 394, 395, 399
\@gls@combined@current@opts 392, 393, 395–397
\@gls@combined@currentlist . 394–396, 399
\@gls@combined@encapmain 415
\@gls@combined@encapothers 416
\@gls@combined@encapsulator ... 410, 413
\@gls@combined@finalitem 394, 395, 399, 400
\@gls@combined@firstprefix 411
\@gls@combined@firstsuffix 411
\@gls@combined@hyper 410, 415, 416
\@gls@combined@hyper@val 389
\@gls@combined@indexmain 415
\@gls@combined@indexmain@val 388
\@gls@combined@indexothers 416
\@gls@combined@indexothers@val 388
\@gls@combined@mglsopts 409
\@gls@combined@mglsopts@do 389, 390
\@gls@combined@mglsopts@do@not 390
\@gls@combined@mpostlink@nr 391, 411, 412
\@gls@combined@mpostlink@val 391

\@gls@combined@mpostlinkelement@nr 391, 411, 412
\@gls@combined@mpostlinkelement@val 391
\@gls@combined@postlinks@nr ... 391, 413
\@gls@combined@postlinks@val 391
\@gls@combined@textformat 410
\@gls@combined@usedprefix 411
\@gls@combined@usedsuffix 411
\@gls@counter 8, 10, 11, 26, 32, 81, 82, 85, 103, 177
\@gls@currentlettergroup 151, 166, 169, 172
\@gls@declareoption 4
\@gls@default@longpl 231, 232
\@gls@deffile 67
\@gls@do@glsunset 79, 102, 178
\@gls@doautomake 142, 162
\@gls@doautomake@err 162
\@gls@dolast 61
\@gls@donext 61
\@gls@enablesavenonumberlist 67
\@gls@encapchar 218
\@gls@entry@count 123, 124
\@gls@entry@field 35, 43, 44, 64, 99, 115–118, 122, 164, 165
\@gls@entry@unitcount 131, 132
\@gls@field@font 86–94
\@gls@field@link 87–94, 100, 101
\@gls@firstaccess 191, 195
\@gls@firstpluralaccess 191, 195
\@gls@get@counterprefix 32
\@gls@getcounterprefix 10
\@gls@getgrouptitle 149, 150, 166, 169
\@gls@grptitle 106, 150
\@gls@hyp@opt 100, 101, 105, 126, 174, 178–180, 236–246, 426, 449, 450
\@gls@hyp@opt@cs 105
\@gls@hypergroup 107
\@gls@ifaccessattribute@set ... 193, 194
\@gls@ifinlist 170
\@gls@increment@currcount 122
\@gls@increment@currunitcount 131
\@gls@initaccesskeys 231
\@gls@keymap 13, 34, 57, 63, 99, 160, 214
\@gls@label 7, 8, 10, 11, 68, 104, 105, 140, 163, 250
\@gls@levelchar 218
\@gls@link 36, 78, 79, 94–98, 236–246
\@gls@link@checkfirsthyper 79, 137
\@gls@link@label 82, 177
\@gls@link@nocheckfirsthyper 78, 94–98, 236–246
\@gls@link@opts 82
\@gls@list 107, 150
\@gls@local@increment@currcount ... 123
\@gls@local@increment@currunitcount 131
\@gls@location 172, 173
\@gls@loclist 147–149, 172, 173
\@gls@long 231
\@gls@longaccess 191, 193, 194
\@gls@longaccesspl 191, 194
\@gls@longpl 193, 229, 231–233
\@gls@map 214
\@gls@nameaccess 191, 194
\@gls@nohyperlist 53, 55
\@gls@noidx@do 151
\@gls@noidx@getgrouptitle 166, 169
\@gls@noidx@nosanitizesort 141
\@gls@noidx@sanitizesort 141
\@gls@noidxloclist@finalsep ... 147, 148
\@gls@noidxloclist@prev 148
\@gls@noidxloclist@sep 147
\@gls@noref@warn 140, 151
\@gls@org@combined@encapsulator 410, 411
\@gls@org@glsnoidxdisplayloc 148
\@gls@org@glsseeformat 148
\@gls@org@longpl 232
\@gls@org@shortpl 232
\@gls@pluralaccess 191, 195
\@gls@preglossaryhook 144, 168, 222
\@gls@prevlevel 518–520, 524–526
\@gls@quotechar 217
\@gls@reference 68, 139, 140
\@gls@resetlabel 403, 404
\@gls@restore@glslocal 78, 85, 178
\@gls@restoreat 67
\@gls@save@glslocal 78, 82, 178
\@gls@saveentrycounter 14–16, 32, 83, 85, 178
\@gls@see@noindex 31, 159
\@gls@setdefault@glslink@opts 8, 36, 83, 104
\@gls@setsort 83, 85
\@gls@setup@default@access 233
\@gls@setupsort@none 15
\@gls@short 232
\@gls@shortaccess 191, 193–195
\@gls@shortaccesspl 191, 194, 195
\@gls@shortpl 193, 229, 232, 233
\@gls@sort 172

\@gls@textaccess 191, 195
\@gls@thisHloc 32
\@gls@thisattrlabel 201
\@gls@thiscatlabel 201
\@gls@thishypernavlabel 107
\@gls@thislabel 61, 86, 121, 449, 450
\@gls@thisloc 32
\@gls@thisval 214
\@gls@tmp .. 41, 46, 47, 150, 193, 394, 399, 402
\@gls@tmpb 220
\@gls@type .. 136, 137, 140–142, 250, 510–517
\@gls@unsetlabel 404, 405
\@gls@write@entrycounts 123
\@gls@write@entryunitcounts 131
\@gls@write@entryunitcounts@do 132
\@gls@writedef 67, 68
\@gls@xref 12, 63
\@gls@xtradjustedother 452
\@gls@xtrmultientryadjustednamesep 452, 453
\@glsabbrv@current@abbreviation 231, 247
\@glsacronymlists 134–137
\@glsdisp 433
\@glsdoifexistsorwarn 17, 209, 211, 212, 214
\@glsentry 67, 123, 124, 132
\@glsfieldorgls 433
\@glsfirst@ 430
\@glsfullorfirst 430, 431
\@glslink 84, 106, 109, 402
\@glslocalreset 121
\@glslocalunset 120, 121
\@glslongextra@begintab 538–552, 554–556
\@glslongextrawidestname 536, 537
\@glslongortext 430, 431
\@glsname@ 431, 432
\@glsnavhypertarget 107
\@glsnextpages 143, 168
\@glsnonextpages 143, 168
\@glsnumberformat 8, 10, 11, 82, 85, 103, 177, 213, 216, 443, 444
\@glsorder 139
\@glspl@ . 110, 125, 126, 179, 428, 434, 435, 449
\@glsplural@ 111
\@glspunc@token 228
\@glsrecordlocref 10
\@glsseeitem 61
\@glsseelastsep 61
\@glsshortortext 430, 431
\@glsshowtarget 27
\@glsshowtargetmarkfmt 28
\@glsstyle@altlist 490
\@glsstyle@altlistgroup 491
\@glsstyle@altlisthypergroup 492
\@glsstyle@alttree 507
\@glsstyle@alttreegroup 519
\@glsstyle@alttreehypergroup 520
\@glsstyle@index 501
\@glsstyle@indexgroup 502
\@glsstyle@indexhypergroup 502
\@glsstyle@inline 500
\@glsstyle@list 489
\@glsstyle@listdotted 488
\@glsstyle@listgroup 491
\@glsstyle@listhypergroup 491
\@glsstyle@mcolalttree 524
\@glsstyle@mcolalttreegroup 524
\@glsstyle@mcolalttreehypergroup .. 525
\@glsstyle@mcolalttreespannav 525
\@glsstyle@mcolindexgroup 520
\@glsstyle@mcolindexhypergroup 521
\@glsstyle@mcolindexspannav 521
\@glsstyle@mcoltreegroup 521
\@glsstyle@mcoltreehypergroup 522
\@glsstyle@mcoltreenonamegroup 523
\@glsstyle@mcoltreenonamehypergroup 523
\@glsstyle@mcoltreenonamespannav .. 523
\@glsstyle@mcoltreespannav 522
\@glsstyle@tree 503
\@glsstyle@treegroup 505
\@glsstyle@treehypergroup 505
\@glsstyle@treenoname 505
\@glsstyle@treenonamegroup 507
\@glsstyle@treenonamehypergroup ... 507
\@glssymbol@ 432
\@glssymbolorGls 432, 433
\@glssymbolorgls 432
\@glstarget 109, 145–147
\@glstext@ 110, 430
\@glsunset 119, 120
\@glswidestname 510, 518
\@glsxr@newmgls@do 426
\@glsxtr 70, 72
\@glsxtr@@do@@wrglossary 140
\@glsxtr@abbreviationsdef 20, 31
\@glsxtr@abbrlists 136
\@glsxtr@accessdisplay 214, 215
\@glsxtr@acronymlists 136, 137

\@glsxtr@activate@initialtagging .. 222, 223
\@glsxtr@addabbreviationlist 233
\@glsxtr@addunitcounter 128
\@glsxtr@addunused 66
\@glsxtr@addunusedxrefs 66
\@glsxtr@altmodifier 106
\@glsxtr@assign@leveloffset 146
\@glsxtr@attrval 84, 208–214, 216, 221
\@glsxtr@autoindex@at 217, 218
\@glsxtr@autoindex@doextra@esc 217
\@glsxtr@autoindex@encap ... 216, 218, 219
\@glsxtr@autoindex@esc 217, 219, 220
\@glsxtr@autoindex@escat 217, 218
\@glsxtr@autoindex@escencap ... 218, 219
\@glsxtr@autoindex@esclevel ... 218, 219
\@glsxtr@autoindex@escquote ... 217, 219
\@glsxtr@autoindex@level 218, 219
\@glsxtr@autoindex@setname 216
\@glsxtr@autoindexcrossrefs 14, 17, 57, 64
\@glsxtr@autoseeindexfalse 14
\@glsxtr@autoseeindextrue 17
\@glsxtr@base@acrcmd 94–99, 136, 137
\@glsxtr@bibgls@removespaces 444
\@glsxtr@bookindex@atendgroup 529, 530, 532
\@glsxtr@bookindex@atsubendgroup .. 531
\@glsxtr@bookindex@atsubsubendgroup 531
\@glsxtr@bookindex@between . 529, 530, 532
\@glsxtr@bookindex@sep 529–531
\@glsxtr@bookindex@subatendgroup .. 529, 530, 532
\@glsxtr@bookindex@subbetween . 529–531
\@glsxtr@bookindex@subsep 529–531
\@glsxtr@bookindex@subsubatendgroup 530, 532
\@glsxtr@bookindex@subsubbetween .. 529–531
\@glsxtr@bookindexgroupskip ... 530, 532
\@glsxtr@cat 122, 133, 177, 222
\@glsxtr@check@bibgls@nameref 159
\@glsxtr@checkgroup 166, 169
\@glsxtr@counterrecordhook 10–12
\@glsxtr@csname 128, 129, 131
\@glsxtr@current@style 73, 526
\@glsxtr@currentunitcount .. 128, 129, 131
\@glsxtr@currunitcount 130, 132
\@glsxtr@debugnr 27
\@glsxtr@debugval 27
\@glsxtr@declareoption ... 4, 18, 20, 23, 26
\@glsxtr@defaultnoglossarywarning .. 24
\@glsxtr@defaultnumberformat 6, 8, 82, 85, 103, 213, 216
\@glsxtr@defpostpunc 18, 19, 28
\@glsxtr@deprecated@abbrstyle 282, 284, 285, 288, 299, 300, 302, 305, 319, 323, 327, 329
\@glsxtr@dialect 48, 49
\@glsxtr@disabledflycommand 72
\@glsxtr@display@loc 151
\@glsxtr@do@@wrindex 104, 105
\@glsxtr@do@autoadd 82
\@glsxtr@do@glsdisablehyperinlist . 102
\@glsxtr@do@inc@linkcount 183
\@glsxtr@do@nameref@record 10, 11
\@glsxtr@do@org@target 107
\@glsxtr@do@record@wrglossary 7, 14
\@glsxtr@do@redef@forglsentries 6
\@glsxtr@do@style 25, 437
\@glsxtr@do@titlecaps@warn 208–211, 215, 223
\@glsxtr@doabbreviationsdef 20
\@glsxtr@doaccsupp 23, 28
\@glsxtr@docdefs@multilist 393
\@glsxtr@docdefsetting 17, 67
\@glsxtr@docdefval 16, 17, 67, 69, 392, 393, 396
\@glsxtr@docounterrecord 12
\@glsxtr@doglossary 166, 167, 169
\@glsxtr@doiflabelinlist 170
\@glsxtr@doloadprefix 23, 28, 31
\@glsxtr@doloctag 75, 76
\@glsxtr@dorecord 7, 9
\@glsxtr@dorecordnodefer 7, 9
\@glsxtr@dosee@alsoindex@glossary .. 16
\@glsxtr@doseeglossary 15, 30
\@glsxtr@doshowtarget 27–29
\@glsxtr@dostylewarn 250
\@glsxtr@enabletagging 222
\@glsxtr@end@ 70
\@glsxtr@endescspch 217–220
\@glsxtr@entrycount@org@localreset 123
\@glsxtr@entrycount@org@localunset 123
\@glsxtr@entrycount@org@reset 123
\@glsxtr@entrycount@org@unset 122
\@glsxtr@entryunitcount@org@localreset 131

\@glsxtr@entryunitcount@org@localunset 131
\@glsxtr@entryunitcount@org@reset . 131
\@glsxtr@entryunitcount@org@unset 130, 131
\@glsxtr@equationsfalse 18
\@glsxtr@err@undefaction 6, 15
\@glsxtr@field@linkdefs 78, 79
\@glsxtr@floatsfalse 18
\@glsxtr@format@overridefalse 215
\@glsxtr@format@overridetrue 216
\@glsxtr@foundinlist 228
\@glsxtr@full 236, 430
\@glsxtr@fullpl 237
\@glsxtr@get@prefixedlabel 449, 450
\@glsxtr@gettype 142
\@glsxtr@glossdescfont 208, 209
\@glsxtr@glossnamefont 209–215
\@glsxtr@glosssymbolfont 221
\@glsxtr@gobbleto@endescspch 220
\@glsxtr@groupheading 166, 169, 172
\@glsxtr@idx@displaynumberlist 141
\@glsxtr@idx@entrynumberlist 141
\@glsxtr@if@record@only 139, 162
\@glsxtr@ifcsstart 70
\@glsxtr@ifischild 172, 173
\@glsxtr@ifnum@mmode 81
\@glsxtr@ifpunctoken 228
\@glsxtr@ifunitcounter 128
\@glsxtr@insert@dots 229
\@glsxtr@insert@dots@next 230
\@glsxtr@insertdots 193, 232
\@glsxtr@label 39, 66, 183, 207
\@glsxtr@labellist 66
\@glsxtr@labelprefixes 448, 449
\@glsxtr@leveloffset 146, 172
\@glsxtr@loadstyles 487, 488
\@glsxtr@local@textformat 82–84
\@glsxtr@locale 48, 49
\@glsxtr@longnewglossaryentry 51
\@glsxtr@mark@wordseps 230
\@glsxtr@mark@wordseps@next 230
\@glsxtr@markwordseps 231–233
\@glsxtr@mgls@or@gls@gcs 427
\@glsxtr@mgls@or@gls@mcs 427
\@glsxtr@mglslabel 425
\@glsxtr@mglslike 426
\@glsxtr@mglslikelist 426
\@glsxtr@mglsreflist 425
\@glsxtr@mglsrefs 425
\@glsxtr@mixed@assign@sortkey 142
\@glsxtr@multientry 397
\@glsxtr@multilabel 66
\@glsxtr@newglslike 174
\@glsxtr@noidx@displaynumberlist .. 141
\@glsxtr@noidx@do 171
\@glsxtr@noidx@entrynumberlist 141
\@glsxtr@noidx@numberlistloop 141
\@glsxtr@nomissingglstextnr 24
\@glsxtr@nomissingglstextval 24
\@glsxtr@noop@recordcounter 12, 15
\@glsxtr@nopostpunc 144
\@glsxtr@nopostpunc@postdesc .. 144, 145
\@glsxtr@notfoundinlist 228
\@glsxtr@op@recordcounter 14, 16
\@glsxtr@optlist 72
\@glsxtr@org@@starttoc 365
\@glsxtr@org@GLS@ 79
\@glsxtr@org@GLSpl@ 79
\@glsxtr@org@Gls@ 78
\@glsxtr@org@Glspl@ 79
\@glsxtr@org@Glsxtrtitlefirst . 366, 368
\@glsxtr@org@Glsxtrtitlefirstplural 366, 368
\@glsxtr@org@Glsxtrtitlefull .. 367, 368
\@glsxtr@org@Glsxtrtitlefullpl 367, 368
\@glsxtr@org@Glsxtrtitlelong .. 366, 368
\@glsxtr@org@Glsxtrtitlelongpl 366, 368
\@glsxtr@org@Glsxtrtitlename .. 366, 367
\@glsxtr@org@Glsxtrtitleplural 366, 367
\@glsxtr@org@Glsxtrtitleshort . 366, 367
\@glsxtr@org@Glsxtrtitleshortpl 366, 367
\@glsxtr@org@Glsxtrtitletext .. 366, 367
\@glsxtr@org@MakeUppercase 366, 367
\@glsxtr@org@addtoacronynlists 134, 137
\@glsxtr@org@checkfirsthyper .. 101, 137
\@glsxtr@org@currentfieldvalue .. 47, 48
\@glsxtr@org@delimN 75, 76
\@glsxtr@org@delimR 75, 76
\@glsxtr@org@doseeglossary 30, 140
\@glsxtr@org@gloautosee 30, 31
\@glsxtr@org@glolinkprefix 82, 84
\@glsxtr@org@gls@ 78
\@glsxtr@org@glsdohypertarget . 146, 147
\@glsxtr@org@glsignore 75, 76
\@glsxtr@org@glspl@ 78
\@glsxtr@org@glsxtrtitlefirst . 366, 368

