glossaries-extra.sty v1.08: documented code

Nicola L.C. Talbot

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

2016-12-13

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) 4
- 1.1 Package Initialisation and Options . . . 4
- 1.2 Extra Utilities . . . 15
- 1.3 Modifications to Commands Provided by glossaries . . . 15
  - 1.3.1 Existence Checks . . . 15
  - 1.3.2 Document Definitions . . . 18
  - 1.3.3 Existing Glossary Style Modifications . . . 23
  - 1.3.4 Entry Formatting, Hyperlinks and Indexing . . . 27
  - 1.3.5 Entry Counting . . . 56
  - 1.3.6 Acronym Modifications . . . 69
  - 1.3.7 Indexing and Displaying Glossaries . . . 72
- 1.4 Integration with glossaries-accsupp . . . 82
- 1.5 Categories . . . 92
- 1.6 Abbreviations . . . 115
  - 1.6.1 Abbreviation Styles Setup . . . 132
  - 1.6.2 Predefined Styles (Default Font) . . . 135
  - 1.6.3 Predefined Styles (Small Capitals) . . . 147
  - 1.6.4 Predefined Styles (Fake Small Capitals) . . . 151
  - 1.6.5 Predefined Styles (Emphasized) . . . 155
  - 1.6.6 Predefined Styles (User Parentheses Hook) . . . 161
- 1.7 Using Entries in Headings . . . 165
- 1.8 Multi-Lingual Support . . . 182

2 Style Adjustments (glossaries-extra-stylemods.sty) 184
- 2.1 Package Initialisation . . . 184
- 2.2 List-Like Styles . . . 185
- 2.3 Longtable Styles . . . 185
- 2.4 Long Ragged Styles . . . 186
- 2.5 Supertabular Styles . . . 188
- 2.6 Super Ragged Styles . . . 189
- 2.7 Inline Style . . . 190
- 2.8 Tree Styles . . . 190

Glossary 202

Change History 203

Index 212

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra}[2016/12/13 v1.08 (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{glossaries-extra}{%
  \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.)}%
}

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

In the event that someone wants to develop a post-processor that needs to know what entries have been used in the document, 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}[2]{}
```

@@glsxtr@record This is the actual code that does the recording 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.

```
\newcommand*{\@@glsxtr@record}[2]{%
 \begingroup
   \def\@glsnumberformat{glsnumberformat}%
   \ifcsdef{glo@#2@counter}%
   {%
     \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
   }%
   {%
```

Entry hasn't been defined, so we'll have to assume the page number by default.

```
     \def\@gls@counter{page}%
   }%
   \setkeys{glslink}{#1}%
   \ifKV@glslink@noindex
   \else
     \glswriteentry{#2}%
     {%
```

Save the entry counter.

```
       \glsxtr@saveentrycounter
```

Temporarily redefine \@@do@@wrglossary so we can use \glsxtr@@do@wrglossary.

```
       \let\@@do@@wrglossary\@glsxtr@dorecord
       \glsxtr@@do@wrglossary{#2}%
     }%
   \fi
 \endgroup
}
```

glsxtr@dorecord

```
\newcommand*\@glsxtr@dorecord{%
   \protected@write\@auxout{}{\string\glsxtr@record
      {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
      {\@glslocref}}%
}
```

\glsxtr@record

```
\newcommand*{\glsxtr@record}[5]{}
```

tr@setup@record Initialise.

```
\newcommand*{\glsxtr@setup@record}{}
```

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

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

addloclistfield

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

Now define the record package option.

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

Don't record.

```
     \def\glsxtr@setup@record{%
       \renewcommand*{\@glsxtr@record}[2]{}%
       \let\@@do@wrglossary\glsxtr@@do@wrglossary
       \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
       \let\glsxtrundefaction\@glsxtr@err@undefaction
       \let\glsxtr@warnonexistsordo\@gobble
     }%
   \or
```

Only record (don't index).

```
     \def\glsxtr@setup@record{%
       \let\@glsxtr@record\@@glsxtr@record
       \let\@@do@wrglossary\@gobble
       \let\@gls@saveentrycounter\relax
       \let\glsxtrundefaction\@glsxtr@warn@undefaction
       \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
```

```
      \glsxtr@addloclistfield
    }%
  \or
```

Record and index.

```
    \def\glsxtr@setup@record{%
      \let\@glsxtr@record\@@glsxtr@record
      \let\@@do@wrglossary\glsxtr@@do@wrglossary
      \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
      \let\glsxtrundefaction\@glsxtr@warn@undefaction
      \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
      \glsxtr@addloclistfield
    }%
  \fi
}
```

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

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

```
\newcount\@glsxtr@docdefval
```

Need to provide conditional commands that are backward compatible:

if@glsxtrdocdef

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

lsxtrdocdeftrue

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

sxtrdocdeffalse

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

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}[\val\nr]%
 {false,true,restricted}[true]%
{%
  \@glsxtr@docdefval=\nr\relax
  \ifnum\@glsxtr@docdefval=2\relax
    \renewcommand*{\@glsdoifexistsorwarn}{\glsdoifexists}%
  \fi
}
```

ocdefrestricted

```
\newcommand*{\if@glsxtrdocdefrestricted}{\ifnum\@glsxtr@docdefval=2 }
```

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}
```

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}%
}
```

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.

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

Disable this command after it's been used.

```
  \let\GlsXtrDefineAbbreviationShortcuts\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 portablity 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}{}
```

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.

```
\define@choicekey{glossaries-extra.sty}{shortcuts}[\val\nr]%
 {acronyms,acro,abbreviations,abbr,other,all,true,none,false}[true]{%
   \ifcase\nr\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
      \DefineAcronymSynonyms
      \GlsXtrDefineAbbreviationShortcuts
      \GlsXtrDefineOtherShortcuts
    }%
  \or % true
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \glsacrshortcutstrue
      \DefineAcronymSynonyms
      \GlsXtrDefineAbbreviationShortcuts
      \GlsXtrDefineOtherShortcuts
    }%
  \else % none, false
    \renewcommand*{\@glsxtr@setupshortcuts}{}%
  \fi
}
```

`\@glsxtr@doaccsupp`

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

`accsupp` If accsupp, load glossaries-accsupp package.

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

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

```
\newcommand{\glsxtrNoGlossaryWarning}[1]{%
  \@glsxtr@defaultnoglossarywarning{#1}%
}
```

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

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

glsxtr@do@style

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

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

```
\define@key{glossaries-extra.sty}{style}{%
 \renewcommand*{\@glsxtr@do@style}{%
```

Set this as the default style:

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

Set this style:

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

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
```

Define abbreviations glossaries if required.

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

Setup shortcuts if required.

```
\@glsxtr@setupshortcuts
```

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

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

Now define the user command:

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

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

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

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
```

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

rifemptyglossary

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

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

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

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.

1.3.1 Existence Checks

\glsdoifexists

Modify \glsdoifexists to take account of the undefaction setting.

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

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

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

glsdoifnoexists

Modify \glsdoifnoexists to take account of the undefaction setting.

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

```
    \glsxtrundefaction{Glossary entry `\glsdetoklabel{#1}'
    has already been defined}{}{#2}%
}
```

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

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

arynoexistsordo Similarly for \doifglossarynoexistsordo.

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

```
      #3%
    }%
    {#2}%
  }%
}
```

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}%
     \@glsxtr@autoindexcrossrefs
   }%
}
\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}{}%
}
```

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}{\@glsxtr@addunusedxrefs{\@glo@label}}{}%
     }%
  }%
}
```

@addunusedxrefs If the given entry has a see field add all unused cross-references.

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

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

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

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

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

xtrunusedformat

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

1.3.2 Document Definitions

noidxglossaries Modify \makenoidxglossaries so that it automatically switches off (unless the restricted setting is on) and disables the docdef key.

```
\let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
```

```
\renewcommand{\makenoidxglossaries}{%
  \glsxtr@orgmakenoidxglossaries
  \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}%
}
```

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 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
  \or
```

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:

```
\let\@glsxtr@orgprintglossary\@printglossary
\renewcommand\@printglossary[2]{%
  \@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 the above.

`etglossarystyle`

```
\renewcommand*{\setglossarystyle}[1]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries}{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}%
    {\glsxtrgenabbrvfmt}%
    {\glsxtrregularfont{\glsgenentryfmt}}%
  }%
}
```

sxtrregularfont Font used for regular entries.

```
\newcommand*{\glsxtrregularfont}[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 enter exists.

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

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

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

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

\@gls@ Save the original definition and redefine.

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

\@GLSpl@ Save the original definition and redefine.

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

\@glsdispl Save the original definition and redefine.

```
\let\@glsxtr@org@glsdisp\@glsdisp
\renewcommand*{\@glsdisp}[3][]{%
  \@glsxtr@record{#1}{#2}%
  \@glsxtr@org@glsdisp[#1]{#2}{#3}%
}
```

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

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

\glsadd Redefine to include \@glsxtr@record

```
\renewrobustcmd*{\glsadd}[2][]{%
  \@gls@adjustmode
  \@glsxtr@record{#1}{#2}%
  \glsdoifexists{#2}%
  {%
```

```
    \def\@glsnumberformat{glsnumberformat}%
    \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
    \setkeys{glossadd}{#1}%
    \@gls@saveentrycounter
    \@@do@wrglossary{#2}%
  }%
}
```

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

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

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

`assignfieldfont`

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

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

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

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

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

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

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

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

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

ecknohyperfirst

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

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

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

Ensure that \glsfirst honours the nohyperfirst attribute.

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

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

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

Ensure that \Glsfirst honours the nohyperfirst attribute.

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

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

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

Ensure that \GLSfirst honours the nohyperfirst attribute.

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

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

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

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

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

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

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

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

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

Ensure that \glsfirstplural honours the nohyperfirst attribute.

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

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

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

Ensure that \glsfirstplural honours the nohyperfirst attribute.

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

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

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

Ensure that \glsfirstplural honours the nohyperfirst attribute.

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

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

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

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

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

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

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

\@glsdesc@

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

\@Glsdesc@ First letter uppercase version.

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

\@GLSdesc@ All uppercase version.

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

@glsdescplural@ No case-changing version.

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

@Glsdescplural@ First letter uppercase version.

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

@GLSdescplural@ All uppercase version.

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

\@glssymbol@

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

\@Glssymbol@ First letter uppercase version.

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

\@GLSsymbol@ All uppercase version.

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

lssymbolplural@ No case-changing version.

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

lssymbolplural@ First letter uppercase version.

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

LSsymbolplural@ All uppercase version.

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

\@Glsuseri@ First letter uppercase version.

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

\@GLSuseri@ All uppercase version.

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

\@Glsuserii@ First letter uppercase version.

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

\@GLSuserii@ All uppercase version.

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

\@Glsuseriii@ First letter uppercase version.

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

\@GLSuseriii@ All uppercase version.

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

\@Glsuseriv@ First letter uppercase version.

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

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

\@GLSuseriv@ All uppercase version.

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

\@Glsuserv@ First letter uppercase version.

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

\@GLSuserv@ All uppercase version.

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

\@Glsuservi@ First letter uppercase version.

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

\@GLSuservi@ All uppercase version.

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

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

\@acrshort No case change.

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

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

`\@Acrshort` First letter uppercase.

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

`\@ACRshort` All uppercase.

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

\@acrshortpl No case change.

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

\@Acrshortpl First letter uppercase.

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

\@ACRshortpl All uppercase.

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

```
}
```

`\@acrlong` No case change.

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

`\@Acrlong` First letter uppercase.

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

`\@ACRlong` All uppercase.

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

```
  }%
  \glspostlinkhook
}
```

`\@acrlongpl` No case change.

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

`\@Acrlongpl` First letter uppercase.

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

`\@ACRlongpl` All uppercase.

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

```
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

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

\@glsaddkey

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

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

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

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

```
    \ifcsdef{@Gls@user@#1@}%
    {%
       \PackageError{glossaries}%
       {Can't define `\string#6' as helper command
```

```
      `\expandafter\string\csname @Gls@user@#1@\endcsname' already
       exists}%
     {}%
  }%
  {%
    \expandafter\newcommand\expandafter*\expandafter
      {\csname @Gls@user@#1\endcsname}[2][]{%
        \new@ifnextchar[%
          {\csuse{@Gls@user@#1@}{##1}{##2}}%
          {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
    \csdef{@Gls@user@#1@}##1##2[##3]{%
      \@gls@field@link[\let\glscapscase\@secondofthree]%
        {##1}{##2}{#4{##2}##3}%
    }%
    \newrobustcmd*{#6}{%
      \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
  }%
```

Finally the all caps version (modified):

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

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

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

`checkfirsthyper` Modify check to determine if the hyperlink should be automatically suppressed, but save the

original in case the acronyms are restored.

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

`\ifglsused` isn't useful in the post link hook as it's already been unset by then, so define a command that can be used in the post link hook. Since `\@gls@link@checkfirsthyper` is only used by commands like `\gls` but not by other commands, this seems the best place to put it.

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

Store the category label for convenience.

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

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

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

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

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

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

lt@glslink@opts

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

```
  }
}
{
```

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

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

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

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

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

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

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

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

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

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

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

Similarly for the "noidx" version:

s@noidxglossary

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

xtr@do@@wrindex

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

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

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

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

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

alt@gls@hyp@opt User version

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

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

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

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

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

rSetAltModifier

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

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

```
\renewcommand*{\glsdohyperlink}[2]{%
 \hyperlink{#1}{{\glsxtrprotectlinks#2}}}
```

glsdisablehyper Redefine in case we have an old version of glossaries.

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

lsdonohyperlink This command was only introduced in glossaries v4.20, so it may not be defined. For older glossaries versions, this won't be used if hyperref hasn't been loaded, which means the indexing will still take place.

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

Reset \@glslink with patched versions:

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

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

```
\newcommand*{\glsxtrprotectlinks}{%
  \KV@glslink@hyperfalse
  \KV@glslink@noindextrue
  \let\@gls@\@glsxtr@p@text@
  \let\@Gls@\@Glsxtr@p@text@
  \let\@GLS@\@GLSxtr@p@text@
  \let\@glspl@\@glsxtr@p@plural@
  \let\@Glspl@\@Glsxtr@p@plural@
  \let\@GLSpl@\@GLSxtr@p@plural@
  \let\@glsxtrshort\@glsxtr@p@short@
```

```
  \let\@Glsxtrshort\@Glsxtr@p@short@
  \let\@GLSxtrshort\@GLSxtr@p@short@
  \let\@glsxtrlong\@glsxtr@p@long@
  \let\@Glsxtrlong\@Glsxtr@p@long@
  \let\@GLSxtrlong\@GLSxtr@p@long@
  \let\@glsxtrshortpl\@glsxtr@p@shortpl@
  \let\@Glsxtrshortpl\@Glsxtr@p@shortpl@
  \let\@GLSxtrshortpl\@GLSxtr@p@shortpl@
  \let\@glsxtrlongpl\@glsxtr@p@longpl@
  \let\@Glsxtrlongpl\@Glsxtr@p@longpl@
  \let\@GLSxtrlongpl\@GLSxtr@p@longpl@
  \let\@acrshort\@glsxtr@p@acrshort@
  \let\@Acrshort\@Glsxtr@p@acrshort@
  \let\@ACRshort\@GLSxtr@p@acrshort@
  \let\@acrshortpl\@glsxtr@p@acrshortpl@
  \let\@Acrshortpl\@Glsxtr@p@acrshortpl@
  \let\@ACRshortpl\@GLSxtr@p@acrshortpl@
  \let\@acrlong\@glsxtr@p@acrlong@
  \let\@Acrlong\@Glsxtr@p@acrlong@
  \let\@ACRlong\@GLSxtr@p@acrlong@
  \let\@acrlongpl\@glsxtr@p@acrlongpl@
  \let\@Acrlongpl\@Glsxtr@p@acrlongpl@
  \let\@ACRlongpl\@GLSxtr@p@acrlongpl@
}
```

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

@glsxtr@p@text@

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

@Glsxtr@p@text@

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

@GLSxtr@p@text@

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

lsxtr@p@plural@

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

lsxtr@p@plural@

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

LSxtr@p@plural@

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

glsxtr@p@short@

```
\def\@glsxtr@p@short@#1#2[#3]{%
 {%
```

```
   \glssetabbrvfmt{\glscategory{#2}}%
   \glsabbrvfont{\glsentryshort{#2}}#3%
  }%
}
```

Glsxtr@p@short@

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

GLSxtr@p@short@

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

sxtr@p@shortpl@

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

sxtr@p@shortpl@

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

Sxtr@p@shortpl@

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

@glsxtr@p@long@

```
\def\@glsxtr@p@long@#1#2[#3]{{\glsentrylong{#2}#3}}
```

```
@Glsxtr@p@long@
1596 \def\@Glsxtr@p@long@#1#2[#3]{{\Glsentrylong{#2}#3}}

@GLSxtr@p@long@
1597 \def\@GLSxtr@p@long@#1#2[#3]{%
1598   {\mfirstucMakeUppercase{\glslongfont{\glsentrylong{#2}}#3}}}

lsxtr@p@longpl@
1599 \def\@glsxtr@p@longpl@#1#2[#3]{{\glsentrylongpl{#2}#3}}

lsxtr@p@longpl@
1600 \def\@Glsxtr@p@longpl@#1#2[#3]{{\glslongfont{\Glsentrylongpl{#2}}#3}}

LSxtr@p@longpl@
1601 \def\@GLSxtr@p@longpl@#1#2[#3]{%
1602   {\mfirstucMakeUppercase{\glslongfont{\glsentrylongpl{#2}}#3}}}

xtr@p@acrshort@
1603 \def\@glsxtr@p@acrshort@#1#2[#3]{{\acronymfont{\glsentryshort{#2}}#3}}

xtr@p@acrshort@
1604 \def\@Glsxtr@p@acrshort@#1#2[#3]{{\acronymfont{\Glsentryshort{#2}}#3}}

xtr@p@acrshort@
1605 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
1606   {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}

r@p@acrshortpl@
1607 \def\@glsxtr@p@acrshortpl@#1#2[#3]{{\acronymfont{\glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
1608 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
1609 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
1610   {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
1611 \def\@glsxtr@p@acrlong@#1#2[#3]{{\glsentrylong{#2}#3}}

sxtr@p@acrlong@
1612 \def\@Glsxtr@p@acrlong@#1#2[#3]{{\Glsentrylong{#2}#3}}

Sxtr@p@acrlong@
1613 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
1614  {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}
```

```
tr@p@acrlongpl@
1615 \def\@glsxtr@p@acrlongpl@#1#2[#3]{{\glsentrylongpl{#2}#3}}
tr@p@acrlongpl@
1616 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{{\Glsentrylongpl{#2}#3}}
tr@p@acrlongpl@
1617 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
1618  {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}
```

Commands to minimise conflict.

```
  \@glsxtrp@opt
1619 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}
```

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

```
1620 \newcommand*{\glsxtrsetpopts}[1]{%
1621   \renewcommand*{\@glsxtrp@opt}{#1}%
1622 }
```

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

```
1623 \newcommand*{\glossxtrsetpopts}{%
1624   \glsxtrsetpopts{noindex}%
1625 }
```

```
     \@@glsxtrp
1626 \newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
1627   {%
1628     \let\glspostlinkhook\relax
1629     \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
1630   }%
1631 }
```

```
      \@glsxtrp
1632 \newrobustcmd*{\@glsxtrp}[2]{%
1633   \ifcsdef{gls#1}%
1634   {%
1635     \@@glsxtrp{gls#1}{#2}%
1636   }%
1637   {%
1638     \ifcsdef{glsxtr#1}%
1639     {%
1640       \@@glsxtrp{glsxtr#1}{#2}%
1641     }%
1642     {%
1643       \PackageError{glossaries-extra}{`#1' not recognised by
1644         \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}%
      }%
    }%
  }
```

```
1735 }
```

Provide short synonyms for the most common option.

\glsps

```
1736 \newcommand*{\glsps}{\glsxtrp{short}}
```

\glspt

```
1737 \newcommand*{\glspt}{\glsxtrp{text}}
```

\Glsxtrp As above but use first letter upper case (but not for the bookmarks, which can't process \uppercase).

```
1738 \ifdef\texorpdfstring
1739 {
1740   \newcommand{\Glsxtrp}[2]{%
1741     \protect\NoCaseChange
1742     {%
1743       \protect\texorpdfstring
1744       {%
1745         \protect\glsxtrifinmark
1746         {%
1747           \ifcsdef{Glsxtrhead#1}%
1748           {%
1749             {\protect\csuse{Glsxtrhead#1}{#2}}%
1750           }%
1751           {%
1752             \protect\@Gls@entry@field{#2}{#1}%
1753           }%
1754         }%
1755         {%
1756           \@Glsxtrp{#1}{#2}%
1757         }%
1758       }%
1759       {%
1760         \protect\@gls@entry@field{#2}{#1}%
1761       }%
1762     }%
1763   }
1764 }
1765 {
1766   \newcommand{\Glsxtrp}[2]{%
1767     \protect\NoCaseChange
1768     {%
1769       \protect\glsxtrifinmark
1770       {%
1771         \ifcsdef{Glsxtrhead#1}%
1772         {%
1773           {\protect\csuse{Glsxtrhead#1}}%
1774         }%
```

```
      {%
        \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}%
```

```
1822          {%
1823            {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
1824          }%
1825          {%
1826            \protect\mfirstucMakeUppercase
1827            {%
1828              \protect\@gls@entry@field{#2}{#1}%
1829            }%
1830          }%
1831        }%
1832        {%
1833          \@GLSxtrp{#1}{#2}%
1834        }%
1835      }%
1836    }
1837 }
```

1.3.5 Entry Counting

The 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.

First adjust definitions of the unset and reset commands to provide a hook.

\@glsunset Global unset.

```
1838 \renewcommand*{\@glsunset}[1]{%
1839   \@@glsunset{#1}%
1840   \glsxtrpostunset{#1}%
1841 }%
```

glsxtrpostunset

```
1842 \newcommand*{\glsxtrpostunset}[1]{}
```

\@glslocalunset Local unset.

```
1843   \renewcommand*{\@glslocalunset}[1]{%
1844     \@@glslocalunset{#1}%
1845     \glsxtrpostlocalunset{#1}%
1846   }%
```

rpostlocalunset

```
1847 \newcommand*{\glsxtrpostlocalunset}[1]{}
```

\@glsreset Global reset.

```
1848 \renewcommand*{\@glsreset}[1]{%
1849   \@@glsreset{#1}%
1850   \glsxtrpostreset{#1}%
1851 }%
```

glsxtrpostreset

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

\@glslocalreset Local reset.

```
\renewcommand*{\@glslocalreset}[1]{%
  \@@glslocalreset{#1}%
  \glsxtrpostlocalreset{#1}%
}%
```

rpostlocalreset

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

leEntryCounting The first argument is the list of categories and the second argument is the value of the entrycount attribute.