\@glsxtr@org@glsxtrtitlefirstplural 366, 368
\@glsxtr@org@glsxtrtitlefull .. 366, 368
\@glsxtr@org@glsxtrtitlefullpl 366, 368
\@glsxtr@org@glsxtrtitlelong .. 366, 368
\@glsxtr@org@glsxtrtitlelongpl 366, 368
\@glsxtr@org@glsxtrtitlename .. 366, 367
\@glsxtr@org@glsxtrtitleorpdforheading 366, 367
\@glsxtr@org@glsxtrtitleplural 366, 367
\@glsxtr@org@glsxtrtitleshort . 366, 367
\@glsxtr@org@glsxtrtitleshortpl 366, 367
\@glsxtr@org@glsxtrtitletext .. 366, 367
\@glsxtr@org@makeglossaries ... 138, 139
\@glsxtr@org@markboth 365
\@glsxtr@org@markright 365
\@glsxtr@org@newacronymstyle .. 135, 137
\@glsxtr@org@postdescription 145, 223, 224
\@glsxtr@org@see@noindex 159
\@glsxtr@org@setacronymlists 137
\@glsxtr@org@setacronymstyle .. 135, 137
\@glsxtr@org@theHvalue 7–9
\@glsxtr@org@unset@buffer 119, 120
\@glsxtr@orgprefix 10
\@glsxtr@orgprintglossary 72, 145
\@glsxtr@orgwarndep 231
\@glsxtr@p@acrlong@ 110
\@glsxtr@p@acrlongpl@ 110
\@glsxtr@p@acrshort@ 110
\@glsxtr@p@acrshortpl@ 110
\@glsxtr@p@long@ 110
\@glsxtr@p@longpl@ 110
\@glsxtr@p@plural@ 110
\@glsxtr@p@short@ 110
\@glsxtr@p@shortpl@ 110
\@glsxtr@p@text@ 110
\@glsxtr@pagestag 75, 76
\@glsxtr@pagetag 75, 76
\@glsxtr@prefix 449
\@glsxtr@prevunitcount 130
\@glsxtr@printglossnr 145
\@glsxtr@printglossopts . 72, 142, 145, 167
\@glsxtr@printglossval 145
\@glsxtr@printunsrtglossaryskipentry 166, 169, 170
\@glsxtr@provide@addstoragekey 35
\@glsxtr@provide@storagekey 34
\@glsxtr@providenewgls 174
\@glsxtr@record 14–16, 78, 79, 85, 236, 238–246
\@glsxtr@record@noglossarywarning .. 15
\@glsxtr@record@only@setup 15, 16
\@glsxtr@record@setting 7, 9–11, 14, 15, 32, 63, 68, 69, 106, 139, 157–159, 161, 162, 174, 425, 426
\@glsxtr@record@setting@alsoindex 7, 9, 15, 16, 63, 139
\@glsxtr@record@setting@nameref 10, 11, 14, 32, 159, 161
\@glsxtr@record@setting@off 68, 106, 174, 425, 426
\@glsxtr@record@setting@only 14, 32
\@glsxtr@recordsee 14, 30, 63
\@glsxtr@redef@forglsentries 6, 31
\@glsxtr@redefstyles 24, 25, 437
\@glsxtr@reg@glosslist 139–142, 147
\@glsxtr@restore@postpunc 144
\@glsxtr@rglstrigger@record ... 179, 180
\@glsxtr@s@longnewglossaryentry 51
\@glsxtr@saveprelocatag 75, 76
\@glsxtr@seefirstitem 61
\@glsxtr@seeitem 61
\@glsxtr@setentrycountunsetattr ... 121
\@glsxtr@setentryunitcountunsetattr 133
\@glsxtr@setupshortcuts 22, 23, 31
\@glsxtr@shortcutsnr 22
\@glsxtr@shortcutsval 22, 161
\@glsxtr@showtargetsnr 27, 28
\@glsxtr@showtargetsval 27
\@glsxtr@swaptwo 228
\@glsxtr@tag 222
\@glsxtr@taggingcs 222
\@glsxtr@textformat 84
\@glsxtr@theHentrycounter 10
\@glsxtr@theHvalue . 7–9, 80, 83, 85, 177, 178
\@glsxtr@theentrycounter 10
\@glsxtr@thevalue .. 7–9, 80, 83, 85, 177, 178
\@glsxtr@thisloctag 76
\@glsxtr@titlelabel 149, 150, 171
\@glsxtr@tmp 24, 25, 81, 152
\@glsxtr@type 207
\@glsxtr@unitcountlist 128
\@glsxtr@unset 119, 120
\@glsxtr@unset@buffer 119, 120
\@glsxtr@unsrt@getgrouptitle .. 166, 169
\@glsxtr@use@equation@counter . 8, 81, 83
\@glsxtr@usesee 58

\@glsxtr@warn@hybrid@noprintgloss . 139
\@glsxtr@warn@onexistsordo 6, 14, 16
\@glsxtr@warn@undefaction 6, 14, 16
\@glsxtr@wrglossary@locationhyperlink 26
\@glsxtr@wrglossnr 80
\@glsxtr@wrglossval 80
\@glsxtrbuffer@nodup@unset 119
\@glsxtrbuffer@unset 119
\@glsxtrdialecthook 437
\@glsxtrdocdeffalse 69
\@glsxtrentryfmt 37
\@glsxtrfmt 35
\@glsxtrforcsvfield 38
\@glsxtrglossentry 163
\@glsxtrglossentryother 164
\@glsxtrhypernameprefix 145, 146, 171
\@glsxtrifhasfield 39–41, 46, 47
\@glsxtrifhyphenstart 344
\@glsxtrindexaliased 103
\@glsxtrindexcrossrefsfalse 17
\@glsxtrindexcrossrefstrue 17
\@glsxtrinmark 365
\@glsxtrlong 110, 241, 430
\@glsxtrlongpl 110, 245
\@glsxtrmultientryadjustedname 451, 452
\@glsxtrmultientryadjustednamefirstfmt 451, 452
\@glsxtrmultientryadjustednamefirstother 451, 452
\@glsxtrmultientryadjustednamefmt 451, 452
\@glsxtrmultientryadjustednameother 451, 452
\@glsxtrmultientryadjustednamepostsep 451, 452
\@glsxtrmultientryadjustednamepresep 451, 452
\@glsxtrmultientryadjustednamesep 451–453
\@glsxtrnewgls 175, 176
\@glsxtrnewgls@inner 174
\@glsxtrnewgls@innercsname 174, 175
\@glsxtrnotinmark 365
\@glsxtrp 115, 116
\@glsxtrp@opt 113
\@glsxtrpl 71, 72
\@glsxtrpostloctag 74, 75, 77
\@glsxtrpreloctag 74, 75, 77
\@glsxtrsetaliasnoindex 103, 104
\@glsxtrshort 110, 239, 430
\@glsxtrshortpl 110, 243
\@glsxtrshowtargetleft 27, 28
\@glsxtrshowtargetmark 27, 28
\@glsxtrshowtargetright 28
\@glsxtrundeftag 5, 33
\@glsxtrwrglossmark 26, 27
\@gobble .. 6, 15, 18, 86, 136, 140, 170, 171, 230, 405, 406, 410, 411, 418, 489, 529–532
\@gobblefour 397
\@gobbletwo 137, 231, 393
\@gtempa 439
\@ifdefinable 396, 440
\@ifglossaryexists 33
\@ifnextchar 105, 146, 427
\@ifpackageloaded 4, 20, 161, 183, 208, 209, 212, 213, 215, 437, 455, 486
\@ifstar 33–35, 38–42, 46, 47, 51, 52, 54, 69, 105, 119, 165, 222, 427, 439
\@ifundefined 437, 439
\@ignored@glossaries 53–55
\@input 159
\@input@ 154
\@istfilename 139
\@makeglossaries@warn@noprintglossary 140
\@makeglossary 139, 140
\@mfu@domakefirstuc 222, 223
\@mfu@nocaplist 223
\@mgls@all 407, 416, 417
\@mgls@attroptions 409
\@mgls@combinedlink 410
\@mgls@cs 417
\@mgls@current@iffirstuse 415–418
\@mgls@current@options 415–417
\@mgls@disable@writeseparateref@cond 425
\@mgls@do@current@element 405, 406, 410, 411, 415, 418
\@mgls@dooptions 407, 409, 410
\@mgls@hyper 415–417
\@mgls@hyper@nr 401
\@mgls@hyper@val 401
\@mgls@hyperlink 402, 410
\@mgls@main 416
\@mgls@options 409
\@mgls@others 417

\@mgls@post@hookdefs 408, 411, 412, 414, 417–419
\@mgls@previous@iffirstuse 417, 418
\@mgls@previouslabel 413, 417, 418, 452, 453
\@mgls@resetall 404, 407
\@mgls@resetall@nr 403
\@mgls@resetall@val 403
\@mgls@resetmain 404, 407
\@mgls@resetmain@nr 403
\@mgls@resetmain@val 403
\@mgls@resetothers 405, 407
\@mgls@resetothers@nr 403, 404
\@mgls@resetothers@val 403
\@mgls@setup 407, 409
\@mgls@setup@do 401
\@mgls@setup@do@not 401
\@mgls@setupoptions 407, 409, 410
\@mgls@thislabel 398, 399
\@mgls@unsetaction 419
\@mgls@unsetaction@val 401
\@mgls@unsetall 403, 407
\@mgls@unsetall@nr 404
\@mgls@unsetall@val 404
\@mgls@unsetmain 403, 407
\@mgls@unsetmain@nr 404
\@mgls@unsetmain@val 404
\@mgls@unsetothers 404, 407
\@mgls@unsetothers@nr 404
\@mgls@unsetothers@val 404
\@mgls@writeseparaterefsfalse 425
\@mgls@writeseparaterefstrue 425
\@mglslocalreset 398
\@mglslocalunset 398
\@mglsreset 397
\@mglsunset 397
\@multi@glossary@doifexists ... 394, 397
\@multi@glossary@entry 392, 393, 397
\@multi@glossaryentry 392, 393
\@multi@glossaryentry@list . 396, 398, 399
\@multi@glossentry@donext 392, 393
\@multiglossaryentry 392
\@ne 124, 133, 174
\@newglossaryentry@defcounters 122, 130
\@newglossaryentryposthook . 13, 34, 64, 99
\@newglossaryentryprehook 13, 34, 52, 63, 99
\@nil 153, 172, 444
\@nnil 61, 217, 218, 220, 228–230, 413
\@no@glsxtrindexaliased 103, 104
\@no@makeglossaries 158
\@nocounterr 183
\@nopostdesc 144, 168
\@onelevel@sanitize . 10, 12, 63, 72, 150, 171
\@onlypreamble 73, 75, 132, 159, 162, 183, 216, 218, 219, 222
\@org@glossaryentrynumbers 143, 144, 167, 168
\@org@newglossaryentryprehook 52
\@pgls@ 434
\@pglsorgls@ 435
\@pglsorglspl@ 435
\@pglspl@ 434
\@print@unsrt@glossary 165
\@print@unsrt@innerglossary 167
\@printgloss@checkexists 143, 165
\@printgloss@checkexists@allowignored 165
\@printgloss@setsort 142, 143, 168
\@printglossary 72, 165
\@printunsrt@glossary@handler . 167, 169
\@printunsrtglossary 165
\@provide@multi@glossaryentry@noop 393
\@rGLS 180
\@rGLS@ 180
\@rGLSpl 180
\@rGLSpl@ 180
\@rGls 179
\@rGls@ 179
\@rGlspl 179
\@rGlspl@ 180
\@rc@ifdefinable 440
\@rgls 178
\@rgls@ 178
\@rglspl 179
\@rglspl@ 179
\@sGlsXtrEnableOnTheFly 69
\@secondofthree 87–89, 94, 96–98, 100, 237, 238, 240, 242, 244, 246, 418, 419
\@secondoftwo 27, 79, 86, 89–98, 101, 109, 137, 145, 172, 214, 228, 236, 237, 239–246, 262, 264, 287, 289, 304, 306, 328, 330, 366, 367, 397, 402, 406, 408, 409, 411–413, 415, 418, 419, 489, 490
\@sglsxtr@provide@storagekey 34
\@star@or@long 439
\@starttoc 365
\@thirdofthree 87–89, 95–98, 101, 237, 239, 241, 242, 244, 246, 367, 418, 419
\@thirdoftwo 90–94

\@this@key 214
\@tracklang@lang 48, 49
\@warn@nomakeglossaries 154
\@xGlsXtrIfValueInFieldCsvList 40
\@xdy@main@language 154
\@xdycrossrefhook 62
\@xdylanguage 154
\@xdylocationclassorder 62
\@xfor@nextelement 61, 413
[package 442
\\ 153, 444

\␣ 69, 156

A

\AA 469
\aa 469
\AB 20
\Ab 20
\ab 20
abbreviation styles:
  footnote 261
  long-hyphen-postshort-hyphen 349, 350, 352
  long-hyphen-short-hyphen 346, 350
  long-postshort-sc-user 338
  long-postshort-user 337
  long-short 192
  long-short-sc-user 335
  long-short-user 333
  nolong-short 269
  short 266
  short-em-footnote 327
  short-em-postfootnote 330
  short-footnote 260, 302
  short-hyphen-long-hyphen 354, 356
  short-hyphen-postlong-hyphen 355, 356, 358
  short-long-user 338
  short-nolong 266, 268
  short-nolong-desc 268
  short-postfootnote 263
  short-postlong-user 340
  short-sc-footnote 285, 288
  short-sm-postfootnote 305
\abbreviationsname 20
\abbrvpluralsuffix .....
  . 161, 232, 254, 256, 259, 262, 264, 267, 270, 273, 275, 277, 279, 281, 282, 284, 287, 290, 292, 294, 295, 298, 299, 301, 304, 307, 309, 311, 313, 314, 316, 318, 320, 322, 323, 326, 329, 332, 334, 336, 339, 342, 345, 347, 351, 354, 357, 360, 362
\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
\ACRfull 99
\Acrfull 99
\acrfull 98
\ACRfullfmt 99, 135
\Acrfullfmt 99, 135
\acrfullfmt 99, 135
\ACRfullpl 99
\Acrfullpl 99
\acrfullpl 99
\ACRfullplfmt 99, 135
\Acrfullplfmt 99, 135
\acrfullplfmt 99, 135
\ACRlong 97
\Acrlong 97
\acrlong 96
\ACRlongpl 98
\Acrlongpl 98
\acrlongpl 97
\acronymentry 134
\acronymfont 94–98, 112, 113, 136, 137
\acronymname 20
\acronymsort 134
\acronymtype 20, 134, 136, 438
\acrpluralsuffix 134, 161
\ACRshort 95
\Acrshort 94

\acrshort 94
\ACRshortpl 96
\Acrshortpl 95
\acrshortpl 95
\ACS 21
\Acs 21
\acs 21
\ACSP 21
\Acsp 21
\acsp 21
\actualchar 219
\addtolength 519
\advance . 124, 133, 146, 160, 174, 394, 399, 413
\AF 21
\Af 20
\af 20
\AFP 21
\Afp 20
\afp 20
\AL 21
\Al 20
\al 20
\allowbreak 29
\ALP 21
\Alp 20
\alp 20
\alsoname 62
\alt@GlsXtrMglsOrGls 427
amsmath package 11, 26
\AnyTrackedLanguages 437, 486
\appto 13, 25, 34, 57, 62–65, 99, 104, 105, 122, 130, 166, 167, 169, 171, 216, 227, 230, 232, 415, 416, 448, 487
\apptoglossarypreamble 445–447
\arabic 26, 532
\AS 21
\As 20
\as 20
\ASP 21
\Asp 20
\asp 20
\AtBeginDocument 26, 33, 73, 74, 161, 449
\AtEndDocument ... 65, 123, 131, 154, 425, 426

B

babel package 216, 218, 227
\begin 151, 156, 166, 168, 490, 492–499, 521, 522, 524, 525, 529, 538–552, 554, 555, 562
\begingroup . 7, 8, 36, 38, 82, 85, 103, 163–165, 167, 182, 193, 221, 439, 443, 449, 558
\bgroup 52, 143, 397, 408, 451, 452
bib2gls 26, 36, 44, 51, 101, 104, 106, 159, 166, 169, 171, 174, 177, 178, 396, 424, 426, 437–441, 445, 447, 449–451, 454
\bibglsdelimN 438
\bibglshrefchar 159
\bibglslastDelimN 438
\bottomrule 536, 540–542, 544, 545, 547, 549, 550, 552, 553, 555

C

\c@wrglossary 26
\capitalisewords 453
\catcode 67, 159
category attributes:
  accessinsertdots 193
  aposplural 232
  discardperiod 226
  entrycount 118, 121–123, 133
  externallocation 441
  firstshortaccess 195, 258, 261
  firstuc 211
  glossdesc 207
  glossdescfont 208
  glossname 209
  glossnamefont 209, 212
  headuc 368
  indexname 217
  indexonlyfirst 104
  insertdots 232
  linkcount 182
  linkcountmaster 182
  markshortwords 232
  markwords 231, 233, 344, 345, 353
  nameshortaccess 194, 196
  nohyper 102
  nohyperfirst 87–89
  noshortplural 232
  regular .. 77, 127, 253, 255–258, 260, 261, 263, 265–272, 274–276, 278, 279, 282–284, 286–288, 291–294, 296, 298, 300, 301, 303, 305, 308–310, 312, 314–316, 319, 321, 323–328, 330, 332, 341–343, 345–347, 349, 354, 355, 359, 361, 362, 364
  textformat 83
  textshortaccess 195
\cdot 26
\centering 528
\cGLS 20, 21, 121, 133

\cGls 20, 21, 121, 133
\cgls 20, 21, 121, 133
\cGLSformat 125
\cGlsformat 125
\cglsformat 124, 126
\cGLSpl 20, 21, 121, 133
\cGlspl 20, 21, 121, 133
\cglspl 20, 21, 121, 133
\cGLSplformat 126
\cGlsplformat 125
\cglsplformat 125, 127
\char 149
\columnwidth 74
\count@ 123, 124, 132, 133, 399
\csappto 44, 50, 400
\csdef 34, 44, 45, 50, 99–101, 123, 128, 129, 131, 196, 201, 213, 224, 225, 250–252, 261, 263, 287, 288, 303, 305, 328, 330, 333, 335, 337–340, 351, 352, 356, 358, 400, 420, 488, 509
\csedef 129, 394, 395, 399, 400, 560
\csgdef 45, 53–55, 68, 75, 123, 129, 131, 132, 136, 201, 509, 532
\cslet 45, 52, 143, 394–396, 399, 400
\csletcs 45, 252, 395
\csname 5, 35, 54, 63, 68, 73, 77, 79, 82, 85, 94–98, 100, 101, 103, 113, 129, 140, 150, 154, 157, 158, 166–169, 174–178, 182, 207, 231, 236–246, 252, 396–398, 402, 426, 490, 518
\cspreto 50
\csundef 202
\csuse 8, 11, 36, 37, 44, 51, 54, 65, 75, 100, 101, 115–117, 128–130, 132, 143, 149, 151, 152, 163, 164, 171, 172, 202, 213, 224, 225, 250, 252, 391, 396, 410, 420, 421, 442, 444, 447, 509, 510, 512, 513, 536, 560
\csxdef ... 57, 64, 129, 132, 394, 395, 399, 400
\currentglossary 143, 163, 164, 167, 168, 529, 532
\CurrentOption 28, 487, 488
\CurrentTrackedLanguage 485
\CurrentTrackedLanguageTag 161
\CurrentTrackedScript 485, 486
\CurrentTrackedTag 437, 486
\CustomAbbreviationFields 233, 253, 255, 257, 258, 260, 261, 263, 264, 266, 269, 272–278, 280, 284, 286, 288, 290–295, 297, 301–303, 305, 306, 308, 310–314, 316, 318, 320, 323, 325, 327, 328, 330, 332, 333, 335, 337, 338, 340–343, 345–347, 349, 350, 352, 354–356, 358, 359, 361, 362, 364

D

datatool-base package 11
\DeclareAcronymList 134
\DeclareOption 4, 487
\DeclareOptionX 4, 28
\def 8, 10, 12–17, 27–29, 32, 33, 36, 38, 43, 52, 54, 58, 63, 66, 70–72, 74, 77–83, 85, 87–99, 106, 109–113, 119, 124–127, 134, 140, 142, 143, 145–153, 166, 167, 169, 171, 172, 174, 177–180, 191–194, 217–220, 222, 223, 227–232, 236–246, 250, 344, 347, 349, 350, 353, 355, 356, 392, 393, 397, 409–411, 413, 414, 417, 426, 427, 430, 432–434, 442–444, 448, 449, 452, 486, 501, 518–520, 524–526, 529, 530, 536, 537, 557, 561, 562
\defglsentryfmt 53–55
\define@boolkey 17, 18, 80, 102, 146, 390, 391, 401
\define@choicekey 6, 15, 17, 19, 22, 24, 27, 80, 145, 388, 389, 391, 401, 403, 404
\define@key .. 13, 19, 24, 25, 34, 63, 80, 81, 85, 99, 145, 146, 191, 229, 389, 390, 400, 401
\DefineAcronymSynonyms 22
\delimN 75, 76
\delimR 75, 76
\descriptionname 536, 539, 541, 542, 544, 545, 547, 548, 550, 552, 553, 555
\detokenize 70
\dimen@ 138, 423, 424, 509–517, 537
\dimen@i 512, 513
\dimen@ii 509, 510, 512, 513, 537
\dimexpr 73, 74, 508, 537, 538, 559
\disable@keys 20, 31, 33, 69, 159
\do 5, 24, 39, 61, 66, 86, 121, 122, 133, 136, 137, 139, 142, 150, 166, 169, 177, 183, 201, 207, 214, 222, 394, 398, 399, 402–405, 413, 449, 452
\do@gls@link@checkfirsthyper 36, 78, 79, 83, 94–98, 236–246
\do@glsdisablehyperinlist 83, 103
doc package 219
\dolistcsloop 38

\DTLifinlist 39–41, 82, 107, 136, 140, 141, 147, 425, 448
\DTLifint 149

E

\eappto 24, 194, 195, 396, 407, 409, 415–417, 487
\edef 10, 48, 67, 149, 153, 154, 159, 182, 217, 218, 220, 407, 409, 410, 415, 416, 426, 444, 449, 485, 529–531, 538–552, 554, 555, 562
\eglssetwidest 511–517
\egroup 52, 144, 397, 419, 451, 452
\else 7–12, 14, 17–20, 22–24, 29, 30, 32, 36, 43, 49, 50, 61, 67–70, 75, 77, 84, 102–104, 124, 136, 138, 139, 141, 143–145, 149, 151–153, 155–157, 160, 168, 172, 177, 178, 216–218, 220, 228, 230, 232, 239–246, 249, 254, 256, 257, 259, 260, 262, 263, 265, 267–271, 274, 276–285, 287, 288, 290–302, 304, 305, 307, 309, 311, 313, 315–327, 329, 332–334, 336, 339, 340, 342–344, 347, 350, 352, 353, 356–358, 360, 362, 363, 392–396, 399, 400, 402–407, 409, 410, 413, 414, 416–418, 423–426, 444–447, 452, 490, 493–500, 502, 505, 506, 518, 519, 526, 528–531, 539–556, 561
\emph 192, 306
\empty 28, 151–153, 192, 409, 417, 444, 448, 452
\encapchar 220
\end 151, 156, 167, 168, 490, 492–499, 521, 522, 524, 525, 530, 539–546, 548, 549, 551–556, 562
\end@getprefix 32
\end@glsxtr@display@loc 151
\endcsname 5, 35, 54, 63, 68, 73, 77, 79, 82, 85, 94–98, 100, 101, 103, 113, 129, 140, 150, 154, 157, 158, 166–169, 174–178, 182, 207, 231, 236–246, 252, 396–398, 402, 426, 490, 518
\endfoot 536, 539, 541, 542, 544, 545, 547, 548, 550, 551, 553, 555
\endgroup 7, 9, 36, 38, 82, 86, 103, 163, 165, 167, 182, 193, 221, 439, 443, 444, 449, 558
\endhead 536, 539, 541, 542, 544, 545, 547, 548, 550, 551, 553, 555
\ensuremath 26
entry categories:
  abbreviation 231, 246
  general 200, 203
  index 205
  nameshortaccess 316
\entryname 536, 537, 539, 541, 542, 544, 545, 547, 548, 550, 552, 553, 555
\epreto 217
\equal 157, 158, 225
equation (counter) 18, 81
\escapechar 439
etoolbox package 4
\everypar 559
\expandafter 28, 35, 36, 39–41, 48, 58, 60, 61, 66, 70–72, 81, 100, 101, 105, 107, 113, 119, 126–128, 136, 140–142, 147, 150, 152, 153, 166, 168, 169, 172–175, 181, 193, 194, 207, 210, 211, 214–216, 219, 220, 228, 231–233, 344, 396–398, 402, 417, 425, 427, 439, 444, 490
\expandonce .. 134, 172, 193, 194, 217, 218, 254, 347, 393, 396, 407, 409–411, 414, 417, 419, 425, 426, 490, 538–552, 554–556
\ExtraCustomAbbreviationFields 194, 195, 231, 233

F

\fi 6–12, 14, 16–20, 22–24, 27–32, 36, 43, 50, 57, 61, 63–65, 67, 69, 70, 73–75, 77, 80, 83, 84, 102–104, 124, 132, 133, 136, 138, 139, 142–146, 149, 151–162, 166–169, 172, 177, 178, 216–218, 220, 228, 230, 233, 239–246, 249, 254, 256, 257, 259, 260, 262, 263, 265, 267–271, 274–285, 287, 288, 290–302, 304, 305, 307, 309, 311, 313–327, 329, 332–334, 336, 339, 340, 342–344, 347, 350, 352, 353, 356–358, 360, 362, 363, 392–397, 399, 400, 402–407, 410–419, 423–426, 438, 444–447, 452, 490, 493–500, 502, 504–506, 509–519, 526, 528–531, 537, 539–551, 553, 554, 556, 557, 561, 562
\firstacronymfont 136–138
fontspec package 161
\footnote 258
\footnotesize 29
\forallabbreviationlists 136
\forallglossaries 66, 165, 205, 207, 510–517
\forallglsentries 67, 123, 132
\ForEachTrackedDialect 437, 486
\foreignlanguage 47, 49
\forglsentries 5, 66, 205, 207, 510–517
\forlistcsloop 38, 132, 151

\forlistloop 120, 148, 223
\futurelet 228

G

\gappto 411, 412, 418, 419
\gdef 76, 107, 218, 219, 393, 408, 425, 426
\Genacrfullformat 135
\genacrfullformat 135
\GenericAcronymFields 134
\Genplacrfullformat 135
\genplacrfullformat 135
\GetTitleStringDisableCommands 489
\GetTitleStringSetup 489
\GetTrackedDialectFromLanguageTag 47, 48
\GetTrackedDialectToMapping 48
\glo@grabfirst 172
\glo@name 210, 211, 215
\gloaliaslabel 108
\global . 10, 52, 67, 144, 168, 172, 394–399, 425
\glolinkprefix 81, 82, 84, 108, 109, 145, 161, 402
glossaries package 11, 15, 29, 30, 32, 49, 51, 57, 62, 63, 65, 102, 106, 138, 139, 142, 143, 165, 192, 207, 488, 490
glossaries-accsupp package 23, 28, 183, 192, 233
glossaries-extra package 2, 51, 486
glossaries-extra-bib2gls package 15, 32, 437, 486, 537
glossaries-extra-stylemods package 24, 224, 437, 445
glossaries-prefix package 23, 28, 434, 449
glossaries-stylemods package 527
glossaries.sty package 52
\GlossariesExtraWarning 5, 16, 18, 24, 32, 48–50, 70, 72, 84, 94, 137, 141, 152, 156, 158, 159, 166, 169, 208–214, 221–223, 252, 407, 434, 439, 447
\GlossariesExtraWarningNoLine 14, 18, 124, 133, 162
\GlossariesWarning 75, 134, 141, 143, 148, 149, 167, 250
\GlossariesWarningNoLine ... 107, 138, 154
glossary styles:
  altlist 490
  altlistgroup 491
  altlisthypergroup 492
  alttree 445, 446, 503, 507, 508, 518, 536
  alttreegroup 519
  alttreehypergroup 519
  index 501
  indexgroup 502
  indexhypergroup 502
  inline 500
  list 490
  listdotted 488
  listdottedstyle 489
  listgroup 491
  listhypergroup 491
  longragged-booktabs 538
  mcolalttree 524
  mcolalttreegroup 524
  mcolalttreehypergroup 525
  mcolalttreespannav 525
  mcolindexgroup 520
  mcolindexhypergroup 521
  mcolindexspannav 521
  mcoltreegroup 521
  mcoltreehypergroup 522
  mcoltreenonamegroup 523
  mcoltreenonamehypergroup 523
  mcoltreenonamespannav 523
  mcoltreespannav 522
  name-desc 541
  sublistdotted 489
  tree 503
  treegroup 505
  treehypergroup 505
  treenoname 505
  treenonamegroup 507
  treenonamehypergroup 507
glossary-bookindex package 488
glossary-hypernav package 106
glossary-list package 489, 490
glossary-long package 494
glossary-longbooktabs package 494
glossary-longextra package 445–447
glossary-tree package 446, 447, 501, 560
\glossaryentrynumbers 77, 143, 144, 167, 168, 173
\glossaryheader 151, 166, 168, 490–499, 501, 502, 504–507, 518, 520–523, 525, 530, 539, 540, 542–546, 548, 549, 551–557, 561
\glossaryname 143
\glossarypostamble 151, 167, 168, 171
\glossarypreamble 151, 166, 168
\glossarysection 151, 157, 158, 166, 168, 171
\glossarytitle 54, 143, 151, 157, 158, 166–168

\glossarytoctitle 54, 143, 146, 151, 157, 158, 166–168
\glossentry 144, 168, 173, 488, 490, 492–499, 501, 504, 506, 518, 530, 539, 540, 542, 543, 545, 546, 548, 549, 551, 553, 554, 556, 557, 561
\Glossentrydesc 559
\glossentrydesc 488, 489, 492–500, 503–506, 534, 560
\Glossentryname 558
\glossentryname 163, 488, 489, 492–499, 502, 504, 506, 518, 519, 527, 534, 560
\glossentrynameother 165
\glossentrysymbol 494, 496–500, 504, 506, 508, 534, 558, 560
\glossxtrsetpopts 224
\GLS 121, 133, 177
\Gls 71, 121, 133, 176
\gls 49, 50, 71, 72, 121, 133, 141, 155, 176
\gls@assign@desc 52
\gls@assign@field 13, 34, 99
\gls@checkseeallowed 67, 69, 140
\gls@codepage 154
\gls@defdocnewglossaryentry . 67, 122, 130
\gls@defglossaryentry 52, 68, 71, 72
\gls@dotoctitle 143, 167, 168
\gls@glossary 63
\gls@grplabel 106
\gls@hypergrouprerun 107
\gls@ifnotmeasuring 11, 121, 397, 398
\gls@level 172, 173
\gls@noidxglossary 140
\gls@org@glossaryentryfield ... 144, 168
\gls@org@glossarysubentryfield 144, 168
\gls@orgTrackLangRequireDialectPrefix 485, 486
\gls@save@numberlist 74, 75, 77
\gls@set@xr@key 63
\gls@tmplen 510–517, 519, 537, 538
\gls@type 140
\glsabbrvdefaultfont ... 235, 254, 256, 259, 262, 264, 267, 270, 331, 344, 347, 359
\glsabbrvemfont 306–314, 316, 318, 320, 322, 323, 325–330
\glsabbrvfont 111, 112, 136, 239–241, 243, 244, 247, 249, 253–264, 266, 267, 270, 271, 273, 275, 277, 279, 281, 282, 284, 287, 290, 292, 294, 295, 297, 299, 301, 304, 307, 309, 311, 313, 314, 316, 318, 320, 322, 323, 326, 329, 332, 334, 336, 339, 341–343, 345, 347, 350, 351, 354, 357, 360, 362
\glsabbrvhyphenfont 345, 346, 350–352, 354–358
\glsabbrvonlyfont 359–361
\glsabbrvscfont 273–279, 281, 282, 284, 286–288, 334, 361
\glsabbrvsconlyfont 361, 362, 364
\glsabbrvscuserfont 335–338
\glsabbrvsmfont 289–295, 297, 299, 301, 303–305
\glsabbrvuserfont 331–334, 337, 339, 340, 342
\GLSaccessdesc 90
\Glsaccessdesc 90, 208, 220
\glsaccessdesc 90, 208, 226
\GLSaccessdescplural 91
\Glsaccessdescplural 90
\glsaccessdescplural 90
\GLSaccessfirst 88
\Glsaccessfirst 88
\glsaccessfirst 87
\GLSaccessfirstplural 89
\Glsaccessfirstplural 89
\glsaccessfirstplural 89
\Glsaccesslong 97, 234, 242, 254, 265, 270, 274, 280, 281, 283, 291, 297–300, 307, 309, 317–322, 324, 333, 334, 336, 346, 348, 352, 354, 360, 363
\glsaccesslong 96, 97, 234, 241, 242, 254, 256, 259, 260, 262, 263, 265, 267, 268, 270, 271, 274, 276–283, 285, 288, 290, 292–294, 296–302, 304, 305, 307, 309, 311, 313, 315–327, 329, 332, 334, 336, 340, 342, 343, 346, 348, 351, 352, 354, 360, 362, 363
\Glsaccesslongpl 98, 234, 246, 254, 265, 270, 274, 280–283, 291, 297–300, 307, 309, 317–324, 333, 334, 336, 346, 348, 352, 354, 360, 363
\glsaccesslongpl 97, 98, 234, 245, 246, 254, 256, 257, 259, 260, 262, 263, 265, 267, 268, 270, 271, 274, 276–283, 285, 288, 290, 292–294, 296–302, 305, 307, 309, 311, 313, 315–327, 329, 332, 334, 336, 340, 343, 346, 348, 351, 352, 354, 360, 362, 363
\GLSaccessname 90
\Glsaccessname 90