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

Enable entry counting:

```
  \glsenableentrycount
```

Redefine \gls etc:

```
  \renewcommand*{\gls}{\cgls}%
  \renewcommand*{\Gls}{\cGls}%
  \renewcommand*{\glspl}{\cglspl}%
  \renewcommand*{\Glspl}{\cGlspl}%
  \renewcommand*{\GLS}{\cGLS}%
  \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
  \@glsxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
  \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
  \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
    can't be used with \string\GlsXtrEnableEntryCounting}%
   {Use one or other but not both commands}}%
}
```

ycountunsetattr

```
\newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
 \@for\@glsxtr@cat:=#1\do
 {%
   \ifdefempty{\@glsxtr@cat}{}%
   {%
     \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
   }%
 }%
}
```

Redefine the entry counting commands to take into account the entrycount attribute.

nableentrycount

```
\renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
  \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defcounters}%
```

Just in case the user has switched on the docdef option.

```
  \renewcommand*{\gls@defdocnewglossaryentry}{%
    \renewcommand*\newglossaryentry[2]{%
      \PackageError{glossaries}{\string\newglossaryentry\space
      may only be used in the preamble when entry counting has
      been activated}{If you use \string\glsenableentrycount\space
      you must place all entry definitions in the preamble not in
      the document environment}%
    }%
  }%
```

New commands to access new fields:

```
  \newcommand*{\glsentrycurrcount}[1]{%
   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
   {0}{\@gls@entry@field{##1}{currcount}}%
  }%
  \newcommand*{\glsentryprevcount}[1]{%
   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
   {0}{\@gls@entry@field{##1}{prevcount}}%
  }%
```

Adjust post unset and reset:

```
  \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
  \renewcommand*{\glsxtrpostunset}[1]{%
    \@glsxtr@entrycount@org@unset{##1}%
    \@gls@increment@currcount{##1}%
  }%
  \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
  \renewcommand*{\glsxtrpostlocalunset}[1]{%
    \@glsxtr@entrycount@org@localunset{##1}%
    \@gls@local@increment@currcount{##1}%
  }%
  \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
  \renewcommand*{\glsxtrpostreset}[1]{%
    \@glsxtr@entrycount@org@reset{##1}%
    \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
  }%
  \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
  \renewcommand*{\glsxtrpostlocalreset}[1]{%
    \@glsxtr@entrycount@org@localreset{##1}%
    \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
  }%
```

Modifications to take into account the attributes that govern whether the entry should be unset.

```
  \let\@cgls@\@@cgls@
  \let\@cglspl@\@@cglspl@
  \let\@cGLS@\@@cGLS@
  \let\@cGlspl@\@@cGlspl@
  \let\@cGLS@\@@cGLS@
  \let\@cGLSpl@\@@cGLSpl@
```

The rest is as the original definition.

```
  \AtEndDocument{\@gls@write@entrycounts}%
  \renewcommand*{\@gls@entry@count}[2]{%
    \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
  }%
  \let\glsenableentrycount\relax
  \renewcommand*{\glsenableentryunitcount}{%
    \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
     can't be used with \string\glsenableentrycount}%
    {Use one or other but not both commands}%
  }%
}
```

ite@entrycounts Modify this command so that it only writes the information for entries with the entrycount attribute and issue warning if no entries have this attribute set.

```
\renewcommand*{\@gls@write@entrycounts}{%
  \immediate\write\@auxout
    {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
  \count@=0\relax
  \forallglsentries{\@glsentry}{%
    \glshasattribute{\@glsentry}{entrycount}%
    {%
      \ifglsused{\@glsentry}%
      {%
        \immediate\write\@auxout
         {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
      }%
      {}%
      \advance\count@ by \@ne
    }%
    {}%
  }%
  \ifnum\count@=0
    \GlossariesExtraWarningNoLine{Entry counting has been enabled
     \MessageBreak with \string\glsenableentrycount\space but the
     \MessageBreak attribute 'entrycount' hasn't
     \MessageBreak been assigned to any of the defined
     \MessageBreak entries}%
  \fi
}
```

trifcounttrigger | \glsxtrifcounttrigger{⟨*label*⟩}{⟨*trigger format*⟩}{⟨*normal*⟩}

```
\newcommand*{\glsxtrifcounttrigger}[3]{%
 \glshasattribute{#1}{entrycount}%
 {%
   \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
    #3%
   \else
    #2%
   \fi
 }%
 {#3}%
}
```

Actual internal definitions of \cgls used when entry counting is enabled.

\@@cgls@

```
\def\@@cgls@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cglsformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@gls@{#1}{#2}[#3]%
  }%
}%
```

\@@cgls@

```
\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 \cs{cgls} etc so that it can be used
% interchangeable with \cs{gls} etc.
%\begin{macro}{\@cgls@}
%    \begin{macrocode}
\def\@cgls@#1#2[#3]{\@gls@{#1}{#2}[#3]}
```

\@cGls@

```
\def\@cGls@#1#2[#3]{\@Gls@{#1}{#2}[#3]}
```

\@cglspl@

```
\def\@cglspl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}
```

\@cGlspl@

```
\def\@cGlspl@#1#2[#3]{\@Glspl@{#1}{#2}[#3]}
```

Add all upper case versions not provided by glossaries.

\cGLS

```
\newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}
```

\@cGLS Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@cGLS}[2][]{%
  \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[]}%
}
```

\@cGLS@

```
\def\@cGLS@#1#2[#3]{\@GLS@{#1}{#2}[#3]}
```

\cGLSformat Format used by \cGLS if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
\newcommand*{\cGLSformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\cglsformat{#1}{#2}}%
}
```

\cGLSpl

```
\newrobustcmd*{\cGLSpl}{\@gls@hyp@opt\@cGLSpl}
```

\@cGLSpl Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@cGLSpl}[2][]{%
  \new@ifnextchar[{\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2}[]}%
}
```

\@cGLSpl@

```
\def\@cGLSpl@#1#2[#3]{\@GLSpl@{#1}{#2}[#3]}
```

\cGLSplformat Format used by \cGLSpl if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
\newcommand*{\cGLSplformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
}
```

Modify the trigger formats to check for the regular attribute.

\cglsformat

```
\renewcommand*{\cglsformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}#2%
}
```

\cGlsformat

```
\renewcommand*{\cGlsformat}[2]{%
  \glsifregular{#1}
  {\Glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}#2%
}
```

\cglsplformat

```
\renewcommand*{\cglsplformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}#2%
}
```

\cGlsplformat

```
\renewcommand*{\cGlsplformat}[2]{%
  \glsifregular{#1}
  {\Glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}#2%
}
```

New code similar to above for unit counting.

\defunitcounters

```
\newcommand*{\@@newglossaryentry@defunitcounters}{%
  \edef\@glo@countunit{\csuse{@glsxtr@categoryattr@@\@glo@category @unitcount}}%
  \ifdefvoid\@glo@countunit
  {}%
  {%
    \@glsxtr@ifunitcounter{\@glo@countunit}%
    {}%
    {\expandafter\@glsxtr@addunitcounter\expandafter{\@glo@countunit}}%
  }%
}
```

\@glsxtr@unitcountlist List to keep track of which counters are being used by the entry unit count facility.

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

\@addunitcounter

```
\newcommand*{\@glsxtr@addunitcounter}[1]{%
 \listadd{\@glsxtr@unitcountlist}{#1}%
```

```
 \ifcsundef{glsxtr@theunit@#1}
 {%
   \ifcsdef{theH#1}%
   {\csdef{glsxtr@theunit@#1}{\csuse{theH#1}}}%
   {\csdef{glsxtr@theunit@#1}{\csuse{the#1}}}%
 }%
 {}%
}

r@ifunitcounter

\newcommand*{\@glsxtr@ifunitcounter}[3]{%
  \xifinlist{#1}{\@glsxtr@unitcountlist}{#2}{#3}%
}

urrentunitcount

\newcommand*\@glsxtr@currentunitcount[1]{%
 glo@\glsdetoklabel{#1}@currunit@\glsgetattribute{#1}{unitcount}.%
 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
}

eviousunitcount

\newcommand*\@glsxtr@previousunitcount[1]{%
 glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
}

t@currunitcount

\newcommand*{\@gls@increment@currunitcount}[1]{%
  \glshasattribute{#1}{unitcount}%
  {%
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
    \ifcsundef{\@glsxtr@csname}%
    {%
      \csgdef{\@glsxtr@csname}{1}%
      \listcsxadd
       {glo@\glsdetoklabel{#1}@unitlist}%
       {\glsgetattribute{#1}{unitcount}.%
        \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
       }%
    }%
    {%
      \csxdef{\@glsxtr@csname}%
      {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
    }%
  }%
  {}%
}

t@currunitcount
```

```
\newcommand*{\@gls@local@increment@currunitcount}[1]{%
  \glshasattribute{#1}{unitcount}%
  {%
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
    \ifcsundef{\@glsxtr@csname}%
    {%
      \csdef{\@glsxtr@csname}{1}%
      \listcseadd
       {glo@\glsdetoklabel{#1}@unitlist}%
       {\glsgetattribute{#1}{unitcount}.%
        \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
       }%
    }%
    {%
      \csedef{\@glsxtr@csname}%
      {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
    }%
  }%
  {}%
}
```

r@currunitcount

```
\newcommand*{\@glsxtr@currunitcount}[2]{%
 \ifcsundef
 {glo@\glsdetoklabel{#1}@currunit@#2}%
 {0}%
 {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
}%
```

r@prevunitcount

```
\newcommand*{\@glsxtr@prevunitcount}[2]{%
 \ifcsundef
 {glo@\glsdetoklabel{#1}@prevunit@#2}%
 {0}%
 {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
}%
```

eentryunitcount

```
\newcommand*{\glsenableentryunitcount}{%
```

Enable new fields:

```
  \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defunitcounters}%
```

Just in case the user has switched on the docdef option.

```
  \renewcommand*{\gls@defdocnewglossaryentry}{%
    \renewcommand*\newglossaryentry[2]{%
      \PackageError{glossaries}{\string\newglossaryentry\space
      may only be used in the preamble when entry counting has
      been activated}{If you use \string\glsenableentryunitcount\space
      you must place all entry definitions in the preamble not in
```

```
      the document environment}%
    }%
  }%
```

New commands to access new fields:

```
  \newcommand*{\glsentrycurrcount}[1]{%
    \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}.%
     \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
  }%
  \newcommand*{\glsentryprevcount}[1]{%
    \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}.%
     \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
  }%
```

Access total count:

```
  \newcommand*{\glsentryprevtotalcount}[1]{%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
    {0}%
    {%
      \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
    }%
  }%
```

Access max value:

```
  \newcommand*{\glsentryprevmaxcount}[1]{%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
    {0}%
    {%
      \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
    }%
  }%
```

Adjust post unset and reset:

```
  \let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
  \renewcommand*{\glsxtrpostunset}[1]{%
    \@glsxtr@entryunitcount@org@unset{##1}%
    \@gls@increment@currunitcount{##1}%
  }%
  \let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
  \renewcommand*{\glsxtrpostlocalunset}[1]{%
    \@glsxtr@entryunitcount@org@localunset{##1}%
    \@gls@local@increment@currunitcount{##1}%
  }%
  \let\@glsxtr@entryunitcount@org@reset\glsxtrpostreset
  \renewcommand*{\glsxtrpostreset}[1]{%
    \glshasattribute{##1}{unitcount}%
    {%
      \edef\@glsxtr@csname{\@glsxtr@currentunitcount{##1}}%
      \ifcsundef{\@glsxtr@csname}%
      {}%
      {\csgdef{\@glsxtr@csname}{0}}%
```

```
    }%
    {}%
  }%
  \let\@glsxtr@entryunitcount@org@localreset\glsxtrpostlocalreset
  \renewcommand*{\glsxtrpostlocalreset}[1]{%
    \@glsxtr@entryunitcount@org@localreset{##1}%
    \glshasattribute{##1}{unitcount}%
    {%
      \edef\@glsxtr@csname{\@glsxtr@currentunitcount{##1}}%
      \ifcsundef{\@glsxtr@csname}%
      {}%
      {\csdef{\@glsxtr@csname}{0}}%
    }%
    {}%
  }%
```

Modifications to take into account the attributes that govern whether the entry should be unset.

```
  \let\@cgls@\@@cgls@
  \let\@cglspl@\@@cglspl@
  \let\@cGLS@\@@cGLS@
  \let\@cGlspl@\@@cGlspl@
  \let\@cGLS@\@@cGLS@
  \let\@cGLSpl@\@@cGLSpl@
```

Write information to the aux file.

```
  \AtEndDocument{\@gls@write@entryunitcounts}%
  \renewcommand*{\@gls@entry@unitcount}[3]{%
    \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
    {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
    {%
      \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
        \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
    }%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
    {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
    {%
      \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
      \fi
    }%
  }%
  \let\glsenableentryunitcount\relax
  \renewcommand*{\glsenableentrycount}{%
    \PackageError{glossaries-extra}{\string\glsenableentrycount\space
     can't be used with \string\glsenableentryunitcount}%
    {Use one or other but not both commands}%
  }%
}
```

```
\@onlypreamble\glsenableentryunitcount

entry@unitcount

\newcommand*{\@gls@entry@unitcount}[3]{}

ryunitcounts@do

\newcommand*{\@gls@write@entryunitcounts@do}[1]{%
  \immediate\write\@auxout
   {\string\@gls@entry@unitcount
     {\@glsentry}%
     {\@glsxtr@currunitcount{\@glsentry}{#1}%
     }%
     {#1}}%
}

entryunitcounts

\newcommand*{\@gls@write@entryunitcounts}{%
  \immediate\write\@auxout
    {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
  \count@=0\relax
  \forallglsentries{\@glsentry}{%
    \glshasattribute{\@glsentry}{unitcount}%
    {%
      \ifglsused{\@glsentry}%
      {%
        \forlistcsloop
          {\@gls@write@entryunitcounts@do}%
          {glo@\glsdetoklabel{\@glsentry}@unitlist}%
      }%
      {}%
      \advance\count@ by \@ne
    }%
    {}%
  }%
  \ifnum\count@=0
    \GlossariesExtraWarningNoLine{Entry counting has been enabled
     \MessageBreak with \string\glsenableentryunitcount\space but the
     \MessageBreak attribute ‘unitcount’ hasn’t
     \MessageBreak been assigned to any of the defined
     \MessageBreak entries}%
  \fi
}
```

tryUnitCounting The first argument is the list of categories, the second argument is the value of the entrycount attribute and the third is the counter name.

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

Enable entry counting:

```
  \glsenableentryunitcount
```

Redefine \gls etc:

```
  \renewcommand*{\gls}{\cgls}%
  \renewcommand*{\Gls}{\cGls}%
  \renewcommand*{\glspl}{\cglspl}%
  \renewcommand*{\Glspl}{\cGlspl}%
  \renewcommand*{\GLS}{\cGLS}%
  \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
  \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%
```

In case this command is used again:

```
  \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
  \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
    can't be used with \string\GlsXtrEnableEntryUnitCounting}%
   {Use one or other but not both commands}}%
}
```

tcountunsetattr

```
\newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
 \@for\@glsxtr@cat:=#1\do
 {%
   \ifdefempty{\@glsxtr@cat}{}%
   {%
     \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
     \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
   }%
 }%
}
```

1.3.6 Acronym Modifications

It's more consistent to use the abbreviation code for acronyms, but make some adjustments to allow for continued use of the glossaries package's custom acronym format. (For example, user may already have defined some acronym styles with \newacronymstyle which they would like to continue to use.) The original glossaries acronym code can be restored with \RestoreAcronyms, but adjust \SetGenericNewAcronym so that \newacronym adds the category.

nericNewAcronym

```
\renewcommand*{\SetGenericNewAcronym}{%
  \let\@Gls@entryname\@Gls@acrentryname
  \renewcommand{\newacronym}[4][]{%
    \ifdefempty{\@glsacronymlists}%
    {%
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
```

```
    }%
    {}%
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \protected@edef\@do@newglossaryentry{%
      \noexpand\newglossaryentry{\the\glslabeltok}%
      {%
        type=\acronymtype,%
        name={\expandonce{\acronymentry{##2}}},%
        sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
        text={\the\glsshorttok},%
        short={\the\glsshorttok},%
        shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
        long={\the\glslongtok},%
        longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
        category=acronym,
        \GenericAcronymFields,%
        \the\glskeylisttok
      }%
    }%
    \@do@newglossaryentry
  }%
  \renewcommand*{\acrfullfmt}[3]{%
    \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
  \renewcommand*{\Acrfullfmt}[3]{%
    \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
  \renewcommand*{\ACRfullfmt}[3]{%
    \glslink[##1]{##2}{%
      \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
  \renewcommand*{\acrfullplfmt}[3]{%
    \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
  \renewcommand*{\Acrfullplfmt}[3]{%
    \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
  \renewcommand*{\ACRfullplfmt}[3]{%
    \glslink[##1]{##2}{%
      \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%
  \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
  \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
  \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
  \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
}
```

This will cause a problem for glossaries that contain a mixture of acronyms and abbreviations, so redefine `\newacronym` to use the new abbreviation interface.

First save the original definitions:

```
\let\@glsxtr@org@setacronymstyle\setacronymstyle
```

```
\let\@glsxtr@org@newacronymstyle\newacronymstyle
```

msAbbreviations Make acronyms use the same interface as abbreviations. Note that `\newacronymstyle` has a different implementation to `\newabbrevationstyle` so disable `\newacronymstyle` and `\setacronymstyle`.

```
\newcommand*{\MakeAcronymsAbbreviations}{%
   \renewcommand*{\newacronym}[4][]{%
     \newabbreviation[type=\acronymtype,category=acronym,##1]{##2}{##3}{##4}%
   }%
   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
   \renewcommand*{\setacronymstyle}[1]{%
      \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
      unavailable.
      Use \string\setabbreviationstyle\space instead.
      The original acronym interface can be restored with
      \string\RestoreAcronyms}{}%
   }%
   \renewcommand*{\newacronymstyle}[1]{%
      \GlossariesExtraWarning{New acronym style `##1' won't be
      available unless you restore the original acronym interface with
      \string\RestoreAcronyms}%
      \@glsxtr@org@newacronymstyle{##1}%
   }%
}
```

Switch acronyms to abbreviations:

```
\MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

```
\newcommand*{\RestoreAcronyms}{%
  \SetGenericNewAcronym
  \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
  \renewcommand{\acronymfont}[1]{##1}%
  \let\setacronymstyle\@glsxtr@org@setacronymstyle
  \let\newacronymstyle\@glsxtr@org@newacronymstyle
```

Need to restore the original definition of `\@gls@link@checkfirsthyper` but `\glsxtrifwasfirstuse` still needs setting for the benefit of the post-link hook.

```
  \renewcommand*\@gls@link@checkfirsthyper{%
    \ifglsused{\glslabel}%
    {\let\glsxtrifwasfirstuse\@secondoftwo}
    {\let\glsxtrifwasfirstuse\@firstoftwo}%
    \@glsxtr@org@checkfirsthyper
  }
  \glssetcategoryattribute{acronym}{regular}{false}%
  \setacronymstyle{long-short}%
}
```

\glsacspace Allow the user to customise the maximum value.

```
\renewcommand*{\glsacspace}[1]{%
  \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
  \ifdim\dimen@<\glsacspacemax~\else\space\fi
}
```

\glsacspacemax Value used in the above.

```
\newcommand*{\glsacspacemax}{3em}
```

1.3.7 Indexing and Displaying Glossaries

From time-to-time users ask if they can have one glossary sorted normally and another sorted by definition or usage. With the base glossaries package this can only be achieved with the "noidx" commands (Option 1). This is an attempt to mix and match.

First we need a list of the glossaries that require makeindex/xindy.

r@reg@glosslist

```
\newcommand*{\@glsxtr@reg@glosslist}{}
```

Save the original definition of \makeglossaries:

```
\let\@glsxtr@org@makeglossaries\makeglossaries
```

Redefine \makeglossaries to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application.

\makeglossaries

```
\renewcommand*{\makeglossaries}[1][]{%
 \ifblank{#1}%
 {\@glsxtr@org@makeglossaries}%
 {%
   \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}}
   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
```

Iterate through each supplied glossary type and activate it.

```
   \@for\@glo@type:=#1\do{%
    \ifdefempty{\@glo@type}{}{\@makeglossary{\@glo@type}}%
   }%
```

New glossaries must be created before \makeglossaries:

```
   \renewcommand*\newglossary[4][]{%
   \PackageError{glossaries}{New glossaries
```

```
   must be created before \string\makeglossaries}{You need
   to move \string\makeglossaries\space after all your
   \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
  \let\@makeglossary\relax
  \let\makeglossary\relax
  \let\makeglossaries\relax
```

Disable all commands that have no effect after `\makeglossaries`

```
  \@disable@onlypremakeg
```

Allow see key:

```
  \let\gls@checkseeallowed\relax
```

Suppress warning about no `\makeglossaries`

```
  \let\warn@nomakeglossaries\relax
  \def\warn@noprintglossary{%
    \GlossariesWarningNoLine{No \string\printglossary\space
      or \string\printglossaries\space
      found.^^J(Remove \string\makeglossaries\space if you don't want
      any glossaries.)^^JThis document will not have a glossary}%
  }%
```

Adjust display number list to check for type:

```
  \renewcommand*{\glsdisplaynumberlist}[1]{%
    \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
    {\@glsxtr@idx@displaynumberlist{##1}}%
    {\@glsxtr@noidx@displaynumberlist{##1}}%
  }%
```

Adjust entry list:

```
  \renewcommand*{\glsentrynumberlist}[1]{%
    \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
    {\@glsxtr@idx@entrynumberlist{##1}}%
    {\@glsxtr@noidx@entrynumberlist{##1}}%
  }%
```

Adjust number list loop

```
  \renewcommand*{\glsnumberlistloop}[2]{%
    \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
    {%
       \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
        not available for glossary `##1'}{}%
    }%
    {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
  }%
```

Only sanitize sort for normal indexing glossaries.

```
  \renewcommand*{\glsprestandardsort}[3]{%
    \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
    {%
```

```
      \glsdosanitizesort
    }%
    {%
      \ifglssanitizesort
       \@gls@noidx@sanitizesort
      \else
       \@gls@noidx@nosanitizesort
      \fi
    }%
  }%
```

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

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

Only activate sort key for glossaries that aren't listed in `#1` (glossary label is stored in `\@glo@type` but this defaults to `\glsdefaulttype` so some expansion is required).

```
  \renewcommand*{\@printgloss@setsort}{%
    \renewcommand*{\@glo@assign@sortkey}{%
      \edef\@glo@type{\@glo@type}%
      \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glsxtr@reg@glosslist}%
      {%
        \@@glo@no@assign@sortkey
      }%
      {%
        \@@glo@assign@sortkey
      }%
    }%
    \def\@glo@sorttype{\@glo@default@sorttype}%
  }%
```

Check automake setting:

```
    \ifglsautomake
      \renewcommand*{\@gls@doautomake}{%
        \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
          \ifdefempty{\@gls@type}{}{\@gls@automake{\@gls@type}}%
        }%
      }%
    \fi
 }%
}
```

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

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```
\renewcommand{\@print@glossary}{%
  \makeatletter
  \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
  \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
  {}%
  {\glsxtrNoGlossaryWarning{\@glo@type}}%
  \ifglsxindy
    \ifcsundef{@xdy@\@glo@type @language}%
    {%
      \edef\@do@auxoutstuff{%
        \noexpand\AtEndDocument{%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\providecommand\string\@xdylanguage[2]{}}%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
        }%
      }%
    }%
```

```
    {%
      \edef\@do@auxoutstuff{%
        \noexpand\AtEndDocument{%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\providecommand\string\@xdylanguage[2]{}}%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
              @language\endcsname}}%
        }%
      }%
    }%
    \@do@auxoutstuff
    \edef\@do@auxoutstuff{%
      \noexpand\AtEndDocument{%
         \noexpand\immediate\noexpand\write\@auxout{%
          \string\providecommand\string\@gls@codepage[2]{}}%
         \noexpand\immediate\noexpand\write\@auxout{%
          \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
      }%
    }%
    \@do@auxoutstuff
  \fi
  \renewcommand*{\@warn@nomakeglossaries}{%
    \GlossariesWarningNoLine{\string\makeglossaries\space
    hasn't been used,^^Jthe glossaries will not be updated}%
  }%
}
```

Setup the warning text to display if the external file for the given glossary is missing.

oGlsWarningHead Header message.

```
\newcommand{\GlsXtrNoGlsWarningHead}[2]{%
 This document is incomplete. The external file associated with
 the glossary '#1' (which should be called \texttt{#2})
 hasn't been created.%
}
```

rningEmptyStart No entries have been added to the glossary.

```
\newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
  This has probably happened because there are no entries defined
  in this glossary.%
}
```

arningEmptyMain The default "main" glossary is empty.

```
\newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
 If you don't want this glossary,
 add \texttt{nomain} to your package option list when you load
 \texttt{glossaries-extra.sty}. For example:%
}
```

ingEmptyNotMain A glossary that isn't the default "main" glossary is empty.

```
\newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
 Did you forget to use \texttt{type=#1} when you defined your
 entries? If you tried to load entries into this glossary with
 \texttt{\string\loadglsentries} did you remember to use
 \texttt{[#1]} as the optional argument? If you did, check that
 the definitions in the file you loaded all had the type set
 to \texttt{\string\glsdefaulttype}.%
}
```

arningCheckFile Advisory message to check the file contents.

```
\newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
  Check the contents of the file \texttt{#1}. If
  it's empty, that means you haven't indexed any of your entries in this
  glossary (using commands like \texttt{\string\gls} or
  \texttt{\string\glsadd}) so this list can't be generated.
  If the file isn't empty, the document build process hasn't been
  completed.%
}
```

WarningAutoMake Message when automake option has been used.

```
\newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
  You may need to rerun \LaTeX. If you already have, it may be that
  \TeX's shell escape doesn't allow you to run
  \ifglsxindy xindy\else makeindex\fi. Check the
  transcript file \texttt{\jobname.log}. If the shell escape is
  disabled, try one of the following:

  \begin{itemize}
    \item Run the external (Lua) application:

       \texttt{makeglossaries-lite.lua \string"\jobname\string"}

    \item Run the external (Perl) application:

       \texttt{makeglossaries \string"\jobname\string"}
  \end{itemize}

  Then rerun \LaTeX\ on this document.
  \GlossariesExtraWarning{Rerun required to build the
  glossary `#1' or check TeX's shell escape allows
  you to run \ifglsxindy xindy\else makeindex\fi}%
}
```

WarningMisMatch Mismatching `\makenoidxglossaries`.

```
\newcommand{\GlsXtrNoGlsWarningMisMatch}{%
  You need to either replace \texttt{\string\makenoidxglossaries}
  with \texttt{\string\makeglossaries} or replace
```

```
  \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
  \texttt{\string\printnoidxglossary}
  (or \texttt{\string\printnoidxglossaries}) and then rebuild
  this document.%
}
```

arningBuildInfo Build advice.

```
\newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
  Try one of the following:
  \begin{itemize}
    \item Add \texttt{automake} to your package option list when you load
          \texttt{glossaries-extra.sty}. For example:

          \texttt{\string\usepackage[automake]%
              \glsopenbrace glossaries-extra\glsclosebrace}

    \item Run the external (Lua) application:

          \texttt{makeglossaries-lite.lua \string"\jobname\string"}

    \item Run the external (Perl) application:

          \texttt{makeglossaries \string"\jobname\string"}
  \end{itemize}

  Then rerun \LaTeX\ on this document.%
}
```

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
}
```

Provide some commands to accompany the record option.

xtrresourcefile This is provided for the benefit of any external helper application.

```
\newcommand*{\glsxtrresourcefile}[2][]{%
  \protected@write\@auxout{}{\string\glsxtr@resource{#1}{#2}}%
  \InputIfFileExists{#2}{}%
  {%
    \GlossariesExtraWarning{No file `#2'}%
  }%
}
\@onlypreamble\glsxtrresourcefile
```

glsxtr@resource

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

ntunsrtglossary Similar to \printnoidxglossary but it displays all entries defined for the given glossary without sorting.

```
\newcommand*{\printunsrtglossary}[1][type=\glsdefaulttype]{%
  \@printglossary{#1}{\@print@unsrt@glossary}%
}
```

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

```
  \ifcsempty{glolist@\@glo@type}
  {%
     \GlossariesExtraWarning{No entries defined in glossary `\@glo@type'}%
  }%
  {%
    \begin{theglossary}%
    \glossaryheader
    \glsresetentrylist
    \def\@gls@currentlettergroup{}%
    \expandafter\@for\expandafter\glscurrententrylabel\expandafter
      :\expandafter=\csname glolist@\@glo@type\endcsname\do{%
      \ifdefempty{\glscurrententrylabel}
      {}%
      {\@gls@noidx@do\glscurrententrylabel}%
    }%
    \end{theglossary}%
  }%
  \glossarypostamble
}
```

1.4 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}%
  }
```

End of if 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}}}
```

End of else part

```
}
```

1.5 Categories

\glscategory Add a new storage key that can be used to indicate a category. The default category is general.

```
\glsaddstoragekey{category}{general}{\glscategory}
```

\glsifcategory Convenient shortcut to determine if an entry has the given category.

```
\newcommand{\glsifcategory}[4]{%
 \ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
}
```

Categories can have attributes.

ategoryattribute `\glssetcategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨value⟩}`

Set (or override if already set) an attribute for the given category.

```
\newcommand*{\glssetcategoryattribute}[3]{%
  \csdef{@glsxtr@categoryattr@@#1@#2}{#3}%
}
```

ategoryattribute `\glsgetcategoryattribute{⟨category⟩}{⟨attribute-label⟩}`

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
\newcommand*{\glsgetcategoryattribute}[2]{%
  \csuse{@glsxtr@categoryattr@@#1@#2}%
}
```

ategoryattribute `\glshascategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨true⟩}{⟨false⟩}`

Tests if the category has the given attribute set.

```
\newcommand*{\glshascategoryattribute}[4]{%
  \ifcsvoid{@glsxtr@categoryattr@@#1@#2}{#4}{#3}%
}
```

\glssetattribute `\glssetattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨value⟩}`

Short cut where the category label is obtained from the entry information.

```
\newcommand*{\glssetattribute}[3]{%
  \glssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
}
```

\glsgetattribute `\glsgetattribute{⟨entry label⟩}{⟨attribute-label⟩}`

Short cut where the category label is obtained from the entry information.

```
\newcommand*{\glsgetattribute}[2]{%
  \glsgetcategoryattribute{\glscategory{#1}}{#2}%
}
```

\glshasattribute `\glshasattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨true⟩}{⟨false⟩}`

Short cut to test if the given attribute has been set where the category label is obtained from the entry information.

```
\newcommand*{\glshasattribute}[4]{%
  \ifglsentryexists{#1}%
  {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
  {#4}%
}
```

ategoryattribute `\glsifcategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨value⟩}{⟨true part⟩}{⟨false part⟩}`

True if category has the attribute with the given value.

```
\newcommand{\glsifcategoryattribute}[5]{%
 \ifcsundef{@glsxtr@categoryattr@@#1@#2}%
 {#5}%
 {\ifcsstring{@glsxtr@categoryattr@@#1@#2}{#3}{#4}{#5}}%
}
```

\glsifattribute `\glsifattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨value⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if the given entry has a category with the given attribute set.

```
\newcommand{\glsifattribute}[5]{%
  \ifglsentryexists{#1}%
  {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
  {#5}%
}
```

Set attributes for the default general category:

```
\glssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
\glssetcategoryattribute{acronym}{regular}{true}
```

regularcategory Convenient shortcut to create add the regular attribute.

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

fregularcategory `\glsifregularcategory{⟨category⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```
\newcommand{\glsifregularcategory}[3]{%
  \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
}
```

tregularcategory `\glsifnotregularcategory{⟨category⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```
\newcommand{\glsifnotregularcategory}[3]{%
  \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
}
```

\glsifregular `\glsifregular{⟨entry label⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if an entry has a regular attribute set to true.

```
\newcommand{\glsifregular}[3]{%
  \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
}
```

\glsifnotregular `\glsifnotregular{⟨entry label⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if an entry has a regular attribute set to false.

```
\newcommand{\glsifnotregular}[3]{%
  \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
}
```

oreachincategory `\glsforeachincategory[⟨glossary labels⟩]{⟨category-label⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}`

Iterates through all entries in all the glossaries (or just those listed in ⟨*glossary labels*⟩) and does ⟨*body*⟩ if the category matches ⟨*category-label*⟩. The control sequences ⟨*glossary-cs*⟩ and ⟨*label-cs*⟩ may be used in ⟨*body*⟩ to access the glossary label and entry label for the current iteration.

```
\newcommand{\glsforeachincategory}[5][\@glo@types]{%
```

```
  \forallglossaries[#1]{#3}%
  {%
     \forglsentries[#3]{#4}%
     {%
       \glsifcategory{#4}{#2}{#5}{}%
     }%
  }%
}
```

achwithattribute

\glsforeachwithattribute[⟨*glossary labels*⟩]{⟨*attribute-label*⟩}{⟨*attribute-value*⟩}{⟨*glossary-cs*⟩}{⟨*label-cs*⟩}{⟨*body*⟩}

Iterates through all entries in all the glossaries (or just those listed in ⟨*glossary labels*⟩) and does ⟨*body*⟩ if the category attribute ⟨*attribute-label*⟩ matches ⟨*attribute-value*⟩. The control sequences ⟨*glossary-cs*⟩ and ⟨*label-cs*⟩ may be used in ⟨*body*⟩ to access the glossary label and entry label for the current iteration.

```
\newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
  \forallglossaries[#1]{#4}%
  {%
     \forglsentries[#4]{#5}%
     {%
       \glsifattribute{#5}{#2}{#3}{#6}{}%
     }%
  }%
}
```

If \newterm has been defined, redefine it so that it automatically sets the category label to index and add \glsxtrpostdescription.

```
\ifdef\newterm
{%
```

\newterm

```
  \renewcommand*{\newterm}[2][]{%
    \newglossaryentry{#2}%
    {type={index},category=index,name={#2},%
     description={\glsxtrpostdescription\nopostdesc},#1}%
  }
```

Indexed terms are regular by default.

```
  \glssetcategoryattribute{index}{regular}{true}
```

trpostdescindex

```
  \newcommand*{\glsxtrpostdescindex}{}

}
{}
```

If the symbols package option was used, define a similar command for symbols, but set the default sort to the label rather than the name as the symbols will typically contain commands that will confuse makeindex and xindy.

```
\ifdef\printsymbols
{%
```

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

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

Symbols are regular by default.

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

rpostdescsymbol

```
  \newcommand*{\glsxtrpostdescsymbol}{}

}
{}
```

Similar for the numbers option.

```
\ifdef\printnumbers
{%
```

glsxtrnewnumber

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

Numbers are regular by default.

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

rpostdescnumber

```
  \newcommand*{\glsxtrpostdescnumber}{}

}
{}
```

sxtrsetcategory Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
\newcommand*{\glsxtrsetcategory}[2]{%
  \@for\@glsxtr@label:=#1\do
  {%
    \glsfieldxdef{\@glsxtr@label}{category}{#2}%
  }%
}
```

tcategoryforall Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
\newcommand*{\glsxtrsetcategoryforall}[2]{%
  \forallglossaries[#1]{\@glsxtr@type}{%
    \forglsentries[\@glsxtr@type]{\@glsxtr@label}%
    {%
      \glsfieldxdef{\@glsxtr@label}{category}{#2}%
    }%
  }%
}
```

trfieldtitlecase `\glsxtrfieldtitlecase{⟨label⟩}{⟨field⟩}`

Apply title casing to the contents of the given field.

```
\newcommand*{\glsxtrfieldtitlecase}[2]{%
  \expandafter\glsxtrfieldtitlecasecs\expandafter
    {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
}
```

ieldtitlecasecs The command used by `\glsxtrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```
\newcommand*{\glsxtrfieldtitlecasecs}[1]{\xcapitalisewords{#1}}
```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

\glossentrydesc If the glossdesc attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\glossentrydesc}[1]{%
    \glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossdescfont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossdescfont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          `\@glsxtr@attrval' supplied in glossdescfont attribute
```

```
          for entry ‘#1’. Ignoring}%
          \let\@glsxtr@glossdescfont\@firstofone
        }%
      }%
      {\let\@glsxtr@glossdescfont\@firstofone}%
      \glsifattribute{#1}{glossdesc}{firstuc}%
      {%
        \@glsxtr@glossdescfont{\Glsaccessdesc{#1}}%
      }%
      {%
        \glsifattribute{#1}{glossdesc}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \glsdescriptionaccessdisplay
          {%
            \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
          }%
          {#1}%
        }%
        {%
          \@glsxtr@glossdescfont{\glsaccessdesc{#1}}%
        }%
      }%
    }%
  }
}
{
  \renewcommand*{\glossentrydesc}[1]{%
    \glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glshasattribute{#1}{glossdescfont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossdescfont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          ‘\@glsxtr@attrval’ supplied in glossdescfont attribute
          for entry ‘#1’. Ignoring}%
          \let\@glsxtr@glossdescfont\@firstofone
        }%
      }%
      {\let\@glsxtr@glossdescfont\@firstofone}%
      \glsifattribute{#1}{glossdesc}{firstuc}%
      {%
        \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
```

```
      }%
      {%
        \glsifattribute{#1}{glossdesc}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
        }%
        {%
          \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
        }%
      }%
    }%
  }
}
```

\glossentryname If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossnamefont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
          for entry ‘#1’. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \glsifattribute{#1}{glossname}{firstuc}%
      {%
        \glsnameaccessdisplay
        {%
          \@glsxtr@glossnamefont{\Glsentryname{#1}}%
        }%
        {#1}%
      }%
      {%
        \glsifattribute{#1}{glossname}{title}%
```

```
        {%
          \@glsxtr@do@titlecaps@warn
          \glsnameaccessdisplay
          {%
            \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
          }%
          {#1}%
        }%
        {%
          \glsifattribute{#1}{glossname}{uc}%
          {%
            \glsnameaccessdisplay
            {%
```

Hide the label from the upper-casing command.

```
               \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
               \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
            }%
            {#1}%
          }%
          {%
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \glsnameaccessdisplay
            {%
              \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
            }%
            {#1}%
          }%
        }%
      }%
```

Do post-name hook:

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
{
  \renewcommand*{\glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glshasattribute{#1}{glossnamefont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
```

```
          `\@glsxtr@attrval' supplied in glossnamefont attribute
          for entry `#1'. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \glsifattribute{#1}{glossname}{firstuc}%
      {%
        \@glsxtr@glossnamefont{\Glsentryname{#1}}%
      }%
      {%
        \glsifattribute{#1}{glossname}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
        }%
        {%
          \glsifattribute{#1}{glossname}{uc}%
          {%
```

Hide the label from the upper-casing command.

```
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
          }%
          {%
```

This little trick is used by glossaries to allow the user to redefine \glsnamefont to use \makefirstuc. Support it even though they can now use the firstuc attribute.

```
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
          }%
        }%
      }%
```

Do post-name hook.

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
```

\Glossentryname Redefine to set the abbreviation format and accessibility support.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\Glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossnamefont}%
      {%
```

```
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          `\@glsxtr@attrval' supplied in glossnamefont attribute
          for entry `#1'. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \glsnameaccessdisplay
      {%
        \@glsxtr@glossnamefont{\Glsentryname{#1}}%
      }%
      {#1}%
```

Do post-name hook:

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
{
  \renewcommand*{\Glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glshasattribute{#1}{glossnamefont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          `\@glsxtr@attrval' supplied in glossnamefont attribute
          for entry `#1'. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \@glsxtr@glossnamefont{\Glsentryname{#1}}%
```

Do post-name hook:

```
      \glsxtrpostnamehook{#1}%
    }%
  }
```

```
}
```

Provide a convenient way to also index the entries using the standard \index mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

xtrpostnamehook Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```
\newcommand*{\glsxtrpostnamehook}[1]{%
  \def\@glsnumberformat{glsnumberformat}%
  \glsxtrdoautoindexname{#1}{indexname}%
```

Allow categories to hook in here.

```
  \csuse{glsxtrpostname\glscategory{\glscurrententrylabel}}%
}
```

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
      \ifdefstring{\@glsnumberformat}{glsnumberformat}{}%
      {\let\@glsxtr@attrval\@glsnumberformat}%
    \fi
    \ifdefstring{\@glsxtr@attrval}{true}%
    {}%
    {\eappto\@glo@name{\@glsxtr@autoindex@encap\@glsxtr@attrval}}%
    \expandafter\index\expandafter{\@glo@name}%
  }%
  {}%
}
```

toindex@setname Assign \@glo@name for use with indexname attribute.

```
\newcommand*{\@glsxtr@autoindex@setname}[1]{%
  \def\@glo@name{\string\glsentryname{#1}}%
  \glsletentryfield{\@glo@sort}{#1}{sort}%
  \@gls@checkmkidxchars\@glo@sort
  \@glsxtr@autoindex@doextra@esc\@glo@sort
  \epreto\@glo@name{\@glo@sort\@glsxtr@autoindex@at}%
}
```

dex@doextra@esc

```
\newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@esc\@gls@quotechar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
        \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape actual character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@at\@gls@actualchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
        \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
```

```
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape level character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@level\@gls@levelchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
        \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape encap character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@encap\@gls@encapchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
        \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
}
```

The user commands here have a preamble-only restriction to ensure they are set before required and also to reduce the chances of complications caused by babel's shorthands.

`tr@autoindex@at` Actual character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@at}{}
```

`trSetActualChar` Set the actual character.

```
\newcommand*{\GlsXtrSetActualChar}[1]{%
  \gdef\@glsxtr@autoindex@at{#1}%
  \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
  }%
}
\@onlypreamble\GlsXtrSetActualChar
\makeatother
\GlsXtrSetActualChar{@}
\makeatletter
```

`autoindex@encap` Encap character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@encap}{}
```

XtrSetEncapChar Set the encap character.

```
\newcommand*{\GlsXtrSetEncapChar}[1]{%
  \gdef\@glsxtr@autoindex@encap{#1}%
  \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetEncapChar{|}
\@onlypreamble\GlsXtrSetEncapChar
```

autoindex@level Level character for use with \index.

```
\newcommand*{\@glsxtr@autoindex@level}{}
```

XtrSetLevelChar Set the encap character.

```
\newcommand*{\GlsXtrSetLevelChar}[1]{%
  \gdef\@glsxtr@autoindex@level{#1}%
  \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetLevelChar{!}
\@onlypreamble\GlsXtrSetLevelChar
```

r@autoindex@esc Escape character for use with \index.

```
\newcommand*{\@glsxtr@autoindex@esc}{"}
```

lsXtrSetEscChar Set the escape character.