\glsaccessname 89
\GLSaccessplural 88
\Glsaccessplural 88
\glsaccessplural 88
\Glsaccessshort 94, 240, 249, 256, 259, 260, 262, 263, 267, 268, 276, 278, 279, 285, 287, 288, 292, 294, 296, 302, 304, 305, 311, 313, 315, 316, 326, 327, 329, 339, 340, 343, 351, 357, 358
\glsaccessshort 94, 95, 234, 239, 241, 249, 254, 256, 259, 262, 265, 267, 268, 270, 274, 275, 277–281, 283–285, 287, 288, 290–292, 294–298, 300–302, 304, 307, 309, 311, 313–317, 319–322, 324, 326, 329, 332–334, 336, 339, 340, 342, 346, 348, 351, 354, 357, 360, 363
\Glsaccessshortpl 96, 244, 249, 256, 259, 260, 262, 263, 267, 269, 276, 278, 279, 285, 288, 293, 294, 296, 302, 304, 305, 311, 313, 315, 316, 326, 327, 329, 339, 340, 343, 351, 357, 358, 361, 363
\glsaccessshortpl 95, 96, 234, 243, 244, 249, 254, 256, 259, 262, 265, 267, 268, 270, 274, 276–283, 285, 287, 288, 290–292, 294–298, 300–302, 304, 307, 309, 311, 313, 315–317, 319–324, 326, 327, 329, 332–334, 336, 339, 340, 343, 346, 348, 351, 354, 357, 360, 363
\GLSaccesssymbol 91
\Glsaccesssymbol 91, 221
\glsaccesssymbol 91, 221, 226
\GLSaccesssymbolplural 92
\Glsaccesssymbolplural 91
\glsaccesssymbolplural 91
\GLSaccesstext 87
\Glsaccesstext 87
\glsaccesstext 87
\glsacrshortcutstrue 22, 23
\glsacspacemax 138
\glsadd 36, 66, 82, 86, 155
\glsadd options
  theHvalue 9
  thevalue 8, 9
\glsaddpostsetkeys 9, 85
\glsaddpresetkeys 9, 85
\glsaddstoragekey 65, 200, 441
\glsaltlistitem 490
\glsalttreechildpredesc 508
\glsalttreepredesc 507, 508
\glsbackslash 70
\glscapitalisewords 207, 453
\glscapscase 79, 86–98, 100, 101, 236–248, 418–420
\glscategory 77, 86, 101, 111, 112, 202–204, 208, 209, 211–214, 220, 221, 224, 225, 236–241, 243, 244
\glscategorylabel 101, 102, 192, 194, 195, 229, 231–233, 250, 253, 255, 257, 258, 260, 261, 263, 264, 266, 271, 273–278, 284, 286–288, 290–293, 295, 297, 301–303, 305, 306, 308–314, 316, 318, 319, 321, 325, 327, 328, 330, 332, 333, 335, 337–343, 345, 346, 349–353, 355, 356, 358, 359, 361–363
\glsclosebrace 62, 156–158
\glscombinedfirstsep 417, 422, 424
\glscombinedfirstsepfirst 417, 422–424, 453
\glscombinedsep 417, 421–423
\glscombinedsepfirst 417, 422, 423
\glscounter 8, 18
\glscurrententrylabel 74–76, 144, 153, 163, 164, 166–170, 223, 224
\glscurrentfieldvalue 36, 37, 39–41, 43, 46–48, 59, 60, 331, 433, 440
\glscustomtext .. 78, 79, 94–98, 236–247, 249
\glsdefaultshortaccess 193, 194
\glsdefaulttype 5, 19, 50, 142, 143, 155, 165, 167, 170
\glsdescriptionaccessdisplay 187, 188, 208
\glsdescriptionpluralaccessdisplay 188
\glsdescwidth 492–499, 536–538
\glsdetoklabel 7, 8, 33, 37, 38, 41, 43–45, 49–52, 56, 58, 60, 61, 66, 68, 70, 82, 85, 103, 108, 122, 123, 128–132, 140, 144, 147, 148, 163, 164, 168, 172, 176, 177, 207, 210, 211, 215, 490, 512, 513
\glsdisp 450
\glsdisplaynumberlist 141, 147
\glsdohyperlink 106, 444
\glsdohypertarget 109, 145, 147
\glsdoifexists 17, 30, 44, 45, 51, 55, 58–61, 63, 78, 79, 85, 94–98, 109, 121, 140, 148, 149, 163, 164, 236–246, 393, 438
\glsdoifexistsordo 36, 37, 79
\glsdoifexistsorwarn 17, 208, 220, 221
\glsdoifnoexists 52
\glsdonohyperlink 84, 109, 402

\glsdosanitizesort 141
\glsdoshowtarget 109
\glsenableentrycount 121, 124, 132
\glsenableentryunitcount 123, 133
\glsentrycounter 26, 152, 153, 444
\GlsEntryCounterLabelPrefix 49
\glsentrycurrcount 122, 124, 130
\Glsentrydesc 188, 199, 209
\glsentrydesc 187, 188, 198, 199, 209
\Glsentrydescplural 188, 199
\glsentrydescplural 188, 199
\Glsentryfirst 127, 181, 185, 198
\glsentryfirst 127, 181, 185, 186, 198, 382, 383, 423, 424
\Glsentryfirstplural ... 127, 181, 186, 198
\glsentryfirstplural 127, 181, 186, 198, 383, 384
\glsentryfmt 53–55, 136
\Glsentryfull 135
\glsentryfull 135
\Glsentryfullpl 135
\glsentryfullpl 135
\glsentryitem ... 163, 164, 488, 489, 492–499, 501, 504, 506, 518, 530, 534, 558, 560
\Glsentrylong ... 112, 113, 127, 181, 190, 199
\glsentrylong ... 112, 113, 127, 181, 190, 199, 200, 261, 263, 287, 289, 303, 305, 328, 330, 339, 341, 356, 384, 386, 423, 424
\Glsentrylongpl 112, 113, 127, 190, 200
\glsentrylongpl 112, 113, 127, 190, 200, 385, 386
\Glsentrylongplural 181
\glsentrylongplural 181
\Glsentryname 184, 197, 210–213, 453
\glsentryname 163, 183, 184, 197, 217, 380, 453, 511–517, 533
\glsentrynumberlist 141, 149, 515–517
\glsentrypdfsymbol 220
\Glsentryplural 185, 197
\glsentryplural 185, 197, 381, 382
\glsentryprevcount 122, 124, 130
\glsentryprevmaxcount 130
\glsentryprevtotalcount 130
\Glsentryshort 111, 112, 189, 199
\glsentryshort 111, 112, 138, 189, 199, 333, 335, 337, 338, 350, 378, 379, 386, 423, 424
\Glsentryshortpl 112, 189, 199
\glsentryshortpl 111–113, 189, 190, 199, 379, 386
\Glsentrysymbol 187, 198
\glsentrysymbol . 186, 187, 198, 221, 514–516
\Glsentrysymbolplural 187, 198
\glsentrysymbolplural 187, 198
\Glsentrytext 184, 197
\glsentrytext 109, 184, 197, 380, 381, 423, 424
\glsentrytitlecase 453
\glsentrytype 164, 233
\Glsentryuseri 92
\glsentryuseri 92
\Glsentryuserii 92
\glsentryuserii 92
\Glsentryuseriii 92
\glsentryuseriii 93
\Glsentryuseriv 93
\glsentryuseriv 93
\Glsentryuserv 93
\glsentryuserv 93
\Glsentryuservi 93
\glsentryuservi 94
\glsextrapostnamehook 213
\glsfieldfetch 108
\glsfieldxdef 207
\glsFindWidestLevelTwo 447
\glsFindWidestTopLevelName 447
\glsfindwidesttoplevelname 510
\GLSfirst 373
\Glsfirst 373
\glsfirst 373
\glsfirstabbrvdefaultfont 235, 254, 256, 259, 262, 264, 267, 270, 347
\glsfirstabbrvemfont 306–330
\glsfirstabbrvfont .. 136, 234, 253–270, 273, 275, 277, 279, 281, 282, 284, 287, 290, 292, 294, 295, 298, 299, 301, 304, 307, 309, 311, 313, 314, 316, 318, 320, 322, 323, 326, 329, 332, 334, 336, 339, 342, 345, 347, 348, 351, 354, 357, 360, 362
\glsfirstabbrvhyphenfont 344–346, 350, 351, 353–358
\glsfirstabbrvonlyfont 360, 361
\glsfirstabbrvscfont 273–288
\glsfirstabbrvsconlyfont 362, 363
\glsfirstabbrvscuserfont ... 335, 336, 338
\glsfirstabbrvsmfont 290–305
\glsfirstabbrvuserfont 332–334, 337, 339–343
\glsfirstaccessdisplay 185, 186

\glsfirstlongdefaultfont 254, 256, 265, 267, 270, 273–283, 290–300, 306–308, 310–312, 314–319, 322, 323
\glsfirstlongemfont 308–310, 312–314, 320, 321, 323–325
\glsfirstlongfont 234, 253–257, 259, 262, 265, 267–273, 275, 277, 279, 281, 282, 284, 287, 290, 292, 294, 295, 298, 299, 301, 304, 307, 309, 311, 313, 314, 316, 318, 320, 322, 324, 326, 329, 332, 334, 336, 339, 342, 345, 348, 351, 354, 357, 360, 362
\glsfirstlongfootnotefont 258–263, 284–289, 301–305, 325–330
\glsfirstlonghyphenfont 344–357
\glsfirstlongonlyfont 359–364
\glsfirstlonguserfont 332–343
\GLSfirstplural 374
\Glsfirstplural 374
\glsfirstplural 374
\glsfirstpluralaccessdisplay 186
\GLSfmtname 60
\Glsfmtname 59
\glsfmtname 58–60
\GLSfmttext 60
\Glsfmttext 59
\glsfmttext 58–60
\glsforeachincategory 250
\glsgenentryfmt 77
\glsgetattribute 84, 108, 124, 128–130, 153, 177, 182, 208, 209, 211–214, 216, 221, 421–424
\glsgetcategoryattribute 202
\glsgetgrouptitle 489, 491, 492
\glsgetwidestname 508
\glsgroupheading 172, 490–499, 501, 502, 504–507, 518–525, 531, 539, 540, 542, 543, 545, 546, 548, 549, 551, 553, 554, 556, 557, 561
\glsgroupskip 172, 490, 493–500, 502, 505, 506, 519, 531, 539, 541–545, 547, 548, 550, 551, 553, 554, 556, 558, 562
\glshasattribute 83, 108, 123, 124, 128, 129, 131, 132, 153, 177, 182, 208, 209, 211, 212, 214, 216, 221, 253, 255–257, 259, 261–263, 266, 268, 269, 271–276, 284, 286, 287, 289–293, 301, 303–305, 307–310, 312, 314, 321, 325–328, 330, 332, 333, 335, 337–339, 341–343, 345–347, 349, 351, 353–356, 358, 359, 361, 362, 364, 421–424
\glshascategoryattribute 203, 409
\glshex 456–462, 464–470, 472–475, 477–485
\glshyperlink 109, 440
\glshypernavsep 150
\glshypernumber 153, 216, 442, 443
\glsifattribute 80, 81, 87, 102, 104, 115, 182, 205, 208–211, 215, 223, 226, 227, 368–378
\glsifcategory 205
\glsifcategoryattribute 101, 102, 192, 194, 195, 203, 204, 231–233
\glsifnotregular 86
\glsifnotregularcategory 204
\glsifplural 79, 86, 88–92, 94–98, 226, 227, 236–248, 418–420
\glsifregular 77, 86, 127, 181
\glsifregularcategory 204
\glsifusetranslator 54
\glsignore 75, 76
\glsinlinedescformat 500
\glsinlinesubdescformat 500
\glsinsert 79, 86, 94–98, 236–249, 344, 351, 353, 356, 358
\glskeylisttok 134, 231, 233
\glslabel 7, 8, 36, 37, 56, 77, 80–84, 101–103, 108, 109, 137, 177, 178, 182, 224–226, 247–249, 261, 263, 287, 289, 303, 305, 328, 330, 333, 335, 337–339, 341, 351, 353, 356, 358, 420
\glslabeltok 134, 231, 233, 253, 255–264, 266, 268, 269, 271–277, 279, 281, 284, 286, 287, 289–295, 297, 301, 303–314, 316, 318, 320, 321, 323, 325–328, 330, 332, 333, 335, 337–343, 345–347, 349, 351, 353–356, 358, 359, 361, 362, 364
\glsletentryfield 217
\glslink 135, 450
\glslink options
  counter 9
  format 215
  hyper 364
  hyperoutside 80, 81
  noindex 8, 102, 364
  textformat 83
  theHvalue 83
  thevalue 83, 174
  wrgloss 7, 80

\glslinkcheckfirsthyperhook 102
\glslinkpostsetkeys 9, 83, 178
\glslinkpresetkeys 9, 83, 178
\glslinkvar 105
\glslinkwrcontent 84, 178
\glslistchildpostlocation 490
\glslistchildprelocation 490, 491
\glslistdesc 490, 491
\glslistdottedwidth 488
\glslistexpandedname 489
\glslistgroupafterheader 491, 492
\glslistgroupheaderfmt 489, 491, 492
\glslistgroupheaderitem 491, 492
\glslistgroupskip 490
\glslistinit 490
\glslistitem 490
\glslistnavigationitem 491, 492
\glslistprelocation 489–491
\glslocalreset 403
\glslocalunset 102, 402, 403
\glslongaccessdisplay 190
\glslongdefaultfont 235, 254, 256, 258, 265, 267, 270, 273, 275, 277, 279, 281–283, 290, 292, 294, 295, 297–299, 307, 311, 314, 316, 318, 322, 331, 345, 359
\glslongemfont 306, 309, 312, 313, 320, 323, 324
\glslongextraDescAlign 538–549, 551, 552, 554–556
\glslongextraDescFmt 535, 539, 540, 542, 543, 545, 546, 548, 550, 551, 553, 554, 556
\glslongextraDescNameHeader 542
\glslongextraDescNameTabularFooter 541
\glslongextraDescNameTabularHeader 541, 542
\glslongextraDescSymNameHeader 554
\glslongextraDescSymNameTabularFooter 553, 554
\glslongextraDescSymNameTabularHeader 553, 554
\glslongextraGroupHeading 539, 540, 542, 543, 545, 546, 548, 549, 551, 553, 554, 556
\glslongextraHeaderFmt . 536, 537, 539, 541, 542, 544, 545, 547–550, 552, 553, 555
\glslongextraLocationAlign 540, 543, 546, 549, 552, 555, 556
\glslongextraLocationDescNameHeader 543
\glslongextraLocationDescNameTabularFooter 542, 543
\glslongextraLocationDescNameTabularHeader 542, 543
\glslongextraLocationDescSymNameHeader 556
\glslongextraLocationDescSymNameTabularFooter 555
\glslongextraLocationDescSymNameTabularHeader 555
\glslongextraLocationFmt 540, 543, 546, 550, 553, 556
\glslongextraLocationSymDescNameHeader 553
\glslongextraLocationSymDescNameTabularFooter 551, 552
\glslongextraLocationSymDescNameTabularHeader 551, 552
\glslongextraLocSetDescWidth .. 540, 543
\glslongextraNameAlign 538–549, 551, 552, 554–556
\glslongextraNameDescHeader 539
\glslongextraNameDescLocationHeader 540
\glslongextraNameDescLocationTabularFooter 539, 540
\glslongextraNameDescLocationTabularHeader 539, 540
\glslongextraNameDescSymHeader 545
\glslongextraNameDescSymLocationHeader 546
\glslongextraNameDescSymLocationTabularFooter 545, 546
\glslongextraNameDescSymLocationTabularHeader 545, 546
\glslongextraNameDescSymTabularFooter 544
\glslongextraNameDescSymTabularHeader 544
\glslongextraNameDescTabularFooter 536, 539
\glslongextraNameDescTabularHeader 536, 539
\glslongextraNameFmt ... 539, 540, 542, 543, 545, 546, 548, 550, 551, 553, 554, 556
\glslongextraNameSymDescHeader 548
\glslongextraNameSymDescLocationHeader 549
\glslongextraNameSymDescLocationTabularFooter 548, 549
\glslongextraNameSymDescLocationTabularHeader 548, 549