```
\newcommand*{\GlsXtrSetEscChar}[1]{%
  \gdef\@glsxtr@autoindex@esc{#1}%
  \def\@glsxtr@autoindex@escquote##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetEscChar{"}
\@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```
\ifdef\actualchar
 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
 {}
```

Quote character \quotechar:

```
\ifdef\quotechar
 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
 {}
```

Level character \levelchar:

```
\ifdef\levelchar
 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
 {}
```

Encap character \encapchar:

```
\ifdef\encapchar
 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
 {}
```

leto@endescspch

```
\def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

toindex@esc@spch

\@@glsxtr@autoindex@escspch{⟨*char*⟩}{⟨*cs*⟩}{⟨*pre*⟩}{⟨*mid*⟩}{⟨*post*⟩}

```
\newcommand*{\@@glsxtr@autoindex@escspch}[5]{%
  \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
  \toks@={#3}%
  \ifx\@nnil#3\relax
    \def\@@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
  \else
    \ifx\@nnil#4\relax
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
     \def\@@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
        #4#5\@glsxtr@endescspch}%
    \else
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
       \@glsxtr@autoindex@esc#1}%
      \def\@@glsxtr@checkspch{#2#5#1\@nnil#1\@glsxtr@endescspch}%
    \fi
  \fi
  \@@glsxtr@checkspch
}
```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\Glossentrydesc}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \Glsaccessdesc{#1}%
  }%
}
```

lossentrysymbol Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \glsaccesssymbol{#1}%
  }%
}
```

lossentrysymbol Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\Glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \Glsaccesssymbol{#1}%
  }%
}
```

Allow initials to be marked but only use the formatting for the tag in the glossary.

eInitialTagging Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```
\newcommand*{\GlsXtrEnableInitialTagging}{%
  \@ifstar\s@glsxtr@enabletagging\@glsxtr@enabletagging
}
\@onlypreamble\GlsXtrEnableInitialTagging
```

r@enabletagging Starred version undefines command.

```
\newcommand*{\s@glsxtr@enabletagging}[2]{%
  \undef#2%
  \@glsxtr@enabletagging{#1}{#2}%
}
```

r@enabletagging Internal command.

```
\newcommand*{\@glsxtr@enabletagging}[2]{%
```

Set attributes for categories given in the first argument.

```
  \@for\@glsxtr@cat:=#1\do
  {%
    \ifdefempty\@glsxtr@cat
    {}%
    {\glssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%
  }%
  \newrobustcmd*#2[1]{##1}%
  \def\@glsxtr@taggingcs{#2}%
  \renewcommand*\@glsxtr@activate@initialtagging{%
    \let#2\@glsxtr@tag
  }%
  \ifundef\@gls@preglossaryhook
  {\GlossariesExtraWarning{Initial tagging requires at least
    glossaries.sty v4.19 to work correctly}}%
  {}%
}
```

Are we using an old version of mfirstuc that has a bug in `\capitalisewords`? If so, patch it so we don't have a problem with a combination of tagging and title case.

fu@checkword@do If this command hasn't been defined, then we have pre v2.02 of mfirstuc

```
\ifundef\mfu@checkword@do
{
  \newcommand*{\mfu@checkword@do}[1]{%
   \ifdefstring{\mfu@checkword@arg}{#1}%
   {%
     \let\@mfu@domakefirstuc\@firstofone
     \listbreak
   }%
   {}%
  }
```

\mfu@checkword \capitalisewords was introduced in mfirstuc v1.06. If \mfu@checkword hasn't been defined mfirstuc is too old to support the title case attribute.

```
  \ifundef\mfu@checkword
  {
    \newcommand{\@glsxtr@do@titlecaps@warn}{%
     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
      support not available}%
```

One warning should suffice.

```
      \let\@glsxtr@do@titlecaps@warn\relax
    }
  }
  {
    \renewcommand*{\mfu@checkword}[1]{%
      \def\mfu@checkword@arg{#1}%
      \let\@mfu@domakefirstuc\makefirstuc
      \forlistloop\mfu@checkword@do\@mfu@nocaplist
    }
  }
}
{}% no patch required
```

@titlecaps@warn Do warning if title case not supported.

```
\newcommand*{\@glsxtr@do@titlecaps@warn}{}
```

@initialtagging Used in \printglossary but at least v4.19 of glossaries required.

```
\newcommand*\@glsxtr@activate@initialtagging{}
```

\@glsxtr@tag Definition of tagging command when used in glossary.

```
\newrobustcmd*{\@glsxtr@tag}[1]{%
  \glsifattribute{\glscurrententrylabel}{tagging}{true}%
  {\glsxtrtagfont{#1}}{#1}%
}
```

\glsxtrtagfont Used in the glossary.

```
\newcommand*{\glsxtrtagfont}[1]{\underline{#1}}
```

preglossaryhook This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

```
\ifdef\@gls@preglossaryhook
{
  \renewcommand*{\@gls@preglossaryhook}{%
    \@glsxtr@activate@initialtagging
    \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}{}
```

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}%
 {\glsxtrpostlink}%
}
```

\glsxtrpostlink

```
\newcommand*{\glsxtrpostlink}{%
 \csuse{glsxtrpostlink\glscategory{\glslabel}}%
}
```

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(\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(\glsaccesssymbol{\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

```
3838 \def\@glsxtr@notfoundinlist#1{\@secondoftwo}
```

glsxtrdopostpunc

\glsxtrdopostpunc{⟨*code*⟩}

If this is followed be a punctuation character, do ⟨*code*⟩ after the character otherwise do ⟨*code*⟩ before whatever comes next.

```
3839 \newcommand{\glsxtrdopostpunc}[1]{%
3840   \glsxtrifnextpunc{\@glsxtr@swaptwo{#1}}{#1}%
3841 }
```

@glsxtr@swaptwo

```
3842 \newcommand{\@glsxtr@swaptwo}[2]{#2#1}
```

1.6 Abbreviations

The "acronym" code from glossaries is misnamed as it's more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there's a style for the given category, apply it.

```
3843 \define@key{glsxtrabbrv}{category}{%
3844  \edef\glscategorylabel{#1}%
3845  \ifcsdef{@glsabbrv@current@#1}%
3846  {%
```

Warning should already have been issued.

```
3847    \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
3848    \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
3849    \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
3850    \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
3851  }%
3852  {}%
3853 }
```

Save the short plural form. This may be needed before the entry is defined.

```
3854 \define@key{glsxtrabbrv}{shortplural}{%
3855   \def\@gls@shortpl{#1}%
3856 }
```

Similarly for the long plural form.

```
3857 \define@key{glsxtrabbrv}{longplural}{%
3858   \def\@gls@longpl{#1}%
3859 }
```

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%
}
```

\newabbreviation Define a new generic abbreviation.

```
\newcommand*{\newabbreviation}[4][]{%
  \glskeylisttok{#1}%
  \glslabeltok{#2}%
  \glsshorttok{#3}%
  \glslongtok{#4}%
```

Get the category.

```
  \def\glscategorylabel{abbreviation}%
  \glsxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
  \setkeys*{glsxtrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
  \def\@gls@longpl{#4\glspluralsuffix}%
```

Has the insertdots attribute been set?

```
\glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
{%
  \@glsxtr@insertdots\@gls@short{#3}%
  \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
  \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
  {%
    \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
      '\abbrvpluralsuffix}%
  }%
  {%
    \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
    {%
      \let\@gls@shortpl\@gls@short
    }%
    {%
      \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
        \abbrvpluralsuffix}%
    }%
  }%
}%
{%
```

insertdots not true.

```
  \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
  {%
    \def\@gls@shortpl{#3'\abbrvpluralsuffix}%
  }%
  {%
    \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
    {%
      \def\@gls@shortpl{#3}%
    }%
    {%
      \def\@gls@shortpl{#3\abbrvpluralsuffix}%
    }%
  }%
}%
```

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.

```
\setkeys*{glsxtrabbrv}[category]{#1}%
```

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}%
```

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,%
      \the\glskeylisttok
    }%
  }%
  \@do@newglossaryentry
  \GlsXtrPostNewAbbreviation
}
```
evpresetkeyhook Hook for extra stuff in \newabbreviation
```
\newcommand*{\glsxtrnewabbrevpresetkeyhook}[3]{}
```
NewAbbreviation Hook used by abbreviation styles.
```
\newcommand*{\GlsXtrPostNewAbbreviation}{}
```
bbreviationhook Hook for use with \newabbreviation.
```
\newcommand*{\newabbreviationhook}{}
```
reviationFields
```
\newcommand*{\CustomAbbreviationFields}{}
```
lsxtrfullformat Full format without case change.
```
\newcommand*{\glsxtrfullformat}[2]{%
  \glsfirstlongfont{\glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
  (\protect\glsfirstabbrvfont{\glsaccessshort{#1}})%
}
```
lsxtrfullformat Full format with case change.
```
\newcommand*{\Glsxtrfullformat}[2]{%
  \glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
  (\protect\glsfirstabbrvfont{\glsaccessshort{#1}})%
}
```
xtrfullplformat Plural full format without case change.
```
\newcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
  (\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}})%
}
```

xtrfullplformat Plural full format with case change.

```
\newcommand*{\Glsxtrfullplformat}[2]{%
  \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
  (\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}})%
}
```

\glsxtrfullsep Separator used by full format is a space by default. The argument is the entry's label.

```
\newcommand*{\glsxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with \glsentryfull (for example, first use suppresses the long form or uses a footnote).

nlinefullformat Full format without case change.

```
\newcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}
```

nlinefullformat Full format with case change.

```
\newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

xtrfullplformat Plural full format without case change.

```
\newcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}
```

inefullplformat Plural full format with case change.

```
\newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}
```

Redefine \glsentryfull etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the \glsxtrfull set of commands instead.

\glsentryfull

```
\renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}
```

\Glsentryfull

```
\renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

\glsentryfullpl

```
\renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}
```

\Glsentryfullpl

```
\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

sfirstabbrvfont Font changing command used for the abbreviation on first use or in the full format.

```
\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}
```

bbrvdefaultfont Font changing command used for the abbreviation on first use or in the full format.

```
\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}
```

\glsabbrvfont Font changing command used for the abbreviation on subsequent use.

```
\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}
```

bbrvdefaultfont

```
3976 \newcommand*{\glsabbrvdefaultfont}[1]{#1}
```

\glslongfont Font changing command used for the long form in commands like \glsxtrlong.

```
3977 \newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}
```

longdefaultfont Default font changing command used for the long form in commands like \glsxtrlong.

```
3978 \newcommand*{\glslongdefaultfont}[1]{#1}
```

lsfirstlongfont Font changing command used for the long form on first use or in the full format.

```
3979 \newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}
```

longdefaultfont

```
3980 \newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}
```

brvpluralsuffix Default plural suffix.

```
3981 \newcommand*{\abbrvpluralsuffix}{\glspluralsuffix}
```

\glsxtrfull Full form (no case-change).

```
3982 \newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}
3983 \newcommand*\ns@glsxtrfull[2][]{%
3984   \new@ifnextchar[{\@glsxtr@full{#1}{#2}}%
3985                   {\@glsxtr@full{#1}{#2}[]}%
3986 }
```

\@glsxtr@full Low-level macro:

```
3987 \def\@glsxtr@full#1#2[#3]{%
3988   \glsdoifexists{#2}%
3989   {%
3990     \glssetabbrvfmt{\glscategory{#2}}%
3991     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3992     \let\glsifplural\@secondoftwo
3993     \let\glscapscase\@firstofthree
3994     \let\glsinsert\@empty
3995     \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.

```
3996     \glsxtrsetupfulldefs
3997     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
3998   }%
3999   \glspostlinkhook
4000 }
```

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]{%
  \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]{%
  \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]{%
  \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]{%
  \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]{%
  \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]{%
  \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]{%
  \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]{%
  \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]{%
  \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]{%
  \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]{%
  \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]{%
```

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

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:

```
          \glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert
        }%
        {%
```

Subsequent plural form, make first letter upper case:

```
          \glsabbrvfont{\Glsaccessshortpl{\glslabel}}\glsinsert
        }%
        {%
```

Subsequent plural form, all caps:

```
          \mfirstucMakeUppercase
            {\glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert}%
        }%
      }%
      {%
```

Subsequent singular form

```
        \glscapscase
        {%
```

Subsequent singular form, don't adjust case:

```
          \glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert
        }%
        {%
```

Subsequent singular form, make first letter upper case:

```
          \glsabbrvfont{\Glsaccessshort{\glslabel}}\glsinsert
        }%
        {%
```

Subsequent singular form, all caps:

```
            \mfirstucMakeUppercase
              {\glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert}%
          }%
        }%
      }%
      {%
```

First use:

```
        \glsifplural
        {%
```

First use plural form:

```
          \glscapscase
          {%
```

First use plural form, don't adjust case:

```
            \glsxtrfullplformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use plural form, make first letter upper case:

```
            \Glsxtrfullplformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use plural form, all caps:

```
            \mfirstucMakeUppercase
              {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
          }%
        }%
        {%
```

First use singular form

```
          \glscapscase
          {%
```

First use singular form, don't adjust case:

```
            \glsxtrfullformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use singular form, make first letter upper case:

```
            \Glsxtrfullformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use singular form, all caps:

```
            \mfirstucMakeUppercase
             {\glsxtrfullformat{\glslabel}{\glsinsert}}%
          }%
        }%
      }%
```

```
  }%
  {%
```

User supplied text.

```
    \glscustomtext
  }%
}
```

1.6.1 Abbreviation Styles Setup

breviationstyle

```
\newcommand*{\setabbreviationstyle}[2][abbreviation]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#2}
  {%
    \PackageError{glossaries-extra}{Undefined abbreviation style `#2'}{}%
  }%
  {%
```

Have abbreviations already been defined for this category?

```
    \ifcsstring{@glsabbrv@current@#1}{#2}%
    {%
```

Style already set.

```
    }%
    {%
      \def\@glsxtr@dostylewarn{}%
      \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
      {%
        \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
          style has been switched \MessageBreak
          for category `#1', \MessageBreak
          but there have already been entries \MessageBreak
          defined for this category. Unwanted \MessageBreak
          side-effects may result}}%
        \@endfortrue
      }%
      \@glsxtr@dostylewarn
```

Set up the style for the given category.

```
      \csdef{@glsabbrv@current@#1}{#2}%
      \glsxtr@applyabbrvstyle{#2}%
    }%
  }%
}
```

applyabbrvstyle Apply the abbreviation style without existence check.

```
\newcommand*{\glsxtr@applyabbrvstyle}[1]{%
  \csuse{@glsabbrv@dispstyle@setup@#1}%
  \csuse{@glsabbrv@dispstyle@fmts@#1}%
}
```

r@applyabbrvfmt Only apply the style formats.

```
\newcommand*{\glsxtr@applyabbrvfmt}[1]{%
  \csuse{@glsabbrv@dispstyle@fmts@#1}%
}
```

breviationstyle This is different from \newacronymstyle. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```
\newcommand*{\newabbreviationstyle}[3]{%
  \ifcsdef{@glsabbrv@dispstyle@setup@#1}
  {%
    \PackageError{glossaries-extra}{Abbreviation style `#1' already
     defined}{}%
  }%
  {%
    \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
     #2}%
    \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
    \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
    \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
    \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
    \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
    #3}%
  }%
}
```

breviationstyle

```
\newcommand*{\renewabbreviationstyle}[3]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#1}
  {%
    \PackageError{glossaries-extra}{Abbreviation style `#1' not defined}{}%
  }%
  {%
    \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
     #2}%
    \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
    \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
    \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
    \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
    \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
```

```
    #3}%
  }%
}
```

breviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
\newcommand*{\letabbreviationstyle}[2]{%
  \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
  \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
}
```

ecated@abbrstyle `\@glsxtr@deprecated@abbrstyle{⟨old-name⟩}{⟨new-name⟩}`

Define a synonym for a deprecated abbreviation style.

```
\newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
  \csdef{@glsabbrv@dispstyle@setup@#1}{%
    \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
    \csuse{@glsabbrv@dispstyle@setup@#2}%
  }%
  \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
}
```

ecatedAbbrStyle Generate warning for deprecated style use.

```
\newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
  \GlossariesExtraWarning{Deprecated abbreviation style name `#1',
  use `#2' instead}%
}
```

eAbbrStyleSetup

```
\newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
  {%
     \PackageError{glossaries-extra}%
     {Unknown abbreviation style definitions `#1'}{}%
  }%
  {%
     \csname @glsabbrv@dispstyle@setup@#1\endcsname
  }%
}
```

seAbbrStyleFmts

```
\newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
  \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
  {%
     \PackageError{glossaries-extra}%
     {Unknown abbreviation style formats `#1'}{}%
```

```
  }%
  {%
     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
  }%
}
```

1.6.2 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
```

long-short

```
\newabbreviationstyle{long-short}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
    plural={\protect\glsabbvfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

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

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

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

```
  \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}%
    (\glsfirstabbrvfont{\glsaccessshort{##1}})%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstabbrvfont{\glsaccessshort{##1}})%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
  }%
}
```

Set this as the default style for general abbreviations:

```
\setabbreviationstyle{long-short}
```

ngshortdescsort

```
\newcommand*{\glsxtrlongshortdescsort}{\the\glslongtok\space(\the\glsshorttok)}
```

long-short-desc User supplies description. The long form is included in the name.

```
\newabbreviationstyle{long-short-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsxtrfullformat{\the\glslabeltok}{}},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\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}%
}
```

`short-long` Short form followed by long form in parenthesis on first use.

```
\newabbreviationstyle{short-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstlongfont{\the\glslongtok})},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstlongfont{\the\glslongpltok})},%
    plural={\protect\glsabbvfont{\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}{\glspluralsuffix}%
  \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}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
}
```

short-long-desc User supplies description. The long form is included in the name.

```
\newabbreviationstyle{short-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsxtrfullformat{\the\glslabeltok}{}},
    sort={\the\glsshorttok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstlongfont{\the\glslongtok})},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstlongfont{\the\glslongpltok})},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

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

```
{%
  \GlsXtrUseAbbrStyleFmts{short-long}%
}
```

ongfootnotefont Only used by the “footnote” styles.

```
\newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

ongfootnotefont Only used by the “footnote” styles.

```
\newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

xtrabbrvfootnote `\glsxtrabbrvfootnote{⟨label⟩}{⟨long⟩}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument ⟨*long*⟩ includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```
\newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}
```

footnote Short form followed by long form in footnote on first use.

```
\newabbreviationstyle{footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%

    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
    plural={\protect\glsabbvfont{\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}{\glspluralsuffix}%
\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}%
  (\glsfirstlongfootnotefont{\glsaccesslong{##1}})%
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  (\glsfirstlongfootnotefont{\glsaccesslong{##1}})%
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
```

```
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
  }%
}
```

short-footnote

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

postfootnote

Similar to the above but the footnote is placed afterwards, outside the link. This avoids nested links and can also move the footnote marker after any following punctuation mark. Pre v1.07 included `\footnote` in the first keys, which was incorrect as it caused duplicate footnotes.

```
\newabbreviationstyle{postfootnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
    plural={\protect\glsabbvfont{\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}{\glspluralsuffix}%
  \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}%
    (\glsfirstlongfootnotefont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfootnotefont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
  }%
}
```

rt-postfootnote

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

short Provide a style that only displays the short form on first use, but the short and long form can be displayed with the "full" commands that use the inline format. If the user supplies a description, the long form won't be displayed in the predefined glossary styles, but the post description hook can be employed to automatically insert it.

```
\newabbreviationstyle{short}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    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}{\glspluralsuffix}%
  \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}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\Glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\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}
```

short-desc The user must supply the description in this style. The long form is added to the name. The short style (possibly with the post-description hooks set) might be a better option.

```
\newabbreviationstyle{short-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsxtrinlinefullformat{\the\glslabeltok}{}},
    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}{\glspluralsuffix}%
  \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}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\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` Provide a style that only displays the long form, but the long and short form can be displayed with the "full" commands that use the inline format. The predefined glossary styles won't

show the short form. The user must supply a description for this style.

```
\newabbreviationstyle{long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\protect\glsfirstlongfont{\the\glslongtok}},
    sort={\the\glslongtok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
    text={\the\glslongtok},
    plural={\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}{\glspluralsuffix}%
  \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 long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\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}
```

`long` It doesn't really make a great deal of sense to have a long-only style that doesn't have a description, but the best course of action here is to use the short form as the name and the long form as the description.

```
\newabbreviationstyle{long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
    text={\the\glslongtok},
    plural={\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}
```

1.6.3 Predefined Styles (Small Capitals)

These styles use:

`\glsxtrscfont`

```
\newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

sxtrfirstscfont

```
5026 \newcommand*{\glsxtrfirstscfont}[1]{\glsxtrscfont{#1}}
```

and for the default short form suffix:

\glsxtrscsuffix

```
5027 \newcommand*{\glsxtrscsuffix}{\glstextup{\glspluralsuffix}}
```

long-short-sc

```
5028 \newabbreviationstyle{long-short-sc}%
5029 {%
5030   \GlsXtrUseAbbrStyleSetup{long-short}%
5031 }%
5032 {%
```

Mostly as long-short style:

```
5033   \GlsXtrUseAbbrStyleFmts{long-short}%
```

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

```
5034   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
5035   \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
5036   \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
5037 }
```

g-short-sc-desc

```
5038 \newabbreviationstyle{long-short-sc-desc}%
5039 {%
5040   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
5041 }%
5042 {%
```

Mostly as long-short-desc style:

```
5043   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
```

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

```
5044   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
5045   \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
5046   \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
5047 }
```

Now the short (long) version

```
5048 \newabbreviationstyle{short-sc-long}%
5049 {%
5050   \GlsXtrUseAbbrStyleSetup{short-long}%
5051 }%
5052 {%
```

Mostly as short-long style:

```
5053   \GlsXtrUseAbbrStyleFmts{short-long}%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
}
```

As before but user provides description

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

Mostly as short-long-desc style:

```
  \GlsXtrUseAbbrStyleFmts{short-long-desc}%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
}
```

short-sc

```
\newabbreviationstyle{short-sc}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong}%
}%
{%
```

Mostly as short style:

```
  \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
}
```

short-sc-nolong

```
\letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

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

Mostly as short style:

```
  \GlsXtrUseAbbrStyleFmts{short-desc}%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
}
```

-sc-nolong-desc

```
\letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
\newabbreviationstyle{long-noshort-sc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort}%
}%
{%
```

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{long-noshort}%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
}
```

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}%
}%
{%
```

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
}
```

long-desc-sc Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

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

Mostly as long style:

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

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

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
}
```

footnote-sc

Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```

sc-postfootnote

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

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstscfont{##1}}%
}
```

postfootnote-sc

Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}
```

1.6.4 Predefined Styles (Fake Small Capitals)

These styles require the relsize package, which must be loaded by the user. These styles all use:

\glsxtrsmfont

```
\newcommand*{\glsxtrsmfont}[1]{\textsmaller{#1}}
```

sxtrfirstsmfont

```
\newcommand*{\glsxtrfirstsmfont}[1]{\glsxtrsmfont{#1}}
```

and for the default short form suffix:

```
\glsxtrsmsuffix
5136 \newcommand*{\glsxtrsmsuffix}{\glspluralsuffix}

  long-short-sm
5137 \newabbreviationstyle{long-short-sm}%
5138 {%
5139   \GlsXtrUseAbbrStyleSetup{long-short}%
5140 }%
5141 {%
```

Mostly as long-short style:

```
5142   \GlsXtrUseAbbrStyleFmts{long-short}%
5143   \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
5144   \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
5145   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
5146 }

g-short-sm-desc
5147 \newabbreviationstyle{long-short-sm-desc}%
5148 {%
5149   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
5150 }%
5151 {%
```

Mostly as long-short-desc style:

```
5152   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
5153   \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
5154   \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
5155   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
5156 }
```

```
  short-sm-long
```

Now the short (long) version

```
5157 \newabbreviationstyle{short-sm-long}%
5158 {%
5159   \GlsXtrUseAbbrStyleSetup{short-long}%
5160 }%
5161 {%
```

Mostly as short-long style:

```
5162   \GlsXtrUseAbbrStyleFmts{short-long}%
5163   \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
5164   \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
5165   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
5166 }
```

```
rt-sm-long-desc
```

As before but user provides description

```
5167 \newabbreviationstyle{short-sm-long-desc}%
5168 {%
5169   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
5170 }%
5171 {%
```

Mostly as short-long-desc style:

```
  \GlsXtrUseAbbrStyleFmts{short-long-desc}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
}
```

short-sm

```
\newabbreviationstyle{short-sm}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong}%
}%
{%
```

Mostly as short style:

```
  \GlsXtrUseAbbrStyleFmts{short-nolong}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
}
```

short-sm-nolong

```
\letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
\newabbreviationstyle{short-sm-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
}%
{%
```

Mostly as short style:

```
  \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
}
```

-sm-nolong-desc

```
\letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}
```

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}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort}%
}%
{%
```

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{long-noshort}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
}
```

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}%
}%
{%
```

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
}
```

long-desc-sm Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}
```

ort-sm-footnote

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

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{short-footnote}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
}
```

footnote-sm Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}
```

sm-postfootnote

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

```
}%
{%
```

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
  \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
}
```

postfootnote-sm Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}
```

1.6.5 Predefined Styles (Emphasized)

These styles use \emph for the short form.