\glslongextraNameSymDescTabularFooter 547
\glslongextraNameSymDescTabularHeader 547, 548
\glslongextraSetDescWidth 537–539, 541, 542
\glslongextraSubDescFmt 539, 541–543, 545, 547, 548, 550, 551, 553, 554, 556
\glslongextraSubLocationFmt 541, 543, 547, 550, 553, 556
\glslongextraSubNameFmt 539, 541–543, 545, 546, 548, 550, 551, 553, 554, 556
\glslongextraSubSymbolFmt 545, 547, 548, 550, 551, 553, 554, 556
\glslongextraSymbolAlign 544–552, 554–556
\glslongextraSymbolFmt 535, 545, 546, 548, 550, 551, 553, 554, 556
\glslongextraSymDescNameHeader 551
\glslongextraSymDescNameTabularFooter 550, 551
\glslongextraSymDescNameTabularHeader 550, 551
\glslongextraSymLocSetDescWidth 546, 549, 552, 555
\glslongextraSymSetDescWidth 538, 544, 547, 548, 550, 551, 554
\glslongextraTabularVAlign 538, 540, 541, 543, 544, 546, 547, 549, 550, 552, 554, 555
\glslongextraUpdateWidest 445–447
\glslongextraUpdateWidestChild 445–447
\GlsLongExtraUseTabularfalse 538
\glslongfont 112, 235, 241, 242, 245–247, 254, 256, 257, 259, 260, 262, 265–267, 269, 270, 272, 273, 275, 277, 279, 281, 282, 284, 287, 290, 292, 294, 295, 298, 299, 301, 304, 307, 309, 311, 313, 314, 316, 318, 320, 322, 324, 326, 329, 332, 334, 336, 339, 342, 345, 348, 351, 354, 357, 360–362
\glslongfootnotefont 258, 259, 262, 284, 287, 301, 304, 326, 329
\glslonghyphenfont 345, 347–349, 351, 354, 356, 357
\glslongonlyfont 359, 360, 362
\glslongpltok 233, 253, 255–258, 260, 269, 272–276, 281, 284, 286, 290–293, 297, 301, 303, 306, 308, 310, 312, 314, 318, 320, 323, 325, 327, 332, 333, 335, 337, 338, 340–343, 345–347, 349, 350, 352, 354, 355, 359, 361, 362, 364
\glslongpluralaccessdisplay 190
\glslongtok 134, 231, 233, 253–258, 260, 261, 264, 266, 269, 272–277, 280, 281, 284, 286, 290–294, 297, 301, 303, 306–310, 312–314, 318, 320, 323, 325, 327, 328, 332, 333, 335, 337–343, 345–347, 349–352, 354–356, 359, 361, 362, 364
\glslonguserfont 331, 332, 334, 336, 337, 339, 340, 342
\glsmcols 521, 522, 524, 525
\GLSname 370, 371
\Glsname 370, 371
\glsname 370
\glsnameaccessdisplay . . 183, 184, 210, 212
\glsnamefont 210–213, 215, 537
\glsnavhyperlink 150
\glsnavhyperlinkname 106, 107
\glsnavhypertarget 490–492, 502, 505, 507, 520–526
\glsnavigation 490–492, 502, 505, 507, 520–525
\glsnextpages 143, 168
\glsnoidxdisplayloc 148, 442
\glsnoidxdisplayloclisthandler 148
\glsnoidxloclist 149, 173
\glsnoidxnumberlistloophandler 148
\glsnonextpages 143, 168
\glsnonumberlistfalse 74
\glsnonumberlisttrue 75
\glsnopostdotfalse 144
\glsnopostdottrue 144
\glsnumberlistloop 141
\glsnumlistlastsep 147, 438
\glsnumlistsep 147, 438
\glsopenbrace 62, 156–158
\glsorder 139, 162
\glspagelistwidth 493, 495, 497, 499, 536, 538
\glspar 171, 558
\glspatchLToutput 539, 540, 542–544, 546, 548, 549, 551, 552, 554, 555
\glspdffmtfull 386
\glspdffmtfullpl 387
\glspenaltygroupskip . . . 539, 541, 542, 544, 545, 547, 548, 550, 551, 553, 554, 556
\GLSpl 121, 133, 177
\Glspl 72, 121, 133, 177
\glspl 71, 121, 133, 177

\GLSplural 372
\Glsplural 372
\glsplural 372
\glspluralaccessdisplay 185
\glspluralsuffix 161, 231, 235
\glspostdescription ... 18, 19, 144, 223, 488, 489, 492–500, 503–506, 534, 559, 560
\glspostinline 500
\glspostlinkhook 78–80, 94–98, 113, 236–246, 413, 414, 420
\glsprestandardsort 141
\glsreset 403
\glsresetentrylist 151, 166, 168
\glssee 63–65
\glsseefirstitem 62
\glsseeformat 58, 61, 68, 140, 148
\glsseeitem 62
\glsseelastoxfordsep 61
\glsseelastsep 61, 62
\glsseelist 61
\glsseesep 61
\glssetabbrvfmt 77, 86, 111, 112, 208, 209, 211, 212, 214, 220, 221, 236–241, 243, 244, 247
\glssetattribute 253, 255–257, 259, 261–264, 266, 268, 269, 271–277, 279, 281, 284, 286, 287, 289–295, 297, 301, 303–305, 307–312, 314, 316, 318, 320, 321, 323, 325–328, 330, 332, 333, 335, 337–339, 341–343, 345–347, 349, 351, 353–356, 358, 359, 361, 362, 364
\glssetcategoriesattribute 201
\glssetcategoryattribute 122, 133, 137, 177, 183, 196, 197, 202, 203, 205, 206, 222
\glssetnoexpandfield 13
\glssettoctitle 143, 167
\glssetwidest 446
\glsshortaccessdisplay 189
\glsshortaccsupp 196
\glsshortpltok 233, 253, 255–258, 260, 261, 263, 264, 266, 273–278, 284, 286, 288, 290–295, 301, 303, 305, 307–310, 312, 314, 316, 325, 327, 328, 330, 332, 333, 335, 337–343, 345, 346, 351, 352, 354–356, 358, 359, 361, 362, 364
\glsshortpluralaccessdisplay .. 189, 190
\glsshorttok 134, 231–233, 253–258, 260, 261, 263, 264, 266, 271–278, 280, 284, 286, 288, 290–295, 297, 301–303, 305–310, 312–314, 316, 318, 320, 325, 327, 328, 330, 332, 333, 335, 337–343, 345, 346, 349, 350, 352, 354–356, 358, 359, 361, 362, 364
\glsshowtargetfont 29
\glsshowtargetinner 27–29
\glsshowtargetinnersymleft 28, 29
\glsshowtargetinnersymright 28, 29
\glsshowtargetouter 27, 28
\glsshowtargetsymbol 29
\glssubentryitem 164, 488, 490–500, 502, 504, 506, 518, 531, 535
\glssymbolaccessdisplay 186, 187
\glssymbolpluralaccessdisplay 187
\glstarget ... 163, 165, 488–500, 502, 504, 506, 518, 519, 530, 531, 534, 535, 558, 560
\GLStext 371, 372
\Glstext 371
\glstext 371
\glstextaccessdisplay 184
\glstextformat 80, 84
\glstextup 273
\glstopic@postchildren 561, 562
\glstopic@prechildren 557, 561, 562
\glstopic@prevlevel 557, 561, 562
\glstopicAssignSubIndent 558, 562
\glstopicAssignWidest 559
\glstopicCols 562
\glstopicColsEnv 562
\glstopicDesc 558
\glstopicGroupHeading 557, 561
\glstopicInit 557, 561
\glstopicItem 557, 562
\glstopicLoc 558
\glstopicMarker 558
\glstopicMidSkip 558
\glstopicParIndent 557, 562
\glstopicPostSkip 558
\glstopicPreSkip 558
\glstopicSubIndent 559
\glstopicSubItem 558, 562
\glstopicSubItemBox 560
\glstopicsubitemhangindent 559
\glstopicSubItemParIndent 559
\glstopicSubItemSep 560
\glstopicSubLoc 560
\glstopicSubNameFont 560
\glstopicSubPreLocSep 560
\glstopicTitle 558

\glstopicTitleFont 558
\glstopicwidest 559–561
\glstreechilddesc 504
\glstreeChildDescLoc 502, 505
\glstreechildpredesc 503
\glstreechildprelocation 504, 506
\glstreechildsymbol 502, 505
\glstreedefaultnamefmt 501
\glstreedesc 503
\glstreeDescLoc 502, 504, 508
\glstreegroupheaderfmt .....
..... 502, 505, 507, 519–526, 528
\glstreegroupheaderskip .....
..... 502, 503, 505, 507, 519–526
\glstreegroupskip .. 501, 502, 505, 506, 519
\glstreeindent 504, 506, 517–519
\glstreeitem 501, 521, 530, 531
\glstreenamebox 518, 519
\glstreenamefmt .....
..... 500, 502, 504, 506, 508, 511–519
\glstreenavigationfmt 502, 505, 507, 520–525
\glstreeNoDescSymbolPreLocation 503, 504
\glstreenonamechilddesc 506
\glstreenonameChildDescLoc 506
\glstreenonamedesc 506
\glstreenonameDescLoc 506
\glstreenonamesymbol 506
\glstreepredesc 503, 505
\glstreePreHeader .. 502, 505, 507, 519–525
\glstreeprelocation 501, 503, 506
\glstreesubitem 501, 531
\glstreesubsubitem 501, 531
\glstreesymbol 502, 504
\GlstrLetField 45
\glstype 79, 82, 94–98, 178, 236–246
\glsunset 66, 102, 124–126, 402, 403
\glsupdatewidest 445
\glsuserdescription 332, 333, 335, 339, 342
\glswrite 62, 139
\glswriteentry 7, 8
\Glsxtr 72
\glsxtr 72
\glsxtr@@do@wrglossary 7, 9, 12, 15
\glsxtr@addloclistfield 14, 16
\glsxtr@addunused 66
\glsxtr@applyabbrvfmt 246, 247
\glsxtr@applyabbrvstyle 231, 250
\glsxtr@beginbookindex 529
\glsxtr@counterrecord 163
\glsxtr@dblfloat 18
\glsxtr@do@alsoindex@wrglossary 16
\glsxtr@do@autoadd 8, 82, 83
\glsxtr@dooption 4, 18, 19, 26–28, 31
\glsxtr@endbookindex 530
\glsxtr@fields 160
\glsxtr@float 18
\glsxtr@grptitle ... 502, 505, 507, 519–526
\glsxtr@headentry@p 115, 116
\glsxtr@hyperoutsidefalse 81
\glsxtr@hyperoutsidetrue 80, 81
\glsxtr@ifnextpunc 228
\glsxtr@ifpunctoken 228
\glsxtr@inc@linkcount 83, 183
\glsxtr@inc@wrglossaryctr 7, 8, 26, 32
\glsxtr@indexonly@saveentrycounter
..... 15, 16, 32
\glsxtr@keylist 70–72
\glsxtr@label 533
\glsxtr@langtag 161
\glsxtr@linkprefix 160, 161
\glsxtr@loaddialect 437, 486
\glsxtr@locationhypertext 443, 444
\glsxtr@makeglossaries 139
\glsxtr@mgls@applyopts 409
\glsxtr@mgls@checklastelement 413
\glsxtr@mgls@inner 426
\glsxtr@newabbreviation 136, 230
\glsxtr@newmgls 428–436
\glsxtr@next 228
\glsxtr@org@@do@wrglossary 32
\glsxtr@org@dohyperlink 106, 402
\glsxtr@org@getgrouptitle 150
\glsxtr@org@newignoredglossary 52
\glsxtr@orgmakenoidxglossaries 68
\glsxtr@pluralsuffixes 160, 161
\glsxtr@printgloss@checkexists 144
\glsxtr@printgloss@groupstrue 146
\glsxtr@process 166, 169, 170
\glsxtr@provideignoredglossary 54
\glsxtr@punclist 227, 228
\glsxtr@record 10, 11, 161
\glsxtr@record@nameref 11, 161
\glsxtr@record@nr 15
\glsxtr@recordsee 12
\glsxtr@renewcommand 439
\glsxtr@resource 159, 160
\glsxtr@s@newignoredglossary 52
\glsxtr@s@provideignoredglossary ... 54

\glsxtr@saveentrycounter 7, 9, 12, 103
\glsxtr@setaccessdisplay 214
\glsxtr@setbookindexmark 532
\glsxtr@setup@docurrent 413
\glsxtr@setup@record 14–16, 31, 33
\glsxtr@shortcutsval 160, 161
\glsxtr@texencoding 161
\glsxtr@undefaction@nr 6
\glsxtr@undefaction@val 6
\glsxtr@usesee 58
\glsxtr@warnonexistsordo . 6, 14–16, 56, 57
\glsxtr@writefields 159
\glsxtrabbreviationfont 77
\glsxtrabbrvfootnote ... 258–261, 263, 284–287, 289, 301–303, 305, 325–328, 330
\glsxtrabbrvpluralsuffix ... 161, 194, 236, 254, 256, 259, 262, 264, 267, 270, 273, 289, 306, 331, 345, 347, 351, 357, 359
\glsxtrabbrvtype 20, 233
\glsxtrAccSuppAbbrSetFirstLongAttrs 253, 255, 273, 275, 290, 292, 306, 308, 310, 312, 332, 333, 335, 338, 342, 345, 350, 353, 356, 359, 362
\glsxtrAccSuppAbbrSetNameLongAttrs 260, 263, 278, 286, 288, 293, 302, 305, 327, 330
\glsxtrAccSuppAbbrSetNameShortAttrs 271, 297, 318, 319, 321, 349
\glsxtrAccSuppAbbrSetNoLongAttrs 258, 261, 264, 266, 277, 284, 286, 295, 301, 303, 314, 316, 325, 328
\glsxtrAccSuppAbbrSetTextShortAttrs 255, 257, 274, 276, 291, 293, 308, 309, 311, 313, 337, 338, 340, 341, 343, 346, 352, 355, 358, 361, 363
\glsxtractivatenopost 143
\glsxtraddallcrossrefs 65
\glsxtralias 103
\glsxtrAltTreeIndent 508
\glsxtralttreeInit 518, 524, 525
\glsxtrAltTreePar 508
\glsxtrAltTreeSetHangIndent ... 508, 518
\glsxtrAltTreeSetSubHangIndent 519
\glsxtralttreeSubSymbolDescLocation 519
\glsxtralttreeSymbolDescLocation 508, 518
\glsxtrassignactualsetup 193
\glsxtrassignfieldfont 87–94
\glsxtrautoindex 216
\glsxtrautoindexassignsort 217
\glsxtrautoindexentry 217
\glsxtrautoindexesc 217
\glsxtrbibaddress 441
\glsxtrbibauthor 441
\glsxtrbibbooktitle 441
\glsxtrbibchapter 441
\glsxtrbibedition 441
\glsxtrbibhowpublished 441
\glsxtrbibinstitution 441
\glsxtrbibjournal 441
\glsxtrbibmonth 441
\glsxtrbibnote 441
\glsxtrbibnumber 441
\glsxtrbiborganization 441
\glsxtrbibpages 441
\glsxtrbibpublisher 441
\glsxtrbibschool 441
\glsxtrbibseries 441
\glsxtrbibtitle 441
\glsxtrbibtype 441
\glsxtrbibvolume 441
\glsxtrbookindexatendgroup 530
\glsxtrbookindexatsubendgroup 531
\glsxtrbookindexatsubsubendgroup .. 531
\glsxtrbookindexbetween 530
\glsxtrbookindexbookmark 532
\glsxtrbookindexbookmarkprefix 532
\glsxtrbookindexcols 529, 530
\glsxtrbookindexcolspread 529
\glsxtrbookindexfirstmarkfmt 533
\glsxtrbookindexformatheader 532
\glsxtrbookindexgroupskip 532
\glsxtrbookindexlastmarkfmt 533
\glsxtrbookindexlocation 528, 530
\glsxtrbookindexmulticolsenv .. 529, 530
\glsxtrbookindexname 527, 530
\glsxtrbookindexparentchildsep 528–530
\glsxtrbookindexparentsubchildsep 529–531
\glsxtrbookindexprelocation ... 527, 530
\glsxtrbookindexsubbetween 531
\glsxtrbookindexsublocation 531
\glsxtrbookindexsubname 531
\glsxtrbookindexsubprelocation 531
\glsxtrbookindexsubsubbetween 531
\glsxtrbookindexthepage 532, 533
\glsxtrcat 71, 72
\glsxtrchecknohyperfirst 87–89

\glsxtrcombiningdiacriticIIIrules . 457
\glsxtrcombiningdiacriticIIrules .. 457
\glsxtrcombiningdiacriticIrules ... 457
\glsxtrcombiningdiacriticIVrules .. 457
\glsxtrComputeTreeIndent 518, 519
\glsxtrComputeTreeSubIndent 519
\glsxtrcounterprefix 152
\glsxtrcurrencyrules 460
\glsxtrcurrentgrptitle 532
\glsxtrcurrentmglscsname 426
\GlsXtrDefaultResourceOptions 159
\Glsxtrdefaultsubsequentfmt ... 249, 251
\glsxtrdefaultsubsequentfmt ... 249, 251
\Glsxtrdefaultsubsequentplfmt . 249, 251
\glsxtrdefaultsubsequentplfmt . 249, 251
\GlsXtrDefineAbbreviationShortcuts
..... 22, 23
\GlsXtrDefineAcShortcuts 23
\GlsXtrDefineOtherShortcuts 23
\glsxtrdetoklocation 176
\glsxtrdiscardperiod 225
\glsxtrdisplayendloc 151
\glsxtrdisplayendlochook 152
\glsxtrdisplaysingleloc 151, 152
\glsxtrdisplaystartloc 151
\glsxtrdoautoindexname 104, 105, 213
\glsxtrdohyperlink 106, 109, 110
\glsxtrdopostpunc .....
.... 261, 263, 287, 289, 303, 305, 328, 330
\glsxtrdowrglossaryhook 104, 105
\GlsXtrDualField 440
\glsxtremsuffix 307, 309, 311,
313, 314, 316, 318, 320, 322, 323, 326, 329
\GlsXtrEnableEntryCounting 133
\GlsXtrEnableEntryUnitCounting 121, 122
\GlsXtrEnableOnTheFly 70, 72, 73
\glsxtrendfor 39
\glsxtrfieldlistgadd 162
\glsxtrfieldtitlecase 208–211, 215
\glsxtrfieldtitlecasecs 207
\glsxtrfieldxifinlist 171
\glsxtrfirstscfont 273
\glsxtrfirstsmfont 289
\GlsXtrFmtDefaultOptions 36
\glsxtrfmtdisplay 36
\glsxtrfmtexternalnameref 443, 444
\GlsXtrFmtField 36, 37
\glsxtrfmtinternalnameref 443, 444
\glsxtrfootnotedescname .....
.... 260, 263, 286, 288, 302, 305, 327, 330
\glsxtrfootnotedescsort .....
.... 260, 263, 286, 288, 302, 305, 327, 330
\glsxtrfootnotename .....
.... 258, 261, 284, 286, 301, 303, 325, 328
\GlsXtrForeignTextField 47
\GlsXtrFormatLocationList 74, 77, 515–517
\GLSxtrfull 21, 99, 376–378
\Glsxtrfull 20, 21, 99, 377
\glsxtrfull 20, 21, 98, 376, 377
\Glsxtrfullformat .....
. 234, 248, 251, 254, 256, 259, 262, 265,
267, 271, 274, 276, 278, 279, 282, 283,
285, 287, 290, 292, 295, 296, 299–301,
304, 307, 309, 311, 313, 315, 317, 319,
321, 323, 325, 326, 329, 333, 334, 336,
339, 343, 346, 348, 352, 354, 357, 360, 363
\glsxtrfullformat 234,
248, 249, 251, 254, 256, 259, 262, 265,
267, 270, 274, 275, 278, 279, 282–284,
287, 290, 292, 294, 296, 298, 300, 301,
304, 307, 309, 311, 313, 315, 316, 319,
321, 323, 324, 326, 329, 332, 334, 336,
339, 342, 346, 348, 351, 354, 357, 360, 362
\GLSxtrfullpl 21, 99, 377, 378
\Glsxtrfullpl 20, 21, 99, 378
\glsxtrfullpl 20, 21, 99, 377
\Glsxtrfullplformat 235,
248, 251, 254, 256, 259, 262, 265, 267,
271, 274, 276, 278, 279, 282, 283, 285,
287, 291, 293, 295, 296, 299, 300, 302,
304, 307, 309, 311, 313, 315, 317, 319,
321, 323, 325, 326, 329, 333, 334, 336,
339, 343, 346, 348, 352, 354, 357, 360, 363
\glsxtrfullplformat .....
.... 248, 251, 254, 256, 259, 262, 265,
267, 271, 274, 276, 278, 279, 282, 283,
285, 287, 290, 292, 295, 296, 299–301,
304, 307, 309, 311, 313, 315, 317, 319,
321, 323, 324, 326, 329, 332, 334, 336,
339, 342, 346, 348, 351, 354, 357, 360, 362
\glsxtrfullsep 234,
253–257, 259, 260, 262, 263, 265–270,
273–283, 285, 288, 290–298, 300, 302,
304–317, 319–324, 326, 327, 329, 331,
344–346, 348, 350, 353–356, 360, 361, 363
\glsxtrgenabbrvfmt 77
\glsxtrgeneralpuncIIrules 460

\glsxtrgeneralpuncIrules 460
\glsxtrgetgrouptitle .....
..... 150, 502, 505, 507, 519–525, 532
\glsxtrgroupfield 172
\Glsxtrheadfirst 367
\glsxtrheadfirst 367
\Glsxtrheadfirstplural 367
\glsxtrheadfirstplural 367
\Glsxtrheadfull 367
\glsxtrheadfull 367
\Glsxtrheadfullpl 367
\glsxtrheadfullpl 367
\Glsxtrheadlong 367
\glsxtrheadlong 367
\Glsxtrheadlongpl 367
\glsxtrheadlongpl 367
\Glsxtrheadname 367
\glsxtrheadname 163, 367
\Glsxtrheadplural 367
\glsxtrheadplural 367
\Glsxtrheadshort 367
\glsxtrheadshort 367
\Glsxtrheadshortpl 367
\glsxtrheadshortpl 367
\Glsxtrheadtext 367
\glsxtrheadtext 367
\glsxtrhiernamesep 59, 60
\glsxtrhyperlink 25, 26, 108
\glsxtrhyphensuffix 345, 354
\glsxtridentifyglslike 174
\glsxtrifcounttrigger 124–126
\glsxtrifcustomdiscardperiod 225
\glsxtrifemptyglossary . 151, 157, 166, 169
\GlsXtrIfFieldEqNum 164
\glsxtrifhasfield .. 47, 59, 60, 433, 440, 527
\glsxtrifhyphenstart .....
..... 344, 347, 349, 350, 353, 355, 356
\glsxtrifindexing 104
\glsxtrifinmark 85, 115–118, 366, 367
\glsxtrifmulti 62, 66, 108, 397, 398, 427
\glsxtrifnextpunc 228
\glsxtrifperiod 225, 227
\glsxtrifrecordtrigger 178–180
\GlsXtrIfUnusedOrUndefined 104, 415
\glsxtrifwasfirstuse ... 86–89, 94–98,
101, 137, 226, 236, 239–246, 261–264,
287–289, 303–306, 328, 330, 333, 335,
337–340, 351, 352, 356, 358, 417, 418, 420
\glsxtrinclinkcounter 182
\glsxtrindexaliased 103, 104
\glsxtrindexseealso 64, 65
\glsxtrinithyperoutside 83
\glsxtrinitwrgloss 83, 178
\glsxtrinitwrglossbeforefalse 80
\glsxtrinitwrglossbeforetrue 80
\Glsxtrinlinefullformat .....
..... 235, 237, 251, 260,
263, 265, 267, 268, 270, 277, 279–281,
283, 285, 288, 294, 296–298, 300, 302,
305, 315–317, 319, 321, 322, 324, 327,
329, 334, 336, 340, 348, 352, 358, 360, 363
\glsxtrinlinefullformat .....
..... 235–237, 251, 259, 262,
265, 267, 268, 270, 277, 279–281, 283,
285, 288, 294, 295, 297, 298, 300, 302,
304, 314, 316–318, 320, 322, 324, 326,
329, 334, 336, 339, 348, 352, 357, 360, 363
\Glsxtrinlinefullplformat .....
..... 235, 238, 251, 260,
263, 265, 267, 268, 270, 278–280, 282,
283, 285, 288, 294, 296–298, 300, 302,
305, 315–317, 319, 321, 323, 324, 327,
329, 334, 336, 340, 348, 352, 358, 360, 363
\glsxtrinlinefullplformat .....
..... 234, 235, 238, 239, 251, 259,
262, 265, 267, 268, 270, 277, 279–281,
283, 285, 288, 294, 296–298, 300, 302,
304, 315–317, 319, 320, 322, 324, 327,
329, 334, 336, 340, 348, 352, 357, 360, 363
\glsxtrinsertinsidefalse 253
\GlsXtrInternalLocationHyperlink 26, 153
\glsxtrLatinA 462–467
\glsxtrLatinAELigature 464, 466, 467
\glsxtrLatinE 462–467
\glsxtrLatinEszettSs 463–465, 467
\glsxtrLatinEszettSz 463, 466
\glsxtrLatinEth 462–466
\glsxtrLatinH 462–467
\glsxtrLatinI 462–467
\glsxtrLatinInsularG 466
\glsxtrLatinK 462–467
\glsxtrLatinL 462–467
\glsxtrLatinM 462–467
\glsxtrLatinN 462–467
\glsxtrLatinO 462–467
\glsxtrLatinOELigature 464, 466, 467
\glsxtrLatinP 462–467
\glsxtrLatinS 462–467

\glsxtrLatinT 462–467
\glsxtrLatinThorn 466
\glsxtrLatinX 462–467
\GlsXtrLocationField 172
\glsxtrlocationhyperlink 153
\glsxtrlocrangefmt 152
\GLSxtrlong 21, 97, 374–376
\Glsxtrlong 20, 21, 97, 375, 376
\glsxtrlong 20, 21, 96, 375
\glsxtrlonghyphen 351, 352
\glsxtrlonghyphennoshort 348
\glsxtrlonghyphenshort 346
\glsxtrlongnoshortdescname . 269, 323, 347
\glsxtrlongnoshortname .....
..... 272, 280, 297, 318, 320, 349
\GLSxtrlongpl 21, 98, 375, 376
\Glsxtrlongpl 20, 21, 98, 376
\glsxtrlongpl 20, 21, 97, 375
\glsxtrlongshortdescname .....
..... 255, 274, 291, 308, 310, 346, 352
\glsxtrlongshortdescsort .....
.... 255, 274, 291, 308, 310, 341, 346, 352
\glsxtrlongshortname .....
253, 273, 290, 306, 308, 332, 333, 345, 350
\glsxtrlongshortscuserdescname 338
\glsxtrlongshortscusername 335
\glsxtrlongshortuserdescname .. 337, 341
\glsxtrmarkhook 365
\glsxtrMathItalicAlpha . 471, 472, 475, 476
\glsxtrMathItalicBeta .. 471, 472, 475, 476
\glsxtrMathItalicChi 472, 476, 477
\glsxtrMathItalicDelta . 471, 472, 475, 476
\glsxtrMathItalicEpsilon 471, 472, 475, 476
\glsxtrMathItalicEta ... 471, 472, 475, 476
\glsxtrMathItalicGamma . 471, 472, 475, 476
\glsxtrMathItalicIota .. 471, 472, 475, 476
\glsxtrMathItalicKappa . 471, 472, 475, 476
\glsxtrMathItalicLambda 471, 472, 475, 476
\glsxtrMathItalicMu 471, 472, 475, 476
\glsxtrMathItalicNu 471, 472, 475, 477
\glsxtrMathItalicOmega 472, 476, 477
\glsxtrMathItalicOmicron 471, 472, 476, 477
\glsxtrMathItalicPhi 472, 476, 477
\glsxtrMathItalicPi 471, 472, 476, 477
\glsxtrMathItalicPsi 472, 476, 477
\glsxtrMathItalicRho ... 471, 472, 476, 477
\glsxtrMathItalicSigma 472, 476, 477
\glsxtrMathItalicTau 472, 476, 477
\glsxtrMathItalicTheta . 471, 472, 475, 476
\glsxtrMathItalicUpsilon ... 472, 476, 477
\glsxtrMathItalicXi 471, 472, 476, 477
\glsxtrMathItalicZeta .. 471, 472, 475, 476
\glsxtrmglsWarnAllSkipped 410, 411
\glsxtrmglswrite 419, 425
\GLSxtrmultientryadjustednamefmt .. 452
\GlsXtrmultientryadjustednamefmt .. 451
\Glsxtrmultientryadjustednamefmt .. 451
\glsxtrmultientryadjustednamefmt .. 451
\GLSxtrmultientryadjustednameother 452
\GlsXtrmultientryadjustednameother 451
\Glsxtrmultientryadjustednameother 451
\glsxtrmultientryadjustednameother 451
\glsxtrmultientryadjustednamepostsep
..... 451, 452
\glsxtrmultientryadjustednamepresep
..... 451, 452
\glsxtrmultientryadjustednamesep ..
..... 451–453
\glsxtrmultilastotherindex 406
\glsxtrmultimainindex 406
\glsxtrmultisupplocation 442
\glsxtrnamereflink 442
\glsxtrnewabbrevpresetkeyhook 232
\glsxtrnewnumber 22
\glsxtrnewsymbol 22
\glsxtrNoGlossaryWarning 15, 24, 154
\GlsXtrNoGlsWarningAutoMake 157
\GlsXtrNoGlsWarningBuildInfo 158
\GlsXtrNoGlsWarningCheckFile 157
\GlsXtrNoGlsWarningEmptyMain .. 157, 158
\GlsXtrNoGlsWarningEmptyNotMain ... 157
\GlsXtrNoGlsWarningEmptyStart 157
\GlsXtrNoGlsWarningHead 157
\GlsXtrNoGlsWarningMisMatch 158
\GlsXtrNoGlsWarningNoOut 158
\GlsXtrNoGlsWarningTail 158
\glsxtrnopostpunc 144
\glsxtronlydescname 361, 363
\glsxtronlydescsort 361, 363
\glsxtronlyname 359
\glsxtronlysuffix 360
\glsxtrorg@ifKV@glslink@hyper 78
\glsxtrorglong 193, 231, 254, 347
\glsxtrorgshort 193, 231, 254
\GLSxtrp 115
\Glsxtrp 114
\glsxtrp 114, 116

\glsxtrparen 226, 234, 253–257, 259, 260, 262, 263, 265–270, 273–283, 285, 288, 290–294, 296–298, 300, 302, 304–317, 319–324, 326, 327, 329, 331, 344–346, 348, 350, 353–356, 360, 361, 363
\glsxtrpdfentryfmt 37
\Glsxtrpl 72
\glsxtrpl 72
\glsxtrpostdescription . 144, 205, 224, 500
\glsxtrposthyphenlong 356, 358
\glsxtrposthyphenshort 351, 353
\glsxtrposthyphensubsequent 351, 353, 356, 358
\glsxtrpostlink 225
\glsxtrpostlinkendsentence 225
\glsxtrpostlinkhook 224
\glsxtrpostlocalreset 120, 123, 131
\glsxtrpostlocalunset 120, 123, 131
\glsxtrpostlongdescription 52
\glsxtrpostnamehook 211–213, 215
\GlsXtrPostNewAbbreviation 233, 251, 253, 255–258, 261, 263, 264, 266, 268, 269, 271–277, 279, 281, 284, 286–288, 290–295, 297, 301, 303, 305, 307–310, 312, 314, 316, 318, 320, 321, 323, 325–328, 330, 332, 333, 335, 337–343, 345–347, 349, 351, 352, 354–356, 358, 359, 361, 362, 364
\glsxtrpostreset 120, 123, 131
\glsxtrpostunset 119, 122, 130, 131
\glsxtrprelocation 489, 492, 494, 496, 498, 501, 527
\glsxtrprotectlinks 108, 109
\glsxtrprovideaccsuppcmd 196, 197
\GlsXtrRecordCounter 12
\glsxtrrecordtriggervalue 177
\GlsXtrRecordWarning 158
\glsxtrregularfont 77, 86
\glsxtrresourcecount 160
\glsxtrresourcefile 160
\glsxtrresourceinit 159
\glsxtrrestoremarkhook 365
\glsxtrrestorepostpunc 144, 145
\glsxtrscfont 272
\glsxtrsconlydescname 364
\glsxtrsconlydescsort 364
\glsxtrsconlyname 362
\glsxtrsconlysuffix 362
\glsxtrscsuffix 273, 275, 277, 279, 281, 282, 284, 287, 335, 362
\glsxtrscusersuffix 336
\GlsXtrSetActualChar 219
\glsxtrsetaliasnoindex ... 15, 16, 103, 104
\GlsXtrSetEncapChar 220
\GlsXtrSetEscChar 219
\glsxtrsetfieldifexists 45
\glsxtrsetglossarylabel 145
\GlsXtrSetLevelChar 219
\glsxtrsetpopts 113
\glsxtrsetupfulldefs 236–239, 262, 264, 287, 289, 304, 306, 328, 330
\GLSxtrshort 21, 95, 117, 118, 368, 369
\Glsxtrshort 20, 21, 94, 369
\glsxtrshort 20, 21, 94, 368
\glsxtrshortdescname ... 266, 278, 295, 316
\glsxtrshorthyphen 357
\glsxtrshorthyphenlong 354
\glsxtrshortlongdescname 257, 276, 293, 311, 313, 355, 358
\glsxtrshortlongdescsort 257, 276, 293, 311, 313, 343, 355, 358
\glsxtrshortlongname 256, 275, 292, 310, 312, 338, 342, 354, 356
\glsxtrshortlonguserdescname .. 340, 343
\glsxtrshortnolongname . 264, 277, 294, 314
\GLSxtrshortpl 21, 96, 368–370
\Glsxtrshortpl 20, 21, 95, 369, 370
\glsxtrshortpl 20, 21, 95, 368, 369
\glsxtrshowtargetinner 28, 29
\glsxtrshowtargetouter 28, 29
\glsxtrshowtargetsymbolleft 28, 29
\glsxtrshowtargetsymbolright 28, 29
\glsxtrsmfont 289
\glsxtrsmsuffix 290, 292, 294, 295, 298, 299, 301, 304
\GlsXtrStandaloneEntryName 163
\GlsXtrStandaloneEntryOther 165
\GlsXtrStandaloneGlossaryType . 163, 164
\GlsXtrStandaloneSubEntryItem . 163, 164
\Glsxtrsubsequentfmt 248, 251, 270, 281, 283, 298, 299, 318, 320, 322, 324, 348, 351, 357
\glsxtrsubsequentfmt 247, 248, 251, 270, 281, 282, 298, 299, 318, 320, 322, 324, 348, 351, 357