\glsabbrvemfont

```
\newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%
```

irstabbrvemfont

```
\newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%
```

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

```
\newabbreviationstyle{long-short-em}%
{%
  \GlsXtrUseAbbrStyleSetup{long-short}%
}%
{%
```

Mostly as long-short style:

```
  \GlsXtrUseAbbrStyleFmts{long-short}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
}
```

g-short-em-desc

```
\newabbreviationstyle{long-short-em-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-short-desc}%
}%
{%
```

Mostly as long-short-desc style:

```
  \GlsXtrUseAbbrStyleFmts{long-short-desc}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
}
```

ong-em-short-em

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

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
    plural={\protect\glsabbvfont{\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}%
    }%
    {}%
  }%
}%
{%
```

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

```
  \GlsXtrUseAbbrStyleFmts{long-short}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
}
```

m-short-em-desc

```
\newabbreviationstyle{long-em-short-em-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-short-desc}%
}%
{%
```

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

```
  \GlsXtrUseAbbrStyleFmts{long-short-desc}%
```

```
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
}
```

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

```
\newabbreviationstyle{short-em-long}%
{%
  \GlsXtrUseAbbrStyleSetup{short-long}%
}%
{%
```

Mostly as short-long style:

```
  \GlsXtrUseAbbrStyleFmts{short-long}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
}
```

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

```
\newabbreviationstyle{short-em-long-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{short-long-desc}%
}%
{%
```

Mostly as short-long-desc style:

```
  \GlsXtrUseAbbrStyleFmts{short-long-desc}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
}
```

hort-em-long-em

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

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    description={\protect\glslongemfont{\the\glslongtok}},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstlongfont{\the\glslongtok})},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstlongfont{\the\glslongpltok})},%
    plural={\protect\glsabbvfont{\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.

```
  \GlsXtrUseAbbrStyleFmts{short-long}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
}
```

em-long-em-desc

```
\newabbreviationstyle{short-em-long-em-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{short-long-desc}%
}%
{%
```

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

```
  \GlsXtrUseAbbrStyleFmts{short-long-desc}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
}
```

short-em

```
\newabbreviationstyle{short-em}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong}%
}%
{%
```

Mostly as short style:

```
  \GlsXtrUseAbbrStyleFmts{short-nolong}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
}
```

short-em-nolong

```
\letabbreviationstyle{short-em-nolong}{short-em}
```

short-em-desc

```
\newabbreviationstyle{short-em-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
}%
{%
```

Mostly as short style:

```
  \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
}
```

-em-nolong-desc

```
\letabbreviationstyle{short-em-nolong-desc}{short-em-desc}
```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```
\newabbreviationstyle{long-noshort-em}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort}%
}%
{%
```

Mostly as long-noshort style:

```
  \GlsXtrUseAbbrStyleFmts{long-noshort}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
}
```

long-em Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-em}{long-noshort-em}
```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```
\newabbreviationstyle{long-em-noshort-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
    text={\the\glslongtok},
    plural={\the\glslongpltok},%
    description={\protect\glslongemfont{\the\glslongtok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

Mostly as long-noshort style:

```
  \GlsXtrUseAbbrStyleFmts{long-noshort}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
}
```

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}%
}%
{%
```

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
}
```

long-desc-em Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like \glsshort. The long form is emphasized.

```
\newabbreviationstyle{long-em-noshort-em-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
}%
{%
```

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
}
```

ort-em-footnote

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

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{short-footnote}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
}
```

footnote-em Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

em-postfootnote

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

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
}
```

postfootnote-em Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}
```

1.6.6 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}%
    (#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{})%
  }
}
{
  \newcommand*{\glsxtruserparen}[2]{%
    \glsxtrfullsep{#2}%
    (#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{})%
  }
}
```

Font used for short form:

lsabbrvuserfont

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

Font used for short form on first use:

stabbrvuserfont

```
\newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

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

Font used for long form on first use:

rstlonguserfont

```
\newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
\newcommand*{\glsxtrusersuffix}{\glspluralsuffix}
```

long-short-user

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

\glslonguserfont is used in the description since \glsdesc doesn't set the style.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}{\the\glslabeltok}
    plural={\protect\glsabbvfont{\the\glsshortpltok}},%
    description={\protect\glslonguserfont{\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}{\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]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
  }%
}
```

short-user-desc

```
\newabbreviationstyle{long-short-user-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-short-desc}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-short-user}%
}
```

short-long-user

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

`\glslonguserfont` is used in the description since `\glsdesc` doesn't set the style.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    description={\protect\glslonguserfont{\the\glslongtok}},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtruserparen{\protect\glslonguserfont{\the\glslongtok}}{\the\glslabeltok}},%
```

```
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtruserparen{\protect\glslonguserfont{\the\glslongpltok}}{\the\glslabeltok}},%
    plural={\protect\glsabbvfont{\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]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
  }%
}
```

-long-user-desc

```
\newabbreviationstyle{short-long-user-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{short-long-desc}%
}%
```

```
{%
  \GlsXtrUseAbbrStyleFmts{short-long-user}%
}
```

1.7 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
}
```

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
}
```

\glsxtrifinmark

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

\@glsxtrinmark

```
\newrobustcmd*{\@glsxtrinmark}{%
  \let\glsxtrifinmark\@firstoftwo
}
```

glsxtrnotinmark

```
\newrobustcmd*{\@glsxtrnotinmark}{%
  \let\glsxtrifinmark\@secondoftwo
}
```

\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@glsxtrtitleshort\glsxtrtitleshort
  \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
  \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
  \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
  \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\glsxtrtitleshort\glsxtrheadshort
  \let\glsxtrtitleshortpl\glsxtrheadshortpl
  \let\Glsxtrtitleshort\Glsxtrheadshort
  \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
  \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\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
  \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
  \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
  \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
  \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
  \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
  \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
  \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
  \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
  \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
  \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural
  \let\Glsxtrtitlefirstplural\@glsxtr@org@Glsxtrtitlefirstplural
  \let\glsxtrtitlelong\@glsxtr@org@glsxtrtitlelong
  \let\glsxtrtitlelongpl\@glsxtr@org@glsxtrtitlelongpl
```

```
  \let\Glsxtrtitlelong\@glsxtr@org@Glsxtrtitlelong
  \let\Glsxtrtitlelongpl\@glsxtr@org@Glsxtrtitlelongpl
  \let\glsxtrtitlefull\@glsxtr@org@glsxtrtitlefull
  \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
  \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
  \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
}
```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

glsxtrheadshort Command used to display short form in the page header.

```
\newcommand*{\glsxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshort[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrshort[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitleshort}[1]{%
  \glsxtrshort[noindex,hyper=false]{#1}[]%
}
```

sxtrheadshortpl Command used to display plural short form in the page header. If you want the text converted to upper case, this needs to be redefined to use \GLSxtrshortpl instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadshortpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
      \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
   {%
      \glsxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

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

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

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

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshort[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrshort[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitleshort}[1]{%
  \Glsxtrshort[noindex,hyper=false]{#1}[]%
}
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadshortpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
  \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
}
```

`\glsxtrheadtext` As above but for the text value.

```
\newcommand*{\glsxtrheadtext}[1]{%
 \protect\NoCaseChange
```

```
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLStext[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glstext[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitletext}[1]{%
  \glstext[noindex,hyper=false]{#1}[]%
}
```

`\Glsxtrheadtext` First letter converted to upper case

```
\newcommand*{\Glsxtrheadtext}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLStext[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glstext[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitletext}[1]{%
  \Glstext[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadplural` As above but for the plural value.

```
\newcommand*{\glsxtrheadplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitleplural Command to display plural value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitleplural}[1]{%
  \glsplural[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadplural Convert first letter to upper case.

```
\newcommand*{\Glsxtrheadplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitleplural}[1]{%
  \Glsplural[noindex,hyper=false]{#1}[]%
}
```

glsxtrheadfirst As above but for the first value.

```
\newcommand*{\glsxtrheadfirst}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirst[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsfirst[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefirst}[1]{%
  \glsfirst[noindex,hyper=false]{#1}[]%
}
```

Glsxtrheadfirst First letter converted to upper case

```
\newcommand*{\Glsxtrheadfirst}[1]{%
 \protect\NoCaseChange
 {%
```

```
    \glsifattribute{#1}{headuc}{true}%
    {%
      \GLSfirst[noindex,hyper=false]{#1}[]%
    }%
    {%
      \Glsfirst[noindex,hyper=false]{#1}[]%
    }%
  }%
}
```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitlefirst}[1]{%
  \Glsfirst[noindex,hyper=false]{#1}[]%
}
```

headfirstplural As above but for the firstplural value.

```
\newcommand*{\glsxtrheadfirstplural}[1]{%
 \protect\NoCaseChange
 {%
    \glsifattribute{#1}{headuc}{true}%
    {%
      \GLSfirstplural[noindex,hyper=false]{#1}[]%
    }%
    {%
      \glsfirstplural[noindex,hyper=false]{#1}[]%
    }%
 }%
}
```

itlefirstplural Command to display firstplural value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefirstplural}[1]{%
  \glsfirstplural[noindex,hyper=false]{#1}[]%
}
```

headfirstplural First letter converted to upper case

```
\newcommand*{\Glsxtrheadfirstplural}[1]{%
 \protect\NoCaseChange
 {%
    \glsifattribute{#1}{headuc}{true}%
    {%
      \GLSfirstplural[noindex,hyper=false]{#1}[]%
    }%
    {%
      \Glsfirstplural[noindex,hyper=false]{#1}[]%
    }%
 }%
}
```

itlefirstplural Command to display first value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
  \Glsfirstplural[noindex,hyper=false]{#1}[]%
}
```

\glsxtrheadlong Command used to display long form in the page header.

```
\newcommand*{\glsxtrheadlong}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlong[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrlong[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlelong}[1]{%
  \glsxtrlong[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadlongpl Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use \GLSxtrlongpl instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadlongpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
      \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
   {%
      \glsxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlelongpl}[1]{%
  \glsxtrlongpl[noindex,hyper=false]{#1}[]%
}
```

\Glsxtrheadlong Command used to display long form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadlong}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlong[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrlong[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlelong}[1]{%
  \Glsxtrlong[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadlongpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
  \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
}
```

`\glsxtrheadfull` Command used to display full form in the page header.

```
\newcommand*{\glsxtrheadfull}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrfull[noindex,hyper=false]{#1}[]%
   }%
```

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

glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefull}[1]{%
  \glsxtrfull[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadfullpl Command used to display plural full form in the page header. If you want the text converted to upper case, this needs to be redefined to use \GLSxtrfullpl instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadfullpl}[1]{%
 \protect\NoCaseChange
 {%
    \glsifattribute{#1}{headuc}{true}%
    {%
       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
    }%
    {%
       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
    }%
 }%
}
```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefullpl}[1]{%
  \glsxtrfullpl[noindex,hyper=false]{#1}[]%
}
```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadfull}[1]{%
 \protect\NoCaseChange
 {%
    \glsifattribute{#1}{headuc}{true}%
    {%
      \GLSxtrfull[noindex,hyper=false]{#1}[]%
    }%
    {%
      \Glsxtrfull[noindex,hyper=false]{#1}[]%
    }%
 }%
}
```

Glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlefull}[1]{%
  \Glsxtrfull[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadfullpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
  \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
}
```

\glsfmtshort Provide a way of using the formatted short form in section headings. If hyperref has been loaded, use \texorpdfstring for convenience in PDF bookmarks.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtshort}[1]{%
    \texorpdfstring
      {\glsxtrtitleshort{#1}}%
      {\glsentryshort{#1}}%
  }
}
{
  \newcommand*{\glsfmtshort}[1]{%
   \glsxtrtitleshort{#1}}
}
```

Similarly for the plural version.

\glsfmtshortpl

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtshortpl}[1]{%
    \texorpdfstring
      {\glsxtrtitleshortpl{#1}}%
      {\glsentryshortpl{#1}}%
```

```
  }
}
{
  \newcommand*{\glsfmtshortpl}[1]{%
   \glsxtrtitleshortpl{#1}}
}
```

The case-changing version isn't suitable for PDF bookmarks, so the PDF alternative uses the non-case-changing version.

`\Glsfmtshort` Singular form (first letter uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtshort}[1]{%
    \texorpdfstring
      {\Glsxtrtitleshort{#1}}%
      {\glsentryshort{#1}}%
  }
}
{
  \newcommand*{\Glsfmtshort}[1]{%
   \Glsxtrtitleshort{#1}}
}
```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtshortpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitleshortpl{#1}}%
    {\glsentryshortpl{#1}}%
  }
}
{
  \newcommand*{\Glsfmtshortpl}[1]{%
   \Glsxtrtitleshortpl{#1}}
}
```

`\glsfmttext` As above but for the text value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmttext}[1]{%
    \texorpdfstring
    {\glsxtrtitletext{#1}}%
    {\glsentrytext{#1}}%
  }
}
{
  \newcommand*{\glsfmttext}[1]{%
```

```
    \glsxtrtitletext{#1}}
}
```

`\Glsfmttext` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmttext}[1]{%
    \texorpdfstring
    {\Glsxtrtitletext{#1}}%
    {\glsentrytext{#1}}%
  }
}
{
  \newcommand*{\Glsfmttext}[1]{%
   \Glsxtrtitletext{#1}}
}
```

`\glsfmtplural` As above but for the plural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtplural}[1]{%
    \texorpdfstring
    {\glsxtrtitleplural{#1}}%
    {\glsentryplural{#1}}%
  }
}
{
  \newcommand*{\glsfmtplural}[1]{%
   \glsxtrtitleplural{#1}}
}
```

`\Glsfmtplural` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtplural}[1]{%
    \texorpdfstring
    {\Glsxtrtitleplural{#1}}%
    {\glsentryplural{#1}}%
  }
}
{
  \newcommand*{\Glsfmtplural}[1]{%
   \Glsxtrtitleplural{#1}}
}
```

`\glsfmtfirst` As above but for the first value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfirst}[1]{%
```

```
    \texorpdfstring
    {\glsxtrtitlefirst{#1}}%
    {\glsentryfirst{#1}}%
  }
}
{
  \newcommand*{\glsfmtfirst}[1]{%
   \glsxtrtitlefirst{#1}}
}
```

\Glsfmtfirst First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfirst}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefirst{#1}}%
    {\glsentryfirst{#1}}%
  }
}
{
  \newcommand*{\Glsfmtfirst}[1]{%
   \Glsxtrtitlefirst{#1}}
}
```

\glsfmtfirstpl As above but for the firstplural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfirstpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlefirstplural{#1}}%
    {\glsentryfirstplural{#1}}%
  }
}
{
  \newcommand*{\glsfmtfirstpl}[1]{%
   \glsxtrtitlefirstplural{#1}}
}
```

\Glsfmtfirstpl First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfirstpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefirstplural{#1}}%
    {\glsentryfirstplural{#1}}%
  }
}
{
  \newcommand*{\Glsfmtfirstpl}[1]{%
```

```
    \Glsxtrtitlefirstplural{#1}}
}
```

\glsfmtlong As above but for the long value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtlong}[1]{%
    \texorpdfstring
    {\glsxtrtitlelong{#1}}%
    {\glsentrylong{#1}}%
  }
}
{
  \newcommand*{\glsfmtlong}[1]{%
   \glsxtrtitlelong{#1}}
}
```

\Glsfmtlong First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtlong}[1]{%
    \texorpdfstring
    {\Glsxtrtitlelong{#1}}%
    {\glsentrylong{#1}}%
  }
}
{
  \newcommand*{\Glsfmtlong}[1]{%
   \Glsxtrtitlelong{#1}}
}
```

\glsfmtlongpl As above but for the longplural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtlongpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlelongpl{#1}}%
    {\glsentrylongpl{#1}}%
  }
}
{
  \newcommand*{\glsfmtlongpl}[1]{%
   \glsxtrtitlelongpl{#1}}
}
```

\Glsfmtlongpl First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtlongpl}[1]{%
```

```
    \texorpdfstring
    {\Glsxtrtitlelongpl{#1}}%
    {\glsentrylongpl{#1}}%
  }
}
{
  \newcommand*{\Glsfmtlongpl}[1]{%
   \Glsxtrtitlelongpl{#1}}
}
```

\glsfmtfull In-line full format.

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

\Glsfmtfull First letter converted to upper case.

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

\glsfmtfullpl In-line full plural format.

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

```
    \glsxtrtitlefullpl{#1}}
}
```

\Glsfmtfullpl First letter converted to upper case.

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

1.8 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.)

```
\@ifpackageloaded{tracklang}
{%
  \AnyTrackedLanguages
  {%
    \ForEachTrackedDialect{\this@dialect}{%
      \IfTrackedLanguageFileExists{\this@dialect}%
      {glossariesxtr-}% prefix
      {.ldf}%
      {%
        \RequireGlossariesExtraLang{\CurrentTrackedTag}%
      }%
      {%
```

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

Load glossaries-extra-stylemods if required.

```
\@glsxtr@redefstyles
```

and set the style:

```
\@glsxtr@do@style
```

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}[2016/12/13 v1.08 (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}{}

\DeclareOption*{%
  \IfFileExists{glossary-\CurrentOption.sty}
   {\eappto\@glsxtr@loadstyles{%
      \noexpand\RequirePackage{glossary-\CurrentOption}}}%
   {\PackageError{glossaries-extra-styles}%
    {Unknown option `\CurrentOption'}{}}
}
```

Process the package options:

```
\ProcessOptions
```

Load the required packages:

```
\@glsxtr@loadstyles
```

Adjust the styles that the post description hook added, but only for styles that have already been defined. All the tree styles in glossary-tree include the post description hook, so they don't require adjustment. Similarly for glossary-mcols which builds on the tree styles.

In case we have an old version of glossaries:

ewglossarystyle

```
\providecommand{\renewglossarystyle}[2]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries}{Glossary style `#1' isn't already defined}{}%
```

```
  }%
  {%
    \csdef{@glsstyle@#1}{#2}%
  }%
}
```

2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the listdotted style need modifying.

```
\ifdef{\@glsstyle@listdotted}
{%
 \renewglossarystyle{listdotted}{%
   \setglossarystyle{list}%
   \renewcommand*{\glossentry}[2]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
      \glsentryitem{##1}%
      \glstarget{##1}{\glossentryname{##1}}%
      \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
      \glossentrydesc{##1}\glspostdescription}%
   \renewcommand*{\subglossentry}[3]{%
    \item[]\makebox[\glslistdottedwidth][l]{%
    \glssubentryitem{##2}%
    \glstarget{##2}{\glossentryname{##2}}%
    \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
    \glossentrydesc{##2}\glspostdescription}%
 }
}
{}
```

The sublistdotted style doesn't display the description for top-level entries. Sub-level entries use the listdottedstyle.

2.3 Longtable Styles

The three and four column styles require adjustment, but not the two column styles.

```
\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
    }%
    \renewcommand*{\glsgroupskip}{%
     \ifglsnogroupskip\else & &\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
    }%
    \renewcommand*{\glsgroupskip}{%
      \ifglsnogroupskip\else & & &\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, but not the two column styles.

```
\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
    }%
    \renewcommand*{\glsgroupskip}{%
      \ifglsnogroupskip\else & &\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
    }%
    \renewcommand*{\glsgroupskip}{%
      \ifglsnogroupskip\else & & &\tabularnewline\fi}%
  }
}
{}
```

2.5 Supertabular Styles

The three and four column styles require adjustment, but not the two column styles.

```
\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
    }%
    \renewcommand*{\glsgroupskip}{%
      \ifglsnogroupskip\else & &\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
    }%
    \renewcommand*{\glsgroupskip}{%
```

```
      \ifglsnogroupskip\else & & &\tabularnewline\fi}%
  }
}
{}
```

2.6 Super Ragged Styles

The three and four column styles require adjustment, but not the two column styles.

```
\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
    }%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
    &\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
    }%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else & &
     &\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}
}
{}
```

2.8 Tree Styles

The alttree style is redefined to make it easier to made minor adjustments.

```
\ifdef{\@glsstyle@alttree}
{%
```

Only redefine this style if it's already been defined.

mbolDescLocation `\glsxtralttreeSymbolDescLocation{⟨label⟩}{⟨location list⟩}`

Layout the symbol, description and location for top-level entries.

```
  \newcommand{\glsxtralttreeSymbolDescLocation}[2]{%
    {%
      \let\par\glsxtrAltTreePar
```

```
      \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
      \glossentrydesc{#1}\glspostdescription \space #2\par
    }%
  }
```

trAltTreeIndent Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```
\newlength\glsxtrAltTreeIndent
```

lsxtrAltTreePar Multi-paragraph descriptions need to keep the hanging indent.

```
\newcommand{\glsxtrAltTreePar}{%
  \@@par
  \glsxtrAltTreeSetHangIndent
  \setlength{\parindent}{\dimexpr\hangindent+\glsxtrAltTreeIndent}%
}
```

mbolDescLocation `\glsxtralttreeSubSymbolDescLocation{⟨level⟩}{⟨label⟩}{⟨location list⟩}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```
\newcommand{\glsxtralttreeSubSymbolDescLocation}[3]{%
  \glsxtralttreeSymbolDescLocation{#2}{#3}%
}
```

trtreetopindent The original style has to keep computing the width of the name at each entry. This register allows the style to compute it once for the top-level at the start of the glossary.

```
\newlength\glsxtrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
\newcommand*{\glsxtralttreeInit}{%
  \settowidth{\glsxtrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
  \glsxtrAltTreeIndent=\parindent
}
```

\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}%
}
```

lsgetwidestname Provide a user-level macro to obtain the widest top-level name.

```
\newcommand*{\glsgetwidestname}{\@glswidestname}
```

etwidestsubname Provide a user-level macro to obtain the widest sub-entry name.

```
\newcommand*{\glsgetwidestsubname}[1]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {\@glswidestname}%
  {\csuse{@glswidestname\romannumeral#1}}%
}
```

estTopLevelName CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```
\let\glsFindWidestTopLevelName\glsfindwidesttoplevelname
```

sedTopLevelName Like `\glsfindwidesttoplevelname` but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```
\newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {%
        \ifglshasparent{\@glo@label}%
        {}%
        {%
          \settowidth{\dimen@}%
           {\glstreenamefmt{\glsentryname{\@glo@label}}}%
          \ifdim\dimen@>\gls@tmplen
            \gls@tmplen=\dimen@
            \eglssetwidest{\glsentryname{\@glo@label}}%
          \fi
        }%
      }%
      {}%
    }%
  }%
}
```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```
\newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
```

```
      \ifglsused{\@glo@label}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
      }%
      {}%
    }%
  }%
}
```

ndWidestAnyName Like the above but doesn't check is the entry has been used.

```
\newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \settowidth{\dimen@}%
       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
      \ifdim\dimen@>\gls@tmplen
        \gls@tmplen=\dimen@
        \eglssetwidest{\glsentryname{\@glo@label}}%
      \fi
    }%
  }%
}
```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```
\newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \dimen@i=0pt\relax
  \dimen@ii=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {%
        \ifglshasparent{\@glo@label}%
        {%
          \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
          \ifglshasparent{\@glo@parent}%
          {%
```

```
            \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
            \ifglshasparent{\@glo@parent}%
            {}%
            {%
              \settowidth{\gls@tmplen}%
                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
              \ifdim\gls@tmplen>\dimen@ii
                \dimen@ii=\gls@tmplen
                \eglssetwidest[2]{\glsentryname{\@glo@label}}%
              \fi
            }%
          }%
          {%
            \settowidth{\gls@tmplen}%
               {\glstreenamefmt{\glsentryname{\@glo@label}}}%
            \ifdim\gls@tmplen>\dimen@i
              \dimen@i=\gls@tmplen
              \eglssetwidest[1]{\glsentryname{\@glo@label}}%
            \fi
          }%
        }%
        {%
          \settowidth{\gls@tmplen}%
             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
          \ifdim\gls@tmplen>\dimen@
            \dimen@=\gls@tmplen
            \eglssetwidest{\glsentryname{\@glo@label}}%
          \fi
        }%
      }%
      {}%
    }%
  }%
}
```

dWidestLevelTwo This is like \glsFindWidestUsedLevelTwo but doesn't check if the entry has been used.

```
\newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
  \dimen@=0pt\relax
  \dimen@i=0pt\relax
  \dimen@ii=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglshasparent{\@glo@label}%
      {%
        \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
        \ifglshasparent{\@glo@parent}%
        {%
```

```
            \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
            \ifglshasparent{\@glo@parent}%
            {}%
            {%
              \settowidth{\gls@tmplen}%
                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
              \ifdim\gls@tmplen>\dimen@ii
                \dimen@ii=\gls@tmplen
                \eglssetwidest[2]{\glsentryname{\@glo@label}}%
              \fi
            }%
          }%
          {%
            \settowidth{\gls@tmplen}%
               {\glstreenamefmt{\glsentryname{\@glo@label}}}%
            \ifdim\gls@tmplen>\dimen@i
              \dimen@i=\gls@tmplen
              \eglssetwidest[1]{\glsentryname{\@glo@label}}%
            \fi
          }%
        }%
        {%
          \settowidth{\gls@tmplen}%
             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
          \ifdim\gls@tmplen>\dimen@
            \dimen@=\gls@tmplen
            \eglssetwidest{\glsentryname{\@glo@label}}%
          \fi
        }%
      }%
    }%
  }
```

edAnyNameSymbol Like the \glsFindWidestUsedAnyName but also measures the symbol. The length of the widest symbol is stored in the second argument should be a length register.

```
  \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
    \dimen@=0pt\relax
    \gls@tmplen=0pt\relax
    #2=0pt\relax
    \forallglossaries[#1]{\@gls@type}%
    {%
      \forglsentries[\@gls@type]{\@glo@label}%
      {%
        \ifglsused{\@glo@label}%
        {%
          \settowidth{\dimen@}%
           {\glstreenamefmt{\glsentryname{\@glo@label}}}%
          \ifdim\dimen@>\gls@tmplen
            \gls@tmplen=\dimen@
```

```
            \eglssetwidest{\glsentryname{\@glo@label}}%
          \fi
          \settowidth{\dimen@}%
           {\glsentrysymbol{\@glo@label}}%
          \ifdim\dimen@>#2\relax
            #2=\dimen@
          \fi
        }%
        {}%
      }%
    }%
  }
```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```
  \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
    \dimen@=0pt\relax
    \gls@tmplen=0pt\relax
    #2=0pt\relax
    \forallglossaries[#1]{\@gls@type}%
    {%
      \forglsentries[\@gls@type]{\@glo@label}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
        \settowidth{\dimen@}%
         {\glsentrysymbol{\@glo@label}}%
        \ifdim\dimen@>#2\relax
          #2=\dimen@
        \fi
      }%
    }%
  }
```

eSymbolLocation Like the \glsFindWidestUsedAnyNameSymbol but also measures the location list. This requires \glsentrynumberlist. The length of the widest symbol is stored in the second argument should be a length register. The length of the widest location list is stored in the third argument, which should also be a length register.

```
  \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
    \dimen@=0pt\relax
    \gls@tmplen=0pt\relax
    #2=0pt\relax
    #3=0pt\relax
    \forallglossaries[#1]{\@gls@type}%
    {%
      \forglsentries[\@gls@type]{\@glo@label}%
```

```
    {%
      \ifglsused{\@glo@label}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
        \settowidth{\dimen@}%
         {\glsentrysymbol{\@glo@label}}%
        \ifdim\dimen@>#2\relax
          #2=\dimen@
        \fi
        \settowidth{\dimen@}%
         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
        \ifdim\dimen@>#3\relax
          #3=\dimen@
        \fi
      }%
      {}%
    }%
  }%
}
```

eSymbolLocation Like the \glsFindWidestUsedAnyNameSymbol but doesn't check if the entry has been used.

```
\newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  #3=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \settowidth{\dimen@}%
       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
      \ifdim\dimen@>\gls@tmplen
        \gls@tmplen=\dimen@
        \eglssetwidest{\glsentryname{\@glo@label}}%
      \fi
      \settowidth{\dimen@}%
       {\glsentrysymbol{\@glo@label}}%
      \ifdim\dimen@>#2\relax
        #2=\dimen@
      \fi
      \settowidth{\dimen@}%
        {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
      \ifdim\dimen@>#3\relax
```

```
          #3=\dimen@
        \fi
      }%
    }%
}
```

AnyNameLocation Like the \glsFindWidestUsedAnyNameSymbolLocation but doesn't measure the symbol. The length of the widest location list is stored in the second argument, which should be a length register.

```
\newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \ifglsused{\@glo@label}%
      {%
        \settowidth{\dimen@}%
         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
        \ifdim\dimen@>\gls@tmplen
          \gls@tmplen=\dimen@
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
        \settowidth{\dimen@}%
         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
        \ifdim\dimen@>#2\relax
          #2=\dimen@
        \fi
      }%
      {}%
    }%
  }%
}
```

AnyNameLocation Like the \glsFindWidestAnyNameLocation but doesn't check the first use flag.

```
\newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
  \dimen@=0pt\relax
  \gls@tmplen=0pt\relax
  #2=0pt\relax
  \forallglossaries[#1]{\@gls@type}%
  {%
    \forglsentries[\@gls@type]{\@glo@label}%
    {%
      \settowidth{\dimen@}%
       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
      \ifdim\dimen@>\gls@tmplen
        \gls@tmplen=\dimen@
```

```
          \eglssetwidest{\glsentryname{\@glo@label}}%
        \fi
        \settowidth{\dimen@}%
         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
        \ifdim\dimen@>#2\relax
          #2=\dimen@
        \fi
      }%
    }%
  }
```

mputeTreeIndent Compute the value of \glstreeindent. Argument is the entry label. (Ignored in default definition, but this command may be redefined to take the particular entry into account.) Note that the sub-levels modify \glstreeindent.

```
\newcommand*{\glsxtrComputeTreeIndent}[1]{%
  \glstreeindent=\glsxtrtreetopindent\relax
}
```

teTreeSubIndent

```
%\cs{\glsxtrComputeTreeSubIndent}\marg{level}\marg{label}\marg{register}
%\end{macrocode}
% 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.
%    \begin{macrocode}
\newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {%
    \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
  }%
  {%
    \settowidth{#3}{\glstreenamefmt{%
           \csname @glswidestname\romannumeral#1\endcsname\space}}%
  }%
}
```

eeSetHangIndent Set \hangindent for top-level entries:

```
\newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
\newcommand*{\glsxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
\renewglossarystyle{alttree}{%
  \renewenvironment{theglossary}%
    {%
     \glsxtralttreeInit
     \def\@gls@prevlevel{-1}%
```

```
     \mbox{}\par}%
   {\par}%
 \renewcommand*{\glossaryheader}{}%
 \renewcommand*{\glsgroupheading}[1]{}%
 \renewcommand{\glossentry}[2]{%
   \ifnum\@gls@prevlevel=0\relax
   \else
     \glsxtrComputeTreeIndent{##1}%
   \fi
   \parindent\glstreeindent
   \glsxtrAltTreeSetHangIndent
   \makebox[0pt][r]%
   {%
     \glstreenamebox{\glstreeindent}%
     {%
        \glsentryitem{##1}%
        \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
     }%
   }%
   \glsxtralttreeSymbolDescLocation{##1}{##2}%
   \def\@gls@prevlevel{0}%
 }
 \renewcommand{\subglossentry}[3]{%
   \ifnum##1=1\relax
     \glssubentryitem{##2}%
   \fi
   \ifnum\@gls@prevlevel=##1\relax
   \else
     \glsxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
     \ifnum\@gls@prevlevel<##1\relax
       \setlength\glstreeindent\gls@tmplen
       \addtolength\glstreeindent\parindent
       \parindent\glstreeindent
     \else
       \ifnum\@gls@prevlevel=0\relax
         \glsxtrComputeTreeIndent{##2}%
       \else
         \glsxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
       \fi
       \addtolength\parindent{-\glstreeindent}%
       \setlength\glstreeindent\parindent
     \fi
   \fi
   \glsxtrAltTreeSetSubHangIndent{##1}%
   \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
   \glsxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
   \def\@gls@prevlevel{##1}%
 }%
```

```
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
  }
}%
{%
```

Assume the style isn't required if it hasn't already been defined.