\Glsxtrsubsequentplfmt 247, 251, 270, 281, 283, 298, 299, 318, 320, 322, 324, 348, 351, 357
\glsxtrsubsequentplfmt 247, 251, 270, 281, 282, 298, 299, 318, 320, 322, 324, 348, 351, 357
\glsxtrsupplocationurl 153, 442–444
\glsxtrtagfont 223
\GLSxtrtitlefirst 383
\Glsxtrtitlefirst 366–368, 382, 383
\glsxtrtitlefirst 366–368, 382
\GLSxtrtitlefirstplural 384
\Glsxtrtitlefirstplural 366–368, 383
\glsxtrtitlefirstplural 366–368, 383
\GLSxtrtitlefull 386
\Glsxtrtitlefull 367, 368, 386
\glsxtrtitlefull 366–368, 386
\GLSxtrtitlefullpl 387
\Glsxtrtitlefullpl 367, 368, 387
\glsxtrtitlefullpl 366–368, 387
\GLSxtrtitlelong 384, 385
\Glsxtrtitlelong 366–368, 384
\glsxtrtitlelong 366–368, 384
\GLSxtrtitlelongpl 385
\Glsxtrtitlelongpl 366–368, 385
\glsxtrtitlelongpl 366–368, 385
\GLSxtrtitlename 380
\Glsxtrtitlename 366, 367, 380
\glsxtrtitlename 366, 367, 380
\glsxtrtitleorpdforheading 29, 30, 163, 164, 366, 367
\GLSxtrtitleplural 382
\Glsxtrtitleplural 366, 367, 382
\glsxtrtitleplural 366, 367, 381
\Glsxtrtitleshort 366, 367, 379
\glsxtrtitleshort 366, 367, 378
\Glsxtrtitleshortpl 366, 367, 379
\glsxtrtitleshortpl 366, 367, 379
\GLSxtrtitletext 381
\Glsxtrtitletext 366, 367, 381
\glsxtrtitletext 366, 367, 380, 381
\GlsXtrTotalRecordCount 177
\glsxtrtreechildpredesc 504, 508
\glsxtrtreepredesc 503, 508
\glsxtrtreetopindent 508, 517
\glsxtrundefaction 6, 14–16, 34, 53, 55–57, 397, 398, 408
\glsxtrundeftag 33, 148, 149
\GlsXtrUnknownDialectWarning 49
\glsxtrunsrtdo 170, 171
\glsxtrUpAlpha 470, 471, 475, 476
\glsxtrUpBeta 470, 471, 475, 476
\glsxtrUpChi 470, 471, 476, 477
\glsxtrUpDelta 470, 471, 475, 476
\glsxtrUpDigamma 470, 471, 475
\glsxtrUpEpsilon 470, 471, 475, 476
\glsxtrUpEta 470, 471, 475, 476
\glsxtrUpGamma 470, 471, 475, 476
\glsxtrUpIota 470, 471, 475, 476
\glsxtrUpKappa 470, 471, 475, 476
\glsxtrUpLambda 470, 471, 475, 476
\glsxtrUpMu 470, 471, 475, 477
\glsxtrUpNu 470, 471, 475, 477
\glsxtrUpOmega 470, 471, 476, 477
\glsxtrUpOmicron 470, 471, 476, 477
\glsxtrUpPhi 470, 471, 476, 477
\glsxtrUpPi 470, 471, 476, 477
\glsxtrUpPsi 470, 471, 476, 477
\glsxtrUpRho 470, 471, 476, 477
\glsxtrUpSigma 470, 471, 476, 477
\glsxtrUpTau 470, 471, 476, 477
\glsxtrUpTheta 470, 471, 475, 476
\glsxtrUpUpsilon 470, 471, 476, 477
\glsxtrUpXi 470, 471, 476, 477
\glsxtrUpZeta 470, 471, 475, 476
\GlsXtrUseAbbrStyleFmts 255, 257, 261, 264, 266, 268, 269, 271, 272, 275, 277, 280, 286, 289, 291, 293, 296, 303, 306, 308, 310, 312, 314, 317, 322, 325, 328, 330, 337, 338, 341, 343, 346, 347, 349, 353, 355, 358, 361, 364
\GlsXtrUseAbbrStyleSetup 266, 268, 269, 271, 272, 280, 282, 296, 299, 317, 321, 322, 325
\glsxtrusefield 439
\glsxtruserfield 331
\glsxtruserparen 332–343
\glsxtrusersuffix 332, 334, 339, 342
\glsxtruseseealsoformat 60, 62
\glsxtruseseeformat 58, 61
\GlsXtrWarnDeprecatedAbbrStyle 231, 252
\GlsXtrWarning 71, 72
\glsxtrword 230
\glsxtrwordsep 230, 344, 347, 349, 350, 353, 355, 356
\glsxtrwrglossmark 27

H

\hangindent 504, 506, 508, 518–520, 524–526, 557, 559, 562
\hbox 488
\hfill 488
\href 108
\hsize 73, 74
\hss 488
\hyperlink 16, 109, 442
\hyperpage 216
\hyperref 108, 153, 440, 444
hyperref package 109, 216, 364, 378, 442

I

\if 70
\if@display 11
\if@glsxtr@autoseeindex 30, 31, 57, 64
\if@glsxtr@equations 8, 83
\if@glsxtr@floats 18
\if@glsxtr@format@override 216
\if@glsxtrdocdefrestricted 68
\if@glsxtrindexcrossrefs 17, 65
\if@mgls@writeseparaterefs 425
\if@org@KV@glslink@local 102
\ifblank 34, 35, 71, 72, 139, 452
\ifbool 33, 51, 397, 406
\ifcase ... 6, 15, 22, 24, 27, 28, 69, 80, 145, 403, 404, 411–413, 415, 416, 419, 502, 530
\ifcsdef 34, 44, 50, 53–55, 80, 84, 100, 107, 114–118, 128, 143, 150, 151, 164, 171, 174, 176, 182, 208, 209, 211–214, 221, 225, 231, 246, 250, 391–393, 420, 421, 442, 492–499, 560
\ifcsempty 400
\ifcsname 490
\ifcsstring 34, 203, 250
\ifcsundef 33, 43, 48, 50, 53, 55, 68, 73, 75, 109, 122, 128–132, 149, 150, 154, 171, 182, 183, 196, 203, 249, 251, 252, 395, 399, 400, 408, 488, 509, 510, 518, 532, 560
\ifcsvoid 65, 103, 202
\ifdef . 15, 22, 30, 33, 36, 43, 44, 47, 49, 51, 56, 57, 62, 63, 67, 73, 74, 102, 106, 107, 115–117, 139, 142, 143, 147, 148, 159, 161, 165, 174, 192, 205–207, 219, 220, 223, 331, 366, 378–387, 425, 426, 434, 440, 445–447, 453, 485, 486, 488–492, 500–503, 505, 507, 519–525, 528, 532, 533
\ifdefempty 6–8, 41, 43, 48, 49, 53–55, 58, 60, 61, 83, 85, 122, 133, 134, 136, 139, 142, 152, 159, 166, 169, 172, 177, 178, 193–195, 222, 230, 247, 396, 420, 421, 427, 448, 529
\ifdefequal 46, 47, 68, 106, 134, 136, 158, 172, 174, 214, 402
\ifdefstring 5, 14, 26, 46, 138, 162, 216, 222, 528
\ifdefvoid 57, 64–66, 108, 128, 149, 153, 173, 394, 395, 407, 409, 444
\ifdim . 73, 74, 138, 423, 424, 509–517, 537, 561
\IfFileExists 24, 154, 157, 159, 161, 486, 487
\ifglossaryexists 57, 168, 438
\ifglsacronym 20, 157, 438
\ifglsacrshortcuts 22
\ifglsautomake 142, 157, 161
\ifglsentrycounter 49, 50
\ifglsentryexists 8, 33, 56, 70–72, 74, 75, 86, 172, 203, 224, 449
\ifglsfieldeq 200
\ifglshasdesc ... 503, 504, 557, 558, 560, 562
\ifglshasfield 36, 37, 331
\ifglshaslong 127, 181, 430
\ifglshasparent 163, 164, 166, 169, 510, 512, 513
\ifglshasshort 58–60, 77, 86, 430
\ifglshassymbol 226, 432, 503, 504, 506, 508, 558, 560
\ifglsindexonlyfirst 104
\ifGlsLongExtraUseTabular 538, 540, 541, 543, 544, 546, 547, 549, 550, 552, 553, 555
\ifglsnogroupskip 490, 493–500, 502, 505, 506, 519, 528, 539, 541–543, 545, 547, 548, 550, 551, 553, 554, 556
\ifglsnonumberlist 77
\ifglsnopostdot 18, 19, 144
\ifglssanitizesort 141
\ifglssubentrycounter 49, 50
\ifglsused 66, 101, 123, 132, 137, 247, 510–512, 514–516
\ifglsxindy 154–156
\ifglsxtr@hyperoutside 84
\ifglsxtr@printgloss@groups ... 166, 169
\ifglsxtrinitwrglossbefore ... 80, 84, 178
\ifglsxtrinsertinside 239–246, 249, 254, 256, 257, 259, 260, 262, 263, 265, 267–271, 274–285, 287, 288, 290–302, 304, 305, 307, 309, 311, 313–327, 329, 332–334, 336, 339, 340, 342–344, 347, 350, 352, 353, 356–358, 360, 362, 363

\ifhaslong 423, 424
\ifhasshort 422–424
\ifHy@hyperindex 216
\ifinlist 119
\ifinlistcs 38, 68
\ifinner 29
\ifKV@glslink@hyper 78, 82, 84, 85
\ifKV@glslink@local 102
\ifKV@glslink@noindex 7, 8, 12, 36, 103, 104
\ifKV@glsxtrcombined@firstskipmain 405, 410
\ifKV@glsxtrcombined@firstskipothers 405, 410
\ifKV@glsxtrcombined@usedskipmain 405, 410
\ifKV@glsxtrcombined@usedskipothers 405, 410
\ifKV@mgls@presetlocal 403
\ifmglsused 408, 409
\ifmmode 11, 29, 30
\ifmultiglossaryentryglobal 394–396, 399
\ifnum 16, 17, 43, 67, 124, 132, 133, 149, 160, 172, 177, 392, 393, 395, 396, 399, 401, 406, 410, 445–447, 504, 506, 518, 519, 529, 530, 557, 561, 562
\ifst@rred 11
\ifstrempty 164, 174, 183, 409, 442, 443, 445, 446, 448
\ifstrequal 19, 24, 82, 106, 442
\ifthenelse 157, 158, 225
\IfTrackedDialectHasMapping 48
\IfTrackedLanguageFileExists 437
\ifundef 16, 26, 41, 43, 48, 139, 222, 223, 393, 442, 485, 537
\ifx 7, 9–11, 14, 32, 61, 63, 73, 74, 83, 136, 139, 143, 146, 151–153, 159, 161, 162, 167, 168, 216–218, 220, 228–230, 232, 344, 394, 399, 402, 404, 405, 407, 409, 411, 413–415, 417, 418, 425, 426, 444, 452, 526
\immediate 67, 123, 124, 132, 154, 161, 425, 426
\index 217
\indexspace 489, 501, 528, 532
\input 437
\inputencodingname 161
\InputIfFileExists 67
\istfilename 139
\item 156, 488, 489, 491, 501, 502, 520, 521

J

\jobname 67, 154–158, 160, 161

K

\key@ifundefined 12, 13, 34, 35, 99, 166, 169, 172
\KV@glslink@hyperfalse 87, 101, 102, 109, 110
\KV@glslink@hypertrue 109
\KV@glslink@noindexfalse 86, 102, 103
\KV@glslink@noindextrue 103, 110
\KV@glsxtrcombined@firstskipmainfalse 390
\KV@glsxtrcombined@firstskipothersfalse 390
\KV@glsxtrcombined@usedskipmainfalse 391
\KV@glsxtrcombined@usedskipothersfalse 391
\KV@mgls@presetlocalfalse 401

L

\L 467, 470
\l 470
\label 26, 146
\large 559
\LaTeX 155, 156
\leaders 488
\leavevmode 52, 82
\let 4, 6–10, 12, 14–16, 18, 20–22, 27, 28, 30–32, 36, 39, 47, 48, 52, 61, 67–69, 73, 75, 76, 78, 79, 82–98, 100–106, 109, 110, 113, 119–123, 130–140, 142–150, 159, 161, 162, 165–170, 172, 174, 177, 178, 183, 192, 194, 208–218, 221–223, 228, 230–232, 236–246, 249, 251, 262, 264, 287, 289, 304, 306, 328, 330, 365–368, 390, 392–397, 399, 401, 402, 405–415, 417–420, 425, 438, 440, 443, 451–453, 485, 486, 489, 490, 501, 508, 510, 521, 529–532
\letabbreviationstyle 260, 261, 263, 264, 266, 268, 271, 272, 278, 280, 295, 296, 315, 317
\letcs 34, 41, 43, 58, 60, 61, 66, 80, 84, 99, 107, 108, 147–150, 171, 172, 208–215, 221, 399, 409, 452, 533
\levelchar 219
\linewidth 537
\listadd 128
\listbreak 222

\listcsadd 37
\listcseadd 37, 129
\listcsgadd 37, 68
\listcsxadd 37, 129
\listxadd 119, 393
\loadglsentries 69, 155
\long 52

M

\m@ne 439
\MakeAcronymsAbbreviations 137
\makeatletter 67, 154, 159, 218
\makeatother 218
\makebox 488, 518, 519, 561
\makefirstuc 223, 453
makeglossaries 147
\makeglossaries 138, 154, 156–158, 162
makeindex 15, 138
\makenoidxglossaries 156
\MakeTextUppercase 367
\MakeUppercase 366, 367
\marginpar 29
\markboth 365
\markright 365
\mathit 455
\mathrm 454–456
\maxdimen 73, 74
\mbox 490, 491, 518
\medskip 157, 158, 171, 560
\MessageBreak 69, 72, 124, 133, 139, 142, 143, 158, 167, 250
mfirstuc package 222
\mfirstucMakeUppercase 44, 87–98, 101, 111–113, 115, 117, 118, 126, 127, 135, 181, 184–190, 197–200, 210, 211, 215, 237, 239, 241, 242, 244, 246–249, 453
\mfu@checkword@arg 222, 223
\mfu@checkword@do 223
\mgls 425, 426
\mgls@disable@mglsopts 407
\mgls@disable@setup 409
\mgls@do@postlinkhook .. 408, 411, 412, 419
\mgls@enable@mglsopts 407
\mgls@enable@setup 409, 410
\mgls@org@postlinkhook 413, 414
\mglscurrentcategory ... 410, 411, 420, 421
\mglscurrentlabel 405, 413–415, 417, 418, 452, 453
\mglscurrentlist 403–405, 409, 413
\mglscurrentmainlabel 403–405, 409, 410, 414, 415, 417, 452
\mglscurrentmainlist 452
\mglscurrentmainoptions 452
\mglscurrentmultilabel . 408, 409, 411, 419
\mglscurrentoptions 409
\mglscurrentprefix 411, 420
\mglscurrentsuffix 411, 421
\mglscustompostlinkhook 412
\mglselementindex 394, 395, 399, 400, 406, 413
\mglselementposthook 418
\mglselementprehook 415
\mglselementreset 403, 404
\mglselementunset 404, 405
\mglsfield 433
\mglsforotherelements 402
\mglshascategoryprefix 420
\mglshascategorysuffix 421
\mglsiflast 406, 413, 414, 417
\mglsiflastelementcapscase 408, 411, 419, 420
\mglsiflastelementskipped .. 408, 411, 418
\mglsiflastelementwasfirstuse 408, 411, 419, 420
\mglsiflastelementwasplural 408, 411, 419, 420
\mglsiflastmaincapscase 408, 418, 420
\mglsiflastmainskipped 408, 418
\mglsiflastmainwasfirstuse . 408, 418, 420
\mglsiflastmainwasplural ... 408, 418, 420
\mglsisfirstuse . 405, 408–411, 413, 415, 416
\mglslastcategory 408, 411
\mglslastelementlabel 408, 411, 413, 414, 420
\mglslastelementpostlinkhook .. 411, 412
\mglslastmainlabel 408, 417, 418, 420
\mglslastmainpostlinkhook 411, 412
\mglslastmultilabel 408, 411
\mglslocalunset 419
\mglsname 62
\mglsprefix 413
\mglsreset 399
\mglsseefirstitem 62
\mglsseeitem 62
\mglssuffix 419
\mglsunset 398, 419
\mglsusecategoryprefix 420
\mglsusecategorysuffix 421
\mglswasfirstuse 408, 411, 412
\mglsWriteSeparateRefsFalse 426

\mglsWriteSeparateRefsTrue 425
\midrule 536, 539, 541, 542, 544, 545, 547, 549, 550, 552, 553, 555
\mpgls 434
\mpglsWarning 434
\multiglossaryentry 397, 398, 408
\multiglossaryentryglobalfalse 392
\multiglossaryentryglobaltrue 397

N

\NeedsTeXFormat ... 4, 438, 487, 527, 534, 557
\new@atom@glossaryentry 67
\new@command 440
\new@glossaryentry 69, 142
\new@ifnextchar .. 36, 100, 101, 126, 127, 175, 178–180, 227, 236–246, 426, 449, 450
\newabbr 21
\newabbreviation 21
\newabbreviationhook 233
\newabbreviationstyle 253, 255, 257, 258, 260, 261, 263, 264, 266, 268, 269, 271–278, 280, 282, 284, 286, 288, 290, 291, 293, 295–297, 299, 300, 302, 303, 305–315, 317–319, 321–323, 325, 327, 328, 330, 332, 333, 335, 337, 338, 340–343, 345–347, 349, 350, 352–354, 356, 358, 359, 361–363
\newacronym 134, 136
\newacronymhook 134
\newacronymstyle 135, 137
\newboolean 396
\newcommand 4–14, 16–27, 29–31, 33–47, 49–62, 64, 66, 67, 69–73, 75–77, 80–82, 85–87, 94, 99–101, 103–105, 107–110, 113, 115–122, 124, 126–130, 132, 133, 135–138, 143–150, 152, 153, 155–160, 162–168, 170–172, 174–193, 196–207, 213, 214, 216–231, 233–247, 249–255, 257, 258, 260, 264, 266, 269, 271–273, 289, 306, 331, 334–337, 340, 344, 345, 347, 349, 350, 353, 355, 356, 359, 361–363, 365–394, 396–408, 420–423, 425–427, 433, 434, 437, 439–445, 447–450, 452, 453, 456–485, 487, 489–491, 500, 501, 503–510, 517, 518, 527–529, 532–542, 544, 545, 547–553, 555, 558–561
\newcount 146, 160, 174, 392
\newcounter 26, 182
\newentry 22
\newenvironment 167
\newglossary 20, 139, 140
\newglossaryentry 22, 67, 69, 122, 130, 134, 205, 206, 233
\newglossaryentry options
access 194
alias 17, 57, 61, 63–66
desc 187, 188, 198, 199
descplural 188, 199
first 107, 185, 198, 253, 373, 374, 382
firstaccess 195
firstplural 186, 198, 253, 373, 374, 383
group 171, 172
loclist 37
long 190, 200, 384
longplural 190, 200, 385
name ... 58, 183, 184, 197, 216, 370, 371, 379
plural 184, 185, 197, 253, 372, 381
see 17, 31, 57, 63, 66, 69, 140
seealso 17, 57, 60, 63, 65, 66
short 58, 189, 199, 229
shortaccess 193
shortaccessplural 194
shortplural 190, 199, 229
symbol 186, 187, 198, 432
symbolplural 187, 198
text 58, 107, 184, 197, 253, 255, 371, 380
textaccess 195
user2 49
\newglossarystyle 529, 538, 540, 541, 543, 544, 546, 547, 549, 550, 552, 553, 555, 557, 561
\newif 80, 215, 253, 392, 396, 425, 538
\newlength 508, 559
\newnum 22
\newrobustcmd 35, 37–39, 41, 45–47, 61, 63, 68, 81, 86, 100, 101, 113–115, 126, 144, 150, 163, 164, 170, 171, 174–176, 178–180, 214, 221–223, 236–247, 344, 366, 368–378, 392, 393, 399, 400, 426, 449–452, 510–517
\newsym 22
\newterm 205
\newtoks 229
\newwrite 67, 139
\nfss@text 29, 30
\nobreak 489–491, 501, 558
\NoCaseChange 115–118, 164, 368–378
\noexpand 10, 12, 24, 61, 64, 65, 67, 82, 134, 154, 159, 167,

169, 170, 172, 182, 193, 217, 218, 233, 407, 409–411, 414, 417, 419, 426, 439, 449, 487, 529–531, 538–552, 554, 555, 562
\nofiles ... 157
\noindent ... 157, 158, 505, 507, 521–525, 558
\nopagebreak ... 501, 528, 532, 557, 562
\nopostdesc ... 52, 71, 72, 144, 168, 205
\np@glsxtr@assign@leveloffset ... 146
\ns@GLSxtrfull ... 237
\ns@Glsxtrfull ... 236
\ns@glsxtrfull ... 236
\ns@GLSxtrfullpl ... 238
\ns@Glsxtrfullpl ... 238
\ns@glsxtrfullpl ... 237
\ns@GLSxtrlong ... 242
\ns@Glsxtrlong ... 241, 242
\ns@glsxtrlong ... 241
\ns@GLSxtrlongpl ... 246
\ns@Glsxtrlongpl ... 245
\ns@glsxtrlongpl ... 245
\ns@GLSxtrshort ... 240
\ns@Glsxtrshort ... 240
\ns@glsxtrshort ... 239
\ns@GLSxtrshortpl ... 244
\ns@Glsxtrshortpl ... 243
\ns@glsxtrshortpl ... 243
\null ... 24
\number 67, 129, 130, 132, 159, 173, 174, 395, 400
\numexpr ... 129, 132, 172, 395, 400

O

\O ... 467, 470
\o ... 470
\openout ... 67
\or ... 6, 15, 16, 22, 23, 27, 28, 69, 80, 403–405, 411–416, 419, 502, 531
\org@glossaryentrynumbers ... 74, 143, 168
\org@glossarytitle ... 143, 167, 168
\org@ifKV@glslink@hyper ... 82, 85

P

\p@gls@hyp@opt ... 105
\p@glsxtr@assign@leveloffset ... 146
\p@GlsXtrMglsOrGls ... 427
package options:
  abbreviations ... 20
  accsupp ... 23, 183
  acronym ... 20
  automake ... 142, 155, 161
    immediate ... 161
    true ... 161
  autoseeindex ... 30
    false ... 30
  counter
    wrglossary ... 26
  debug
    showtargets ... 27, 109
  docdef ... 16, 68, 69, 122, 130, 396
    atom ... 67
    false ... 68, 69
    restricted ... 17
    true ... 67, 69
  docdefs
    restricted ... 68
  equations ... 8, 83
  indexcounter ... 440
  nonumberlist ... 74
  nopostdot ... 18
    false ... 18
  numbers ... 21
  order
    letter ... 162
  postdot ... 18
  prefix ... 23
  record ... 6, 13, 15, 68, 78, 138, 158, 159, 236, 238–246
    alsoindex ... 7, 9, 13, 15, 162
    hybrid ... 9, 13, 15
    nameref ... 11, 32, 159, 162
    only ... 7, 157
  shortcuts ... 22
    ac ... 22
    all ... 22
    false ... 22
    none ... 22
    true ... 22
  sort
    use ... 85
  style ... 25
  stylemods ... 25
  symbols ... 21, 206
  undefaction ... 56
    error ... 5
    warn ... 5
  xindy ... 62, 63
\PackageError ... 5, 12, 25, 30, 67, 69, 72, 73, 81, 100, 101, 104, 106, 114, 122, 123, 130, 132, 133, 137, 139, 141, 142,

151, 162, 170, 171, 174, 225, 250–252, 390, 392, 393, 395, 399–401, 425, 426, 488
\PackageWarning 18
\PackageWarningNoLine 18
\pagelistname ... 539, 542, 545, 549, 552, 555
\pageref 25, 49, 50
\par 157, 158, 490, 491, 501, 504–508, 518–520, 522–525, 528, 557–559, 561
\parindent 501, 504, 506, 508, 518–522, 524–526, 529, 557, 559, 562
\parskip 501, 504, 506, 521, 522, 524, 529
\PassOptionsToPackage 4, 24
\pdfbookmark 528, 529
\pdfstringdef 192, 193
\PLUS 427
\pp@glsxtr@assign@leveloffset 146
\preglossarypreamble 50
\preto 103, 448
\print@noop@unsrtglossaryunit ... 12, 15
\print@op@unsrtglossaryunit 15, 16
\printabbreviations 20
\printglossaries 14, 138, 156
\printglossary 14, 20, 138, 156–158
\printglossary options
nonumberlist 76
type 142
\printnoidxglossaries 156
\printnoidxglossary 141, 156
\printnumbers 22, 206
\printsymbols 22, 206
\printunsrtglossary 157, 158, 165, 438
\printunsrtglossaryentryprocesshook 166, 169, 170
\printunsrtglossaryhandler 170, 171
\printunsrtglossarypredoglossary 167, 169
\printunsrtglossaryskipentry .. 166, 169
\printunsrtglossaryunit 15, 16, 171
\printunsrtglossaryunitsetup 170
\ProcessOptions 488
\ProcessOptionsX 28
\protect 29, 107, 115–118, 197–200, 230, 234, 253–261, 263–266, 268–278, 280–295, 297, 298, 300–303, 305–310, 312–328, 330, 332, 333, 335, 337–343, 345–352, 354–356, 358–364, 368–378
\protected@cseappto 55
\protected@csedef . 44, 45, 150, 396, 509, 510
\protected@csxdef ... 45, 150, 395, 509, 510
\protected@eappto 12, 53–55, 136, 167, 169, 172, 216
\protected@edef 5, 7, 8, 10, 32, 37, 41, 46, 47, 53–56, 61, 64, 65, 68, 73, 81, 82, 84, 85, 101, 103, 106, 107, 109, 128, 129, 131, 134, 136, 139–141, 146, 147, 149, 150, 152, 163, 164, 170–172, 177, 178, 193, 194, 208, 209, 211–214, 216, 217, 221, 229, 233, 250, 394, 399, 408, 425, 444, 449, 512, 513
\protected@write 10–12, 68, 76, 106, 107, 139, 140, 159–161, 163, 174, 397, 425, 426, 449, 532
\protected@xdef 144, 168, 170
\providecommand 20, 21, 29, 35, 62, 76, 82, 101–103, 106, 123, 132, 138, 139, 154, 160, 161, 174, 408, 438, 439, 449, 454–456, 488–490, 527
\ProvidesFile 437
\ProvidesPackage .. 4, 438, 487, 527, 534, 557

Q

\quad 561
\quotechar 219

R

\raggedright 494, 495, 498, 499, 530, 536
\refstepcounter 26
\relax 6, 14, 15, 17, 21, 22, 24, 27, 28, 31, 36, 43, 61, 67, 69, 73, 74, 76, 79, 80, 82, 83, 105, 113, 123, 124, 132, 140, 143, 144, 146, 148, 149, 151, 159–162, 167, 168, 172, 174, 177, 217, 218, 220, 223, 225, 229, 230, 344, 392–396, 399, 401, 403, 404, 406, 408, 410–417, 419, 423, 424, 439, 440, 445–447, 490, 502, 504, 506, 510–520, 524–526, 529–532, 557, 559, 561, 562
relsize package 289
\renewcommand 5, 6, 14–20, 22–27, 31–33, 51–58, 67–70, 72–79, 82, 99, 101, 102, 104, 106, 107, 109, 113, 120–123, 127, 130–142, 144–146, 149–152, 154, 168, 171, 176, 177, 205, 208, 209, 211, 212, 220–224, 235, 251, 253–330, 332–343, 345–352, 354–365, 389, 390, 400, 401, 403–405, 422–424, 438, 439, 448, 485, 488, 490–502, 504–507, 518–525, 530, 531, 539–558, 561, 562

\renewenvironment 490, 492–499, 501, 504, 506, 518, 521–525, 529, 538–544, 546–552, 554, 555, 557, 561
\renewglossarystyle 488, 490–499, 501, 502, 504–507, 518–525
\renewrobustcmd 61, 85, 109
\RequireGlossariesExtraLang ... 437, 486
\RequirePackage 4, 15, 23–25, 28, 487, 527, 534, 557
\reserved@a 228
\reserved@b 228
\reserved@d 228
\RestoreAcronyms 134, 137
\rGLS 177
\rGls 176
\rgls 176
\rGLSformat 180
\rGlsformat 179
\rglsformat 179, 181
\rGLSpl 177
\rGlspl 177
\rglspl 177
\rGLSplformat 180
\rGlsplformat 180
\rglsplformat 179, 181
\robustify 439
\romannumeral 509, 510, 518, 560

S

\s@@glsxtrfmt 36
\s@gls@hyp@opt 105
\s@glsxtr@enabletagging 222
\s@glsxtrfmt 35
\s@glsxtrforcsvfield 38
\s@GlsXtrIfFieldCmpNum 42
\s@GlsXtrIfFieldEqNum 42
\s@GlsXtrIfFieldEqStr 46
\s@GlsXtrIfFieldEqXpStr 46
\s@GlsXtrIfFieldNonZero 42, 439
\s@GlsXtrIfFieldValueInCsvList 39
\s@glsxtrifhasfield 39–41, 46, 47
\s@GlsXtrIfHasNonZeroChildCount ... 439
\s@GlsXtrIfValueInFieldCsvList 40
\s@GlsXtrIfXpFieldEqXpStr 47
\s@GlsXtrMglsOrGls 427
\s@GlsXtrStartUnsetBuffering 119
\s@GlsXtrStopUnsetBuffering 119
\s@ifglossaryexists 33
\s@printunsrtglossary 165, 170
\s@xGlsXtrIfValueInFieldCsvList 40
\seealsoname 61, 63
\seename 58, 61
\setabbreviationstyle 137, 254, 266
\SetAcronymLists 135–137
\setacronymstyle 135, 137
\SetDefaultAcronymDisplayStyle 137
\setentrycounter 151, 442, 443
\SetGenericNewAcronym 137
\setglossarystyle 25, 143, 167, 488, 490–492, 502, 505, 507, 519–526, 529
\setkeys 8, 25, 31, 36, 83, 85, 104, 134, 143, 167, 168, 178, 231, 232, 388, 407, 409, 410
\setlength 73, 74, 501, 504, 506, 508, 519, 521, 522, 524, 529, 537, 538, 559, 560
\settowidth . 138, 423, 424, 508–518, 537, 560
\setupglossaries 4, 31
\sfcode 18, 19, 225, 500
\small 60
\smallskip 560
soul package 119
\space 5, 12, 14, 62, 69, 70, 72, 94, 104, 122–124, 130, 132–134, 137–141, 143, 154, 157, 158, 162, 167, 170, 171, 174, 226, 230, 234, 254, 395, 421, 423–426, 434, 488, 490, 500, 503, 504, 506, 508, 518, 527, 558, 560, 561
\spacefactor 18, 19, 225, 500
\stepcounter 182
\string 5, 10–12, 14, 32, 62, 63, 68–70, 72, 76, 84, 94, 100, 101, 104, 106, 107, 114, 115, 122–124, 130, 132–134, 137–143, 154–163, 167, 170, 171, 174, 210–213, 215, 217, 225, 397, 398, 408, 425, 426, 434, 439, 449, 456–485, 532
\strut 488, 490–500, 506, 535
\subglossentry 144, 168, 173, 488, 490–500, 502, 504, 506, 518, 530, 539, 541–543, 545, 546, 548, 550, 551, 553, 554, 556, 557, 562
\subitem 501, 502
\subsubitem 501, 502
\symbolname 537, 544, 545, 547, 548, 550, 552, 553, 555

T

\tabcolsep 537, 538
\tablehead 496–499
\tabletail 496–499

\tabularnewline 492–500, 536, 539–556
\TeX 155
\texorpdfstring .....
36, 37, 43, 44, 115–117, 220, 366, 378–387
\textbf 192, 500, 536, 559, 561
textcase package 364
\textit 192
\textmd 192
\textrm 192
\textsc 192, 272
\textsf 192
\textsl 192
\textsmaller 289
\texttt 155–158, 192
\the 134, 153, 160, 220, 233, 253–264, 266, 268, 269, 271–281, 284, 286–295, 297, 301–314, 316, 318, 320, 321, 323, 325–328, 330, 332, 333, 335, 337–343, 345–347, 349–356, 358, 359, 361, 362, 364, 394, 395, 399, 400, 444, 560
\theglsentrycounter ... 7, 9–11, 83, 85, 178
\theH 32
\theHglsentrycounter .. 7, 9–12, 83, 85, 178
\theindex 216
\thewrglossary 26
\this@dialect 437, 486
\tiny 29
\toks@ 153, 220, 444
\toprule 536, 539, 541, 542, 544, 545, 547, 548, 550, 552, 553, 555
\TrackedDialectClosestSubMatch 48
tracklang package 47, 48, 161, 456
\TrackLangGetDefaultScript 485
\TrackLangIfHasDefaultScript 485
\TrackLangRequireDialectPrefix 485, 486
\triangleleft 29
\triangleright 29, 60
\ttfamily 29

U

\u 439
\undef 15, 16, 67, 222
\underline 223
\unskip 52, 67, 488
upgreek package 455
\usepackage 156–158

W

\warn@nomakeglossaries 140
\warn@noprintglossary .. 139, 140, 144, 168
wrglossary (counter) 25, 26
\write 62, 123, 124, 132, 139, 154, 161
\writemultiglossentry 396, 397

X

\xappto 396, 414, 417, 419, 425, 426
\xcapitalisewords 207
\xdef 411, 439
\xifinlist 128, 393
\xifinlistcs 38
xindy 15, 138
xkeyval package 4
\XKV@checkchoice 77
\XKV@plfalse 76
\XKV@resa 77
\XKV@rm 407, 409
\XKV@sttrue 76
\xmakefirstuc 433