```
}
```

Reset the default style

```
\ifx\@glossary@default@style\relax
\else
  \setglossarystyle{\@glsxtr@current@style}
\fi
```

Glossary

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 177
\glsfmtshort: new 176
\Glsfmtshortpl: new 177
\glsfmtshortpl: new 176
short: switched inline full form to short (long) 143
0.3 (2015-12-02)
\@ACRlong: added redefinition 40
\@ACRlongpl: added redefinition 41
\@ACRshort: added redefinition 38
\@ACRshortpl: added redefinition 39
\@Acrlong: added redefinition 40
\@Acrlongpl: added redefinition 41
\@Acrshort: added redefinition 38
\@Acrshortpl: added redefinition 39
\@GLSdesc@: added redefinition 34
\@GLSdescplural@: added redefinition . 34
\@GLSfirst@: added redefinition 31
\@GLSfirstplural@: added redefinition 33
\@GLSname@: added redefinition 33
\@GLSplural@: added redefinition 32
\@GLSsymbol@: added redefinition 35
\@GLSsymbolplural@: added redefinition 35
\@GLStext@: added redefinition 30
\@GLSuseri@: added redefinition 36
\@GLSuserii@: added redefinition 36
\@GLSuseriii@: added redefinition 36
\@GLSuseriv@: added redefinition 37
\@GLSuserv@: added redefinition 37
\@GLSuservi@: added redefinition 37
\@Glsdesc@: added redefinition 34
\@Glsdescplural@: added redefinition . 34
\@Glsfirst@: added redefinition 31
\@Glsfirstplural@: added redefinition 33
\@Glsname@: added redefinition 33
\@Glsplural@: added redefinition 32
\@Glssymbol@: added redefinition 35
\@Glssymbolplural@: added redefinition 35
\@Glstext@: added redefinition 31
\@Glsuseri@: added redefinition 36
\@Glsuserii@: added redefinition 36
\@Glsuseriii@: added redefinition 36
\@Glsuseriv@: added redefinition 36
\@Glsuserv@: added redefinition 37
\@Glsuservi@: added redefinition 37
\@acrlong: added redefinition 40
\@acrlongpl: added redefinition 41
\@acrshort: added redefinition 37
\@acrshortpl: added redefinition 38
\@gls@field@link: added optional argument 28
\@glsdescplural@: added redefinition . 34
\@glsfirst@: added redefinition 31
\@glsfirstplural@: added redefinition 32
\@glsplural@: added redefinition 32
\@glssymbolplural@: added redefinition 35
\@glsxtr@defaultnoglossarywarning: new 79
\@glsxtr@field@linkdefs: new 30
\@glsxtr@insertdots: new 116
\@print@glossary: added redefinition . 76
\glsabbrvdefaultfont: renamed from \abbrvdefaultfont 120
\glsaccessdesc: new 86
\glsaccessdescplural: new 86
\glsaccessfirst: new 83
\glsaccessfirstplural: new 84
\Glsaccesslong: new 88
\glsaccesslong: new 88
\glsaccessname: new 82
\glsaccessplural: new 83
\Glsaccessshort: new 87
\glsaccessshort: new 87
\Glsaccessshortpl: new 88

\glsaccessshortpl: new 88
\glsaccesssymbol: new 85
\glsaccesssymbolplural: new 85
\glsaccesstext: new 82
\glsentryfmt: added check for short .. 27
\glslongpltok: new 116
\glsshortpltok: new 116
\glsxtrdiscardperiod: added check for plural 113
\GLSxtrlongpl: new 129
\Glsxtrlongpl: new 128
\glsxtrlongpl: new 128
\glsxtrNoGlossaryWarning: new 12
\glsxtrpostlinkAddDescOnFirstUse: new 112
\glsxtrpostlinkAddSymbolOnFirstUse: new 112
\glsxtrpostlinkendsentence: new . 112
\GLSxtrshortpl: new 127
\Glsxtrshortpl: new 127
\glsxtrshortpl: new 126
short-long-desc: fixed name to use \glslabeltok 138
\newabbreviation: fixed family name in \setkeys 116
long-short-desc: fixed name to use \glslabeltok 136
0.4 (2015-12-03)
\@glsxtr@doabbreviationsdef: added redefinition of \acronymtype 10
\Glsfmtshort: changed to use \Glsxtrshort 177
\glsfmtshort: changed to use \glsxtrshort 176
\Glsfmtshortpl: changed to use \glsxtrshortpl 177
\glsfmtshortpl: changed to use \glsxtrshortpl 176
\glsxtrifemptyglossary: new 15
\glsxtrnewnumber: added extra argument 97
\glsxtrnewsymbol: added extra argument 97
\MakeAcronymsAbbreviations: set the default type to \acronymtype 71
\newterm: fixed name argument 96
0.5 (2015-12-07)
\@cGLS: new 62
\@cGLS@: new 62
\@cGLSpl: new 62
\@cGLSpl@: new 62
\@glsxtr@setentrycountunsetattr: new 57
\cGLS: new 62
\cGLSformat: new 62
\cGLSpl: new 62
\cGLSplformat: new 62
\GlossariesExtraWarningNoLine: new 9
\glsenableentrycount: new 58
\glsfirstabbrvdefaultfont: new .. 119
\glsfirstlongdefaultfont: new ... 120
\Glsfmtfirst: new 179
\glsfmtfirst: new 178
\Glsfmtfirstpl: new 179
\glsfmtfirstpl: new 179
\Glsfmtplural: new 178
\glsfmtplural: new 178
\Glsfmtshort: changed to use \Glsxtrtitleshort 177
renamed from \Glsentryfmtshort . 177
\glsfmtshort: changed to use \glsxtrtitleshort 176
renamed from \glsentryfmtshort . 176
\Glsfmtshortpl: changed to use \Glsxtrtitleshortpl 177
renamed from \Glsentryfmtshortpl 177
\glsfmtshortpl: changed to use \glsxtrtitleshortpl 176
renamed from \glsentryfmtshortpl 176
\Glsfmttext: new 178
\glsfmttext: new 177
\glshasattribute: new 94
\glshascategoryattribute: new 93
\GlsXtrEnableEntryCounting: new .. 57
\glsxtrifcounttrigger: new 60
\glsxtrscfont: new 147
\glsxtrscsuffix: new 148
\glsxtrsmfont: new 151
\glsxtrsmsuffix: new 152
short-em: new 158
short-em-desc: new 159
short-em-footnote: new 160
short-em-long: new 157
short-em-long-desc: new 157
short-em-postfootnote: new 161

short-sc-footnote: new 151
short-sc-postfootnote: new 151
short-sm: new 153
short-sm-desc: new 153
short-sm-footnote: new 154
short-sm-long: new 152
short-sm-long-desc: new 152
short-sm-postfootnote: new 154
long-noshort-em: new 159
long-noshort-em-desc: new 160
long-noshort-sm: new 153
long-noshort-sm-desc: new 154
long-short-em: new 155
long-short-em-desc: new 155
long-short-sm: new 152
long-short-sm-desc: new 152
0.5.1 (2015-12-02)
\Glsaccesstext: new 83
0.5.1 (2015-12-07)
\@glsxtr@doaccsupp: new 12
General: removed \ifglsxtruseuchead 168
\Glsaccessdesc: new 86
\Glsaccessdescplural: new 87
\Glsaccessfirst: new 84
\Glsaccessfirstplural: new 84
\Glsaccessname: new 82
\Glsaccessplural: new 83
\Glsaccesssymbol: new 85
\Glsaccesssymbolplural: new 85
\Glsxtrheadfirst: now uses headuc attribute 171
\glsxtrheadfirst: now uses headuc attribute 171
\Glsxtrheadfirstplural: now uses headuc attribute 172
\glsxtrheadfirstplural: now uses headuc attribute 172
\Glsxtrheadplural: now uses headuc attribute 171
\glsxtrheadplural: now uses headuc attribute 170
\Glsxtrheadshort: now uses headuc attribute 169
\glsxtrheadshort: now uses headuc attribute 168
\Glsxtrheadshortpl: now uses headuc attribute 169
\glsxtrheadshortpl: now uses headuc attribute 168
\Glsxtrheadtext: now uses headuc attribute 170
\glsxtrheadtext: now uses headuc attribute 169
short-long: switch off regular attribute if set 137
short-long-desc: switch off regular attribute if set 138
long-short: switch off regular attribute if set 135
long-short-desc: switch off regular attribute if set 137
footnote: switch off regular attribute if set 139
postfootnote: switch off regular attribute if set 141
0.5.2 (2015-12-08)
\@GLSdesc@: added accessibility support 34
\@GLSdescplural@: added accessibility support 34
\@GLSfirst@: added accessibility support 31
\@GLSfirstplural@: added accessibility support 33
\@GLSname@: added accessibility support 33
\@GLSplural@: added accessibility support 32
\@GLSsymbol@: added accessibility support 35
\@GLSsymbolplural@: added accessibility support 35
\@GLStext@: added accessibility support 30
\@Glsdesc@: added accessibility support 34
\@Glsdescplural@: added accessibility support 34
\@Glsfirst@: added accessibility support 31
\@Glsfirstplural@: added accessibility support 33
\@Glsname@: add accessibility support .. 33
\@Glsplural@: added accessibility support 32
\@Glssymbol@: added accessibility support 35
\@Glssymbolplural@: added accessibility support 35
\@Glstext@: added accessibility support 31
\@glsdesc@: added accessibility support 34

\@glsdescplural@: added accessibility support 34
\@glsfirst@: added accessibility support 31
\@glsfirstplural@: added accessibility support 32
\@glsname@: added accessibility support 33
\@glsplural@: added accessibility support 32
\@glssymbol@: added accessibility support 35
\@glssymbolplural@: added accessibility support 35
\@glstext@: added accessibility support 30
\@glsxtr@activate@initialtagging: new 110
\@glsxtr@do@titlecaps@warn: new . 110
\@glsxtr@tag: new 110
General: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded 82
removed \glsxtrabbrvfmt 130
\glossaryentrynumbers: added 24
\Glossentrydesc: added 108
\Glossentryname: added 102
\Glossentrysymbol: added 109
\glossentrysymbol: added 108
\GLSaccessdesc: new 86, 91
\GLSaccessdescplural: new 87, 91
\GLSaccessfirst: new 84, 90
\GLSaccessfirstplural: new 84, 90
\GLSaccesslong: new 88, 92
\GLSaccesslongpl: new 89, 92
\Glsaccesslongpl: new 89
\glsaccesslongpl: new 89
\GLSaccessname: new 82, 89
\GLSaccessplural: new 83, 90
\GLSaccessshort: new 87, 91
\GLSaccessshortpl: new 88, 92
\GLSaccesssymbol: new 85, 90
\GLSaccesssymbolplural: new . . . 86, 91
\GLSaccesstext: new 83, 90
\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here 27
\GlsXtrEnableInitialTagging: new 109
\glsxtrfieldtitlecase: new 98
\GlsXtrFormatLocationList: new . . . 25
\glsxtrnewabbrevpresetkeyhook: new 118
\glsxtrtagfont: new 110
\KV@printgloss@nonumberlist: added 27
\mfu@checkword@do: added 110
\setabbreviationstyle: added check for post-definition style switch 132
0.5.3 (2015-12-09)
\@glsxtr@autoindex@at: new 106
\@glsxtr@autoindex@encap: new . . . 106
\@glsxtr@autoindex@esc: new 107
\@glsxtr@autoindex@level: new . . . 107
\@glsxtr@autoindex@setname: new . 105
\@glsxtr@doabbreviationsdef: new . . 9
General: removed \GlsXtrNoGlsWarningNoAutoMakeMain 78
\glsdescwidth: added 24
\glspagelistwidth: added 24
\glsxtrdoautoindexname: new 104
\glsxtrpostnamehook: new 104
\if@glsxtr@format@override: new . 104
\ProvidesGlossariesExtraLang: new 182
\RequireGlossariesExtraLang: new 182
0.5.4 (2015-12-15)
\@@newglossaryentry@defunitcounters: new 63
\@GLSxtr@p@acrlong@: new 50
\@GLSxtr@p@acrlongpl@: new 51
\@GLSxtr@p@acrshort@: new 50
\@GLSxtr@p@acrshortpl@: new 50
\@GLSxtr@p@long@: new 50
\@GLSxtr@p@longpl@: new 50
\@GLSxtr@p@plural@: new 48
\@GLSxtr@p@short@: new 49
\@GLSxtr@p@shortpl@: new 49
\@GLSxtr@p@text@: new 48
\@GlsXtrEnableOnTheFly: new 20
\@Glsxtr: new 21
\@Glsxtr@p@acrlong@: new 50
\@Glsxtr@p@acrlongpl@: new 51
\@Glsxtr@p@acrshort@: new 50
\@Glsxtr@p@acrshortpl@: new 50
\@Glsxtr@p@long@: new 50
\@Glsxtr@p@longpl@: new 50
\@Glsxtr@p@plural@: new 48
\@Glsxtr@p@short@: new 49
\@Glsxtr@p@shortpl@: new 49
\@Glsxtr@p@text@: new 48

\@Glsxtrpl: new 22
\@alt@gls@hyp@opt: new 46
\@gls@alt@hyp@opt: new 46
\@gls@alt@hyp@opt@char: new 46
\@gls@alt@hyp@opt@keys: new 46
\@gls@increment@currunitcount: new 64
\@gls@local@increment@currunitcount: new 64
\@gls@setdefault@glslink@opts: new 44
\@glsxtr: new 21
\@glsxtr@addunitcounter: new 63
\@glsxtr@currunitcount: new 65
\@glsxtr@ifunitcounter: new 64
\@glsxtr@p@acrlong@: new 50
\@glsxtr@p@acrlongpl@: new 51
\@glsxtr@p@acrshort@: new 50
\@glsxtr@p@acrshortpl@: new 50
\@glsxtr@p@long@: new 49
\@glsxtr@p@longpl@: new 50
\@glsxtr@p@plural@: new 48
\@glsxtr@p@short@: new 48
\@glsxtr@p@shortpl@: new 49
\@glsxtr@p@text@: new 48
\@glsxtr@prevunitcount: new 65
\@glsxtr@setentryunitcountunsetattr: new 69
\@glsxtr@unitcountlist: new 63
\@glsxtrpl: new 21
\@newglossaryentryposthook: added empty see value if not set and added 'see' to field key map 17
\@sGlsXtrEnableOnTheFly: new 20
\cGlsformat: added 63
\cglsformat: added 63
\cGlsplformat: added 63
\cglsplformat: added 63
\glsdisablehyper: added 47
\glsdohyperlink: added 47
\glsdonohyperlink: added 47
\glsenableentryunitcount: new 65
\glshasattribute: added check for entry's existence 94
\glsifattribute: added check for entry's existence 94
\glspostlinkhook: added existence check 111
\Glsxtr: new 21
\glsxtr: new 21
\glsxtrcat: new 20
\glsxtrdowrglossaryhook: new 46
\GlsXtrEnableEntryUnitCounting: new 68
\GlsXtrEnableOnTheFly: new 20
\Glsxtrpl: new 22
\glsxtrpl: new 21
\glsxtrpostlocalreset: new 57
\glsxtrpostlocalunset: new 56
\glsxtrpostreset: new 57
\glsxtrpostunset: new 56
\glsxtrprotectlinks: new 47
\GlsXtrSetAltModifier: new 46
\GlsXtrSetDefaultGlsOpts: new 45
\glsxtrstarflywarn: new 20
\GlsXtrWarning: new 22
\MakeAcronymsAbbreviations: now disables \setacronymstyle 71
1.0 (2016-01-24)
\@glsxtr@autoindexcrossrefs: new . . 9
\@glsxtr@idx@displaynumberlist: new 74
\@glsxtr@idx@entrynumberlist: new 76
\@glsxtr@noidx@displaynumberlist: new 75
\@glsxtr@noidx@entrynumberlist: new 76
\@glsxtr@noidx@numberlistloop: new 75
\@glsxtr@reg@glosslist: new 72
\makeglossaries: new 72
1.01 (2016-02-02)
\glsxtrdiscardperiod: added check for first use 113
short-desc: fixed typo in \glsxtrinlinefullformat and added missing second argument . . . 144
1.02 (2016-04-25)
\@glsxtr@current@style: new 23
\Glsfmtfull: new 181
\glsfmtfull: new 181
\Glsfmtfullpl: new 182
\glsfmtfullpl: new 181
\Glsfmtlong: new 180
\glsfmtlong: new 180
\Glsfmtlongpl: new 180
\glsfmtlongpl: new 180
\Glsxtrheadfull: new 175

\glsxtrheadfull: new 174
\Glsxtrheadfullpl: new 176
\glsxtrheadfullpl: new 175
\Glsxtrheadlong: new 173
\glsxtrheadlong: new 173
\Glsxtrheadlongpl: new 174
\glsxtrheadlongpl: new 173
\Glsxtrtitlefull: new 175
\glsxtrtitlefull: new 175
\Glsxtrtitlefullpl: new 176
\glsxtrtitlefullpl: new 175
\Glsxtrtitlelong: new 174
\glsxtrtitlelong: new 173
\Glsxtrtitlelongpl: new 174
\glsxtrtitlelongpl: new 173
\ifglsxtrinsertinside: new 135
postfootnote: added redef of \glsxtrsetupfulldefs 141
stylemods: new 13
1.03 (2016-04-27)
\@GLSfirstplural@: bug fix: misspelt cs name 33
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural 32
\@Glsfirstplural@: bug fix: misspelt cs name 33
\@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural 32
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural 32
\glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl . . 173
\glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl 168
1.04 (2015-04-30)
short-em-footnote: renamed from "footnote-em" 160
1.04 (2016-05-02)
\@@glsxtrpostloctag: new 26
\@GLSdesc@: set abbreviation and regular format 34
\@GLSdescplural@: set abbreviation and regular format 34
\@GLSfirst@: set abbreviation format . . 31
\@GLSfirstplural@: set abbreviation and regular format 33
\@GLSname@: set abbreviation and regular format 33
\@GLSplural@: set abbreviation and regular format 32
\@GLSsymbol@: set regular format 35
\@GLSsymbolplural@: set regular format 35
\@GLStext@: set abbreviation and regular format 30
\@GLSuseri@: set regular format 36
\@GLSuserii@: set regular format 36
\@GLSuseriii@: set regular format 36
\@GLSuseriv@: set regular format 37
\@GLSuserv@: set regular format 37
\@GLSuservi@: set regular format 37
\@Glsdesc@: set abbreviation and regular format 34
\@Glsdescplural@: set abbreviation and regular format 34
\@Glsfirst@: set abbreviation and regular format 31
\@Glsfirstplural@: set abbreviation and regular format 33
\@Glsname@: set abbreviation and regular format 33
\@Glsplural@: set abbreviation and regular format 32
\@Glssymbol@: set regular format 35
\@Glssymbolplural@: set regular format 35
\@Glstext@: set abbreviation and regular format 31
\@Glsuseri@: set regular format 36
\@Glsuserii@: set regular format 36
\@Glsuseriii@: set regular format 36
\@Glsuseriv@: set regular format 36
\@Glsuserv@: set regular format 37
\@Glsuservi@: set regular format 37
\@gls@preglossaryhook: added check for entry's existence 111
\@glsdesc@: set abbreviation and regular format 34
\@glsdescplural@: set abbreviation and regular format 34
\@glsfirst@: set abbreviation and regular format 31
\@glsfirstplural@: set abbreviation and regular format 32
\@glsname@: set abbreviation and regular format 33
\@glsplural@: set abbreviation and regular format 32
\@glssymbol@: set regular format 35

\@glssymbolplural@: set regular format 35
\@glstext@: set abbreviation and regular format 30
\@glsxtr@deprecated@abbrstyle: new 134
\@glsxtr@do@style: new 13
\@glsxtr@doloctag: new 27
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand 76
\@glsxtr@pagestag: new 26
\@glsxtr@pagetag: new 26
\@glsxtr@preloctag: new 26
\@glsxtrpostloctag: new 26
\@glsxtrpreloctag: new 26
\glossentrydesc: added glossdescfont attribute check 98
\Glossentryname: added glossnamefont attribute check 102
\glossentryname: added glossnamefont attribute check 100
moved post name hook inside condition 102
\glsabbrvemfont: new 155
\glsabbrvuserfont: new 162
\glsfirstabbrvemfont: new 155
\glsfirstabbrvuserfont: new 162
\glsfirstlongemfont: new 155
\glsfirstlonguserfont: new 162
\glsifnotregularcategory: new 95
\glslongdefaultfont: new 120
\glslongemfont: new 155
\glslongfont: new 120
\glslonguserfont: new 162
\glsxtrassignfieldfont: new 30
\GlsXtrEnablePreLocationTag: new . 25
\glsxtrfirstscfont: new 148
\glsxtrfirstsmfont: new 151
\glsxtrlongshortdescsort: new . . . 136
\glsxtrpostnamehook: added category check 104
\glsxtrregularfont: new 27
\glsxtruserfield: new 161
\glsxtruserparen: new 161
\glsxtrusersuffix: new 162
\GlsXtrWarnDeprecatedAbbrStyle: new 134
short-em-long-em: new 157
short-em-long-em-desc: new 158
short-em-nolong: new 158
short-em-nolong-desc: new 159
short-em-postfootnote: renamed from “postfootnote-em” 161
short-footnote: new 141
short-long-user: new 163
short-long-user-desc: new 164
short-nolong: new 144
short-nolong-desc: new 145
short-postfootnote: new 142
short-sc-footnote: renamed from “footnote-sc” 151
short-sc-nolong: new 149
short-sc-nolong-desc: new 150
short-sc-postfootnote: renamed from “postfootnote-sc” 151
short-sm-footnote: renamed from “footnote-sm” 154
short-sm-nolong: new 153
short-sm-nolong-desc: new 153
short-sm-postfootnote: renamed from “postfootnote-sm” 154
\letabbreviationstyle: new 134
\newabbreviationstyle: bug fix: corrected test for existence 133
long-em-noshort-em: new 159
long-em-noshort-em-desc: new 160
long-em-short-em: new 156
long-em-short-em-desc: new 156
long-noshort: new 147
long-noshort-desc: new 147
long-noshort-em: renamed from “long-em” 159
long-noshort-em-desc: renamed from “long-desc-em” 160
long-noshort-sc: renamed from “long-sc” 150
long-noshort-sc-desc: renamed from “long-desc-sc” 150
long-noshort-sm: renamed from “long-sm” 153
long-noshort-sm-desc: renamed from \long-desc-sm 154
long-short-user: new 162
long-short-user-desc: new 163
\renewabbreviationstyle: new 133
style: new 13
1.05 (2016-06-10)
\eglssetwidest: new 191
\glsFindWidestAnyName: new 193

\glsFindWidestAnyNameLocation: new 198
\glsFindWidestAnyNameSymbol: new 196
\glsFindWidestAnyNameSymbolLocation: new 197
\glsFindWidestLevelTwo: new 194
\glsFindWidestUsedAnyName: new . . 192
\glsFindWidestUsedAnyNameLocation: new 198
\glsFindWidestUsedAnyNameSymbol: new 195
\glsFindWidestUsedAnyNameSymbolLocation: new 196
\glsFindWidestUsedLevelTwo: new . 193
\glsFindWidestUsedTopLevelName: new 192
\glsfirstlongfootnotefont: new . . 139
\glsgetwidestname: new 191
\glsgetwidestsubname: new 192
\glslongfootnotefont: new 139
\glsxtrAltTreeIndent: new 191
\glsxtralttreeInit: new 191
\glsxtrAltTreePar: new 191
\glsxtrAltTreeSetHangIndent: new 199
\glsxtrAltTreeSetSubHangIndent: new 199
\glsxtralttreeSubSymbolDescLocation: new 191
\glsxtralttreeSymbolDescLocation: new 190
\glsxtrComputeTreeIndent: new . . . 199
\glsxtrComputeTreeSubIndent: new 199
\glsxtrtreetopindent: new 191
short-em-long: fixed incorrect font used by long form 157
\xglssetwidest: new 191
1.06 (2016-06-18)
\@glsdoifexistsorwarn: new 9
\@glsxtr@docdefval: new 8
\@glsxtr@usesee: new 17
General: disabled docdef key at the start of the document 14
docdef option changed to choice 8
\glsxtr@usesee: new 17
\glsxtrusesee: new 17
\glsxtruseseeformat: new 17
\if@glsxtrdocdefrestricted: new . . . 8
1.07 (2016-08-15)
\@@glsxtrp: new 51
\@GLSfirst@: added check for nohyperfirst attribute 32
\@GLSfirstplural@: added check for nohyperfirst attribute 33
\@GLSxtrp: new 52
\@Glsfirst@: added check for nohyperfirst attribute 31
\@Glsfirstplural@: added check for nohyperfirst attribute 33
\@Glsxtrp: new 52
\@gls@preglossaryhook: added \glossxtrsetpopts 111
\@glsfirst@: added check for nohyperfirst attribute 31
\@glsfirstplural@: added check for nohyperfirst attribute 32
\@glsxtrinmark: new 166
\@glsxtrnotinmark: new 166
\@glsxtrp: new 51
\@glsxtrp@opt: new 51
\glossxtrsetpopts: new 51
\glsps: new 54
\glspt: new 54
\glsxtr@entry@p: new 52
\glsxtrabbrvfootnote: new 139
\glsxtrchecknohyperfirst: new 31
\glsxtrfieldtitlecasecs: new 98
\glsxtrifinmark: new 166
\GLSxtrp: new 55
\Glsxtrp: new 54
\glsxtrp: new 53
\glsxtrsetpopts: new 51
short-long-desc: added text key 138
fixed misspelling of \glsabbrvfont in plural key 138
long-short-desc: added missing text key 136
fixed misspelling of \glsabbrvfont . 137
footnote: changed first forms to use \glsfirstlongfootnotefont . . . 139
postfootnote: removed \footnote from first keys 141
switched from \glsfirstlongfont to \glsfirstlongfootnotefont . . . 142
\RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glsxtrifwasfirstuse 71
1.08 (2016-12-13)
\@@glsxtr@record: new 6

\@GLS@: added \@glsxtr@record 29
\@GLSpl@: added \@glsxtr@record ... 29
\@Gls@: added \@glsxtr@record 28
\@Glspl@: added \@glsxtr@record ... 29
\@gls@: added \@glsxtr@record 28
\@gls@@link@: added \@glsxtr@record 29
\@gls@field@link: added \@glsxtr@record 28
\@gls@saveentrycounter: new 14
\@glsdispl: added \@glsxtr@record . 29
\@glspl@: added \@glsxtr@record ... 28
\@glsxtr@dorecord: new 6
\@glsxtr@err@undefaction: new 5
\@glsxtr@record: new 6
\@glsxtr@warn@onexistsordo: new ... 5
\@glsxtr@warn@undefaction: new 5
\@print@unsrt@glossary: new 81
General: added record package option 7
\glsadd: added \@glsxtr@record 29
\glsdoifexists: now defines \glslabel 15
\glsxtr@@do@wrglossary: new 14
\glsxtr@addloclistfield: new 7
\glsxtr@indexonly@saveentrycounter: new 7
\glsxtr@record: new 6
\glsxtr@resource: new 81
\glsxtr@saveentrycounter: new 14
\glsxtr@setup@record: new 7
\glsxtrassignfieldfont: added check for existence 30
\glsxtrresourcefile: new 81
\printunsrtglossaries: new 81
\printunsrtglossary: new 81

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

\. 112, 190
\@@cGLS@ 59, 67
\@@cGLSpl@ 59, 67
\@@cGlspl@ 59, 67
\@@cgls@ 59, 67
\@@cglspl@ 59, 60, 67
\@@do@@wrglossary 6
\@@do@wrglossary 7, 8, 14, 30
\@@glo@assign@sortkey 74
\@@glo@no@assign@sortkey 74
\@@glo@type 81
\@@glslocalreset 57
\@@glslocalunset 56
\@@glsreset 56
\@@glsunset 56
\@@glsxtr@autoindex@escspch ... 106–108
\@@glsxtr@checkspch 105, 106, 108
\@@glsxtr@disabledflycommand 23
\@@glsxtr@record 7, 8
\@@glsxtrp 51, 52
\@@glsxtrpostloctag 25
\@@glsxtrpreloctag 25, 26
\@@newglossaryentry@defcounters 58
\@@newglossaryentry@defunitcounters 65
\@@par 191
\@ACRlong 48
\@ACRlongpl 48
\@ACRshort 48
\@ACRshortpl 48
\@Acrlong 48
\@Acrlongpl 48
\@Acrshort 48
\@Acrshortpl 48
\@GLS@ 29, 47, 61, 62
\@GLSdesc@ 34
\@GLSpl@ 47, 61, 62
\@GLSplural@ 48
\@GLSsymbol@ 35
\@GLStext@ 48
\@GLSxtr@full 121
\@GLSxtr@fullpl 123
\@GLSxtr@p@acrlong@ 48
\@GLSxtr@p@acrlongpl@ 48
\@GLSxtr@p@acrshort@ 48
\@GLSxtr@p@acrshortpl@ 48
\@GLSxtr@p@long@ 48
\@GLSxtr@p@longpl@ 48
\@GLSxtr@p@plural@ 47
\@GLSxtr@p@short@ 48
\@GLSxtr@p@shortpl@ 48
\@GLSxtr@p@text@ 47
\@GLSxtrlong 48, 126
\@GLSxtrlongpl 48, 129
\@GLSxtrp 55, 56
\@GLSxtrshort 48, 124
\@GLSxtrshortpl 48, 127
\@Gls@ 47, 61, 62
\@Gls@acrentryname 69
\@Gls@entry@field 42, 54, 55
\@Gls@entryname 69
\@GlsXtrEnableOnTheFly 20
\@Glspl@ 47, 61, 62
\@Glsplural@ 48
\@Glstext@ 48
\@Glsxtr 21, 22
\@Glsxtr@full 121
\@Glsxtr@fullpl 122
\@Glsxtr@p@acrlong@ 48
\@Glsxtr@p@acrlongpl@ 48
\@Glsxtr@p@acrshort@ 48
\@Glsxtr@p@acrshortpl@ 48
\@Glsxtr@p@long@ 48
\@Glsxtr@p@longpl@ 48

\@Glsxtr@p@plural@ 47
\@Glsxtr@p@short@ 48
\@Glsxtr@p@shortpl@ 48
\@Glsxtr@p@text@ 47
\@Glsxtrlong 48, 125
\@Glsxtrlongpl 48, 129
\@Glsxtrp 54, 55
\@Glsxtrpl 22
\@Glsxtrshort 48, 124
\@Glsxtrshortpl 48, 127
\@acrlong 48
\@acrlongpl 48
\@acrshort 48
\@acrshortpl 48
\@alt@gls@hyp@opt 46
\@auxout 6, 26, 59, 68, 72, 76, 77, 81
\@cGLS 62
\@cGLS@ 59, 62, 67
\@cGLSpl 62
\@cGLSpl@ 59, 62, 67
\@cGlspl@ 59, 67
\@cgls@ 59, 61, 67
\@cglspl@ 59, 67
\@disable@onlypremakeg 73
\@do@auxoutstuff 76, 77
\@do@newglossaryentry 70, 118
\@empty 30, 38–41, 105, 106, 120–129
\@end@glsxtr@addunused 18
\@end@glsxtr@usesee 17
\@endfortrue 132
\@firstofone 30, 99, 104, 110
\@firstofthree .....
30, 38–41, 46, 120, 122, 123, 125, 127, 128
\@firstoftwo 31–35, 39, 41, 44, 46,
71, 113, 114, 121–123, 127–129, 166, 167
\@for 13, 18, 57, 69, 72, 74, 81, 97, 109
\@glo@assign@sortkey 74
\@glo@category 63
\@glo@counterprefix 6
\@glo@countunit 63
\@glo@default@sorttype 74
\@glo@label 7, 17, 18, 42, 192–199
\@glo@loclist 7
\@glo@name 105
\@glo@parent 193–195
\@glo@see 17, 18
\@glo@sort 105
\@glo@sorttype 74
\@glo@thisvalue 161
\@glo@tmp 42
\@glo@type 18, 69, 72, 74, 76, 77, 79–81
\@glo@types 95, 96, 192–198
\@glossary@default@style 23, 201
\@gls@ 47, 60, 61
\@gls@@link 29
\@gls@actualchar 105
\@gls@adjustmode 29
\@gls@alt@hyp@opt 46
\@gls@alt@hyp@opt@char 46
\@gls@alt@hyp@opt@keys 46
\@gls@automake 74
\@gls@checkedmkidx 105, 106, 108
\@gls@checkmkidxchars 105
\@gls@codepage 77
\@gls@counter 6, 30
\@gls@currentlettergroup 81
\@gls@declareoption 4
\@gls@doautomake 74
\@gls@encapchar 106
\@gls@entry@count 59
\@gls@entry@field 42, 52–56, 58
\@gls@entry@unitcount 67, 68
\@gls@field@font 30–37
\@gls@field@link 30–37, 42, 43
\@gls@hyp@opt 42, 43, 46, 62, 120–129
\@gls@hyp@opt@cs 46
\@gls@increment@currcount 58
\@gls@increment@currunitcount 66
\@gls@keymap 7, 17, 42
\@gls@label 6, 45, 46, 132
\@gls@levelchar 106
\@gls@link 28, 29, 38–42, 120–129
\@gls@link@checkfirsthyper 71
\@gls@link@nocheckfirsthyper .....
..... 28, 38–41, 120–129
\@gls@local@increment@currcount 58
\@gls@local@increment@currunitcount 66
\@gls@loclist 75, 76
\@gls@longpl 115–117
\@gls@noidx@do 81
\@gls@noidx@nosanitizesort 74
\@gls@noidx@sanitizesort 74
\@gls@noidxloclist@finalsep 75
\@gls@noidxloclist@prev 75
\@gls@noidxloclist@sep 75
\@gls@org@glsnoidxdisplayloc 75, 76
\@gls@org@glsseeformat 75, 76
\@gls@preglossaryhook 109

\@gls@prevlevel 199, 200
\@gls@quotechar 105
\@gls@reference 19, 72
\@gls@saveentrycounter 7, 8, 14, 30
\@gls@setdefault@glslink@opts 45
\@gls@short 117
\@gls@shortpl 115, 117
\@gls@tmpb 108
\@gls@type 74, 132, 192–198
\@gls@write@entrycounts 59
\@gls@write@entryunitcounts 67
\@gls@write@entryunitcounts@do 68
\@glsabbrv@current@abbreviation 116, 130
\@glsacronymlists 69
\@glsdisp 29
\@glsdoifexistsorwarn 8, 100–103
\@glsentry 59, 68
\@glslink 47
\@glslocref 6
\@glsnumberformat 6, 30, 104, 105
\@glsorder 72
\@glspl@ 47, 60, 62
\@glsplural@ 48
\@glspunc@token 114
\@glsstyle@alttree 190
\@glsstyle@inline 190
\@glsstyle@listdotted 185
\@glstarget 47
\@glstext@ 48
\@glswidestname 191, 192, 199
\@glsxtr 21, 22
\@glsxtr@abbreviationsdef 10, 14
\@glsxtr@activate@initialtagging ..
..... 109, 111
\@glsxtr@addunitcounter 63
\@glsxtr@addunusedxrefs 18
\@glsxtr@attrval 98–103, 105
\@glsxtr@autoindex@at 105, 106
\@glsxtr@autoindex@doextra@esc 105
\@glsxtr@autoindex@encap 105–107
\@glsxtr@autoindex@esc 105, 107, 108
\@glsxtr@autoindex@escat 105, 106
\@glsxtr@autoindex@escencap ... 106, 107
\@glsxtr@autoindex@esclevel ... 106, 107
\@glsxtr@autoindex@escquote ... 105, 107
\@glsxtr@autoindex@level 106, 107
\@glsxtr@autoindex@setname 105
\@glsxtr@autoindexcrossrefs 9, 17
\@glsxtr@cat 57, 69, 109
\@glsxtr@csname 64–67
\@glsxtr@current@style 23, 201
\@glsxtr@currentunitcount 64–67
\@glsxtr@currunitcount 66, 68
\@glsxtr@declareoption 4, 9, 10, 12
\@glsxtr@defaultnoglossarywarning .. 12
\@glsxtr@deprecated@abbrstyle .....
..... 150, 151, 154, 155, 159–161
\@glsxtr@disabledflycommand 22
\@glsxtr@do@@wrindex 45, 46
\@glsxtr@do@glsdisablehyperinlist .. 44
\@glsxtr@do@style 13, 183
\@glsxtr@do@titlecaps@warn .. 99–102, 110
\@glsxtr@doabbreviationsdef 10
\@glsxtr@doaccsupp 12, 14
\@glsxtr@docdefval 8, 19
\@glsxtr@doloctag 25, 26
\@glsxtr@dorecord 6
\@glsxtr@dostylewarn 132
\@glsxtr@enabletagging 109
\@glsxtr@end@ 20
\@glsxtr@endescspch 105–108
\@glsxtr@entrycount@org@localreset . 58
\@glsxtr@entrycount@org@localunset . 58
\@glsxtr@entrycount@org@reset 58
\@glsxtr@entrycount@org@unset 58
\@glsxtr@entryunitcount@org@localreset
..... 67
\@glsxtr@entryunitcount@org@localunset
..... 66
\@glsxtr@entryunitcount@org@reset .. 66
\@glsxtr@entryunitcount@org@unset .. 66
\@glsxtr@err@undefaction 5, 7
\@glsxtr@field@linkdefs 28
\@glsxtr@format@overridefalse 104
\@glsxtr@format@overridetrue 104
\@glsxtr@foundinlist 114
\@glsxtr@full 120
\@glsxtr@fullpl 122
\@glsxtr@glossdescfont 98–100
\@glsxtr@glossnamefont 100–103
\@glsxtr@gobbleto@endescspch 108
\@glsxtr@idx@displaynumberlist 73
\@glsxtr@idx@entrynumberlist 73
\@glsxtr@ifcsstart 20
\@glsxtr@ifpunctoken 114
\@glsxtr@ifunitcounter 63
\@glsxtr@insert@dots 116
\@glsxtr@insert@dots@next 116

\@glsxtr@insertdots 117
\@glsxtr@label 18, 97, 98
\@glsxtr@loadstyles 184
\@glsxtr@noidx@displaynumberlist ... 73
\@glsxtr@noidx@entrynumberlist 73
\@glsxtr@noidx@numberlistloop 73
\@glsxtr@notfoundinlist 114
\@glsxtr@optlist 22
\@glsxtr@org@GLS@ 29
\@glsxtr@org@GLSpl@ 29
\@glsxtr@org@Gls@ 28
\@glsxtr@org@Glspl@ 29
\@glsxtr@org@Glsxtrtitlefirst . 166, 167
\@glsxtr@org@Glsxtrtitlefirstplural
..... 166, 167
\@glsxtr@org@Glsxtrtitlefull .. 167, 168
\@glsxtr@org@Glsxtrtitlefullpl 167, 168
\@glsxtr@org@Glsxtrtitlelong .. 166, 168
\@glsxtr@org@Glsxtrtitlelongpl 166, 168
\@glsxtr@org@Glsxtrtitleplural 166, 167
\@glsxtr@org@Glsxtrtitleshort . 166, 167
\@glsxtr@org@Glsxtrtitleshortpl 166, 167
\@glsxtr@org@Glsxtrtitletext .. 166, 167
\@glsxtr@org@MakeUppercase 166, 167
\@glsxtr@org@checkfirsthyper 44, 71
\@glsxtr@org@delimN 26
\@glsxtr@org@delimR 26
\@glsxtr@org@gls@ 28
\@glsxtr@org@glsdisp 29
\@glsxtr@org@glsignore 26
\@glsxtr@org@glspl@ 28
\@glsxtr@org@glsxtrtitlefirst . 166, 167
\@glsxtr@org@glsxtrtitlefirstplural
..... 166, 167
\@glsxtr@org@glsxtrtitlefull .. 166, 168
\@glsxtr@org@glsxtrtitlefullpl 166, 168
\@glsxtr@org@glsxtrtitlelong .. 166, 167
\@glsxtr@org@glsxtrtitlelongpl 166, 167
\@glsxtr@org@glsxtrtitleplural 166, 167
\@glsxtr@org@glsxtrtitleshort . 166, 167
\@glsxtr@org@glsxtrtitleshortpl 166, 167
\@glsxtr@org@glsxtrtitletext .. 166, 167
\@glsxtr@org@makeglossaries 72
\@glsxtr@org@markboth 165, 166
\@glsxtr@org@markright 165, 166
\@glsxtr@org@newacronymstyle 71
\@glsxtr@org@postdescription 111
\@glsxtr@org@setacronymstyle 70, 71
\@glsxtr@orgprintglossary 22
\@glsxtr@orgwarndep 115
\@glsxtr@p@acrlong@ 48
\@glsxtr@p@acrlongpl@ 48
\@glsxtr@p@acrshort@ 48
\@glsxtr@p@acrshortpl@ 48
\@glsxtr@p@long@ 48
\@glsxtr@p@longpl@ 48
\@glsxtr@p@plural@ 47
\@glsxtr@p@short@ 47
\@glsxtr@p@shortpl@ 48
\@glsxtr@p@text@ 47
\@glsxtr@pagestag 25, 26
\@glsxtr@pagetag 25, 26
\@glsxtr@prevunitcount 66
\@glsxtr@record 7, 8, 28, 29
\@glsxtr@redefstyles 13, 183
\@glsxtr@reg@glosslist 72–74
\@glsxtr@saveprelocttag 25, 26
\@glsxtr@setentrycountunsetattr 57
\@glsxtr@setentryunitcountunsetattr 69
\@glsxtr@setupshortcuts 11, 12, 14
\@glsxtr@swaptwo 115
\@glsxtr@tag 109
\@glsxtr@taggingcs 109
\@glsxtr@thisloctag 26
\@glsxtr@tmp 13
\@glsxtr@type 98
\@glsxtr@unitcountlist 63, 64
\@glsxtr@usesee 17
\@glsxtr@warn@onexistsordo 5, 7, 8
\@glsxtr@warn@undefaction 5, 7, 8
\@glsxtrdocdeffalse 19
\@glsxtrindexcrossrefsfalse 9
\@glsxtrindexcrossrefstrue 9
\@glsxtrinmark 165, 166
\@glsxtrlong 48, 125
\@glsxtrlongpl 48, 128
\@glsxtrnotinmark 165, 166
\@glsxtrp 53
\@glsxtrp@opt 51
\@glsxtrpl 21, 22
\@glsxtrpostloctag 25, 27
\@glsxtrpreloctag 25, 27
\@glsxtrshort 47, 123
\@glsxtrshortpl 48, 126
\@glsxtrundeftag 5, 14
\@gobble 5, 7, 9, 30, 116
\@gobbletwo 115
\@ifnextchar 46

\@ifpackageloaded 4, 9, 82, 98, 100, 102, 104, 182
\@ifstar 20, 46, 109
\@ifundefined 182
\@input@ 76
\@istfilename 72
\@makeglossary 72, 73
\@mfu@domakefirstuc 110
\@mfu@nocaplist 110
\@ne 59, 68
\@newglossaryentry@defcounters .. 58, 65
\@newglossaryentryposthook 7, 42
\@newglossaryentryprehook 7, 42
\@nnil 105, 106, 108, 114, 116
\@no@makeglossaries 80
\@onelevel@sanitize 22
\@onlypreamble 23, 26, 68, 81, 104, 106, 107, 109
\@print@unsrt@glossary 81
\@printgloss@setsort 74
\@printglossary 22, 81
\@sGlsXtrEnableOnTheFly 20
\@secondofthree 31–33, 38–41, 43, 121, 122, 124, 125, 127, 129
\@secondoftwo 30, 33–41, 44, 47, 71, 115, 120, 121, 123–129, 141, 166, 167
\@thirdofthree 31–33, 38–41, 43, 121, 123, 124, 126, 128, 129
\@thirdoftwo 33–37
\@warn@nomakeglossaries 77
\@xdy@main@language 76
\@xdylanguage 76, 77

\␣ 78, 79

A

\AB 10
\Ab 10
\ab 10
abbreviation styles:
  long-noshort 159, 160
  short 144
\abbreviationsname 9
\abbrvpluralsuffix 117, 135, 137, 140, 142–144, 146, 148–155, 163, 164
\ABP 10
\Abp 10
\abp 10
\ACRfullfmt 70
\Acrfullfmt 70
\acrfullfmt 70
\ACRfullplfmt 70
\Acrfullplfmt 70
\acrfullplfmt 70
\acronymentry 70
\acronymfont 38–41, 50, 71
\acronymname 9
\acronymsort 70
\acronymtype 10, 69–71
\acrpluralsuffix 70
\actualchar 107
\addtolength 200
\advance 59, 68
\AF 10
\Af 10
\af 10
\AFP 10
\Afp 10
\afp 10
\AL 10
\Al 10
\al 10
\ALP 10
\Alp 10
\alp 10
\AnyTrackedLanguages 182
\appto 7, 13, 17, 42, 45, 46, 58, 65, 104, 114, 116
\AS 10
\As 10
\as 10
\ASP 10
\Asp 10
\asp 10
\AtBeginDocument 14, 24
\AtEndDocument 17, 59, 67, 76, 77

B

babel package 105, 106, 113
\begin 61, 78, 79, 81, 185–189, 199
\begingroup 6

C

category attributes:
  discardperiod 113
  entrycount 56, 57, 59, 68, 69
  firstuc 102
  glossdesc 98
  glossdescfont 98
  glossname 100
  glossnamefont 100, 102

headuc 168
indexname 105
indexonlyfirst 45
insertdots 117
nohyper 44
nohyperfirst 31–33
regular 27, 62, 135, 137–139, 141, 144–146, 156, 158, 162, 164
\cGLS 10, 57, 69
\cGls 10, 57, 69
\cgls 10, 57, 69
\cGLSformat 61
\cGlsformat 60
\cglsformat 60, 62
\cGLSpl 10, 57, 69
\cGlspl 10, 57, 69
\cglspl 10, 57, 69
\cGLSplformat 61
\cGlsplformat 61
\cglsplformat 60, 62
\columnwidth 24
\count@ 59, 68
\cs 61, 199
\csdef 42, 43, 58, 64, 65, 67, 93, 132–134, 141, 185
\csedef 65
\csgdef 19, 25, 58, 59, 64, 66, 67
\csletcs 134
\csname . 6, 23, 27, 30, 38–43, 51, 64, 65, 76, 77, 79–81, 98, 115, 120–130, 134, 135, 199
\csuse 25, 42, 43, 53, 54, 63–67, 93, 104, 111, 112, 132–134, 192–195
\csxdef 17, 64, 67
\CurrentOption 13, 184
\CurrentTrackedTag 182
\CustomAbbreviationFields 118, 135–139, 141, 143, 144, 146, 147, 156, 157, 159, 162, 163

D

\DeclareAcronymList 69
\DeclareOption 4, 184
\DeclareOptionX 4, 13
\def 6–8, 14, 17, 18, 20–22, 24, 27–41, 46–51, 60–62, 69, 73–75, 81, 104–110, 114–117, 120–129, 132, 199, 200
\define@boolkey 9, 44
\define@choicekey 5, 7, 8, 11, 12
\define@key 7, 13, 42, 115
\DefineAcronymSynonyms 11, 12
\delimN 26
\delimR 26
\detokenize 20
\dimen@ 72, 192–199
\dimen@i 193–195
\dimen@ii 193–195
\dimexpr 24, 191
\disable@keys 10, 14, 19
\do 13, 18, 57, 69, 72, 74, 81, 97, 109
\do@gls@link@checkfirsthyper 28, 29, 38–41, 120–129
\do@glsdisablehyperinlist 45
doc package 107
\DTLifinlist 73, 74

E

\eappto 13, 105, 184
\edef . . . 6, 15, 30, 44, 63–67, 72, 74, 76, 77, 98–101, 103, 105, 106, 108, 115, 193–195
\eglssetwidest 192–199
\else 6, 7, 9, 10, 12, 19, 20, 25, 27, 45, 60, 72, 74, 78, 80, 104–106, 108, 114, 116, 123–129, 136, 138, 140–147, 163, 164, 186–190, 200, 201
\emph 155
\encapchar 108
\end 78, 79, 81, 185–189, 199
\endcsname 6, 23, 27, 30, 38–43, 51, 64, 65, 76, 77, 79–81, 98, 115, 120–130, 134, 135, 199
\endgroup 6
entry categories:
abbreviation 130
general 92, 94
index 96
\epreto 105
\equal 80
etoolbox package 4
\expandafter 13, 17, 18, 20–22, 42, 43, 46, 51, 62, 63, 73, 74, 81, 98, 101, 102, 105, 107, 108, 114, 117
\expandonce 70, 105, 106

F

\fi 5–10, 12, 17, 19, 20, 23–25, 27, 45, 59, 60, 67, 68, 72, 74, 77, 78, 80, 104–106, 108, 114, 116, 123–129, 136, 138, 140–147, 163, 164, 186–190, 192–201
first use 202
flag 202
text 202

\firstacronymfont 71, 72
\footnote 139
\forallglossaries ... 18, 81, 96, 98, 192–198
\forallglsentries 59, 68
\ForEachTrackedDialect 182
\forglsentries 18, 96, 98, 192–198
\forlistcsloop 68
\forlistloop 75, 110
\futurelet 114

G

\gdef 26, 106, 107
\Genacrfullformat 70
\genacrfullformat 70
\GenericAcronymFields 70
\Genplacrfullformat 70
\genplacrfullformat 70
\glo@name 101, 102
glossaries package 184
glossaries-accsupp package 12, 14, 82
glossaries-extra package 2
glossaries-extra-stylemods package . 12, 111, 183
\GlossariesExtraWarning 5, 9, 20, 22, 71, 78, 81, 98–101, 103, 109, 110, 134
\GlossariesExtraWarningNoLine . 9, 59, 68
\GlossariesWarning 25, 75, 76, 132
\GlossariesWarningNoLine 73, 77
glossary styles:
  alttree 190, 191, 199
  inline 190
  listdotted 185
  listdottedstyle 185
  sublistdotted 185
glossary-long package 186
glossary-longbooktabs package 186
glossary-mcols package 184
glossary-tree package 184
\glossaryentrynumbers 27
\glossaryheader 81, 185–189, 200
\glossarypostamble 81
\glossarypreamble 81
\glossarysection 79, 81
\glossarytitle 79, 81
\glossarytoctitle 79, 81
\glossentry 185–189, 200
\glossentrydesc 185–191
\glossentryname 185–189, 200
\glossentrysymbol 186–191
\glossxtrsetpopts 111
\GLS 57, 69
\Gls 21, 57, 69
\gls 21, 23, 57, 69, 78
\gls@assign@field 7, 42
\gls@checkseeallowed 19, 73
\gls@codepage 77
\gls@defdocnewglossaryentry 58, 65
\gls@defglossaryentry 21, 22
\gls@save@numberlist 24, 25, 27
\gls@tmplen 192–198, 200
\glsabbrvdefaultfont 119, 136, 137, 140, 142–144, 146
\glsabbrvemfont 155–161
\glsabbrvfont 49, 71, 119, 123, 124, 127, 128, 130, 131, 135–144, 146–164
\glsabbrvuserfont 162–164
\glsabbvfont 135, 137, 139, 141, 156, 157, 162, 164
\GLSaccessdesc 34
\Glsaccessdesc 34, 99, 108
\glsaccessdesc 34, 99, 112
\GLSaccessdescplural 34
\Glsaccessdescplural 34
\glsaccessdescplural 34
\GLSaccessfirst 32
\Glsaccessfirst 31
\glsaccessfirst 31
\GLSaccessfirstplural 33
\Glsaccessfirstplural 33
\glsaccessfirstplural 32
\Glsaccesslong 40, 118, 126, 136, 143, 146, 163
\glsaccesslong 40, 118, 125, 126, 136, 138, 140, 142, 143, 145–147, 163, 164
\Glsaccesslongpl 41, 119, 129, 136, 143, 146, 163
\glsaccesslongpl 41, 118, 128, 129, 136, 138, 140–143, 145–147, 163, 164
\GLSaccessname 33
\Glsaccessname 33
\glsaccessname 33
\GLSaccessplural 32
\Glsaccessplural 32
\glsaccessplural 32
\Glsaccessshort 38, 124, 130, 138, 140, 142, 145, 164
\glsaccessshort 38, 118, 123, 124, 130, 131, 136, 138, 140, 142–146, 163, 164
\Glsaccessshortpl 39, 127, 130, 138, 140–142, 145, 164

\glsaccessshortpl 39, 118, 119, 127, 128, 130, 136, 138, 140, 142–146, 163, 164
\GLSAccesssymbol 35
\Glsaccesssymbol 35, 109
\glsaccesssymbol 35, 108, 112
\GLSAccesssymbolplural 35
\Glsaccesssymbolplural 35
\glsaccesssymbolplural 35
\GLSAccesstext 31
\Glsaccesstext 31
\glsaccesstext 30
\glsacrshortcutstrue 11, 12
\glsacspacemax 72
\glsadd 18, 78
\glsaddstoragekey 92
\glsbackslash 20
\glscapscase 30–41, 43, 120–131
\glscategory ... 27, 30, 44, 49, 93–95, 98–104, 108, 109, 111, 112, 120–124, 126–128
\glscategorylabel 44, 115–117, 141
\glsclosebrace 79, 80
\glscurrententrylabel 25, 26, 81, 104, 110, 111
\glscurrentfieldvalue 161
\glscustomtext 28, 38–41, 120–130, 132
\glsdefaulttype 9, 78, 81
\glsdescriptionaccessdisplay 86, 99
\glsdescriptionpluralaccessdisplay 86, 87
\glsdescwidth 185, 187–189
\glsdetoklabel 15–20, 30, 58, 59, 64–68, 75, 76, 98, 101, 102, 193–195
\glsdisplaynumberlist 73, 75
\glsdohyperlink 47
\glsdoifexists 8, 17, 28, 29, 38–41, 75, 76, 120–129
\glsdoifexistsordo 29
\glsdoifexistsorwarn ... 9, 98, 99, 108, 109
\glsdonohyperlink 47
\glsdosanitizesort 74
\glsenableentrycount 57, 59, 67
\glsenableentryunitcount 59, 68
\glsentrycurrcount 58, 59, 66
\Glsentrydesc 86, 91, 99
\glsentrydesc 86, 91, 100
\Glsentrydescplural 87, 91
\glsentrydescplural 86, 87, 91
\Glsentryfirst 63, 84, 90
\glsentryfirst 63, 84, 90, 179
\Glsentryfirstplural 63, 84, 90
\glsentryfirstplural ... 63, 84, 85, 90, 179
\Glsentryfull 70
\glsentryfull 70
\Glsentryfullpl 70
\glsentryfullpl 70
\glsentryitem 185–189, 200
\Glsentrylong 50, 63, 88, 92
\glsentrylong 49, 50, 63, 88, 92, 141, 180
\Glsentrylongpl 50, 51, 63, 89, 92
\glsentrylongpl .. 50, 51, 63, 89, 92, 180, 181
\Glsentryname 82, 89, 100, 102, 103
\glsentryname 82, 89, 105, 192–199
\glsentrynumberlist 73, 76, 197–199
\Glsentryplural 83, 90
\glsentryplural 83, 90, 178
\glsentryprevcount 58, 60, 66
\glsentryprevmaxcount 66
\glsentryprevtotalcount 66
\Glsentryshort 49, 50, 87, 91
\glsentryshort ... 49, 50, 72, 87, 91, 176, 177
\Glsentryshortpl 49, 50, 88, 92
\glsentryshortpl . 49, 50, 88, 91, 92, 176, 177
\Glsentrysymbol 85, 90
\glsentrysymbol 85, 90, 196, 197
\Glsentrysymbolplural 86, 91
\glsentrysymbolplural 85, 86, 91
\Glsentrytext 83, 89
\glsentrytext 82, 83, 89, 90, 177, 178
\Glsentryuseri 36
\glsentryuseri 36
\Glsentryuserii 36
\glsentryuserii 36
\Glsentryuseriii 36
\glsentryuseriii 36
\Glsentryuseriv 37
\glsentryuseriv 37
\Glsentryuserv 37
\glsentryuserv 37
\Glsentryuservi 37
\glsentryuservi 37
\glsfieldxdef 97, 98
\glsfindwidesttoplevelname 192
\GLSfirst 171, 172
\Glsfirst 172
\glsfirst 171
\glsfirstabbrvdefaultfont 119, 136, 137, 140, 142–144, 146
\glsfirstabbrvemfont 156–161

\glsfirstabbrvfont 71, 118, 119, 135–146, 148–164
\glsfirstabbrvuserfont 163, 164
\glsfirstaccessdisplay 84
\glsfirstlongdefaultfont 136, 137, 143, 145, 146
\glsfirstlongemfont 156–158, 160
\glsfirstlongfont ... 118, 119, 135–138, 140, 142, 143, 145–147, 156–160, 162–164
\glsfirstlongfootnotefont 139–142
\glsfirstlonguserfont 163, 164
\GLSfirstplural 172
\Glsfirstplural 172, 173
\glsfirstplural 172
\glsfirstpluralaccessdisplay 84, 85
\glsforeachincategory 132
\glsgenentryfmt 27
\glsgetattribute 60, 64–66, 98–101, 103, 105
\glsgetcategoryattribute 93
\glsgetwidestname 191
\glsgroupheading 185–189, 200
\glsgroupskip 186–190, 201
\glshasattribute 59, 60, 64–68, 98–104, 135, 137–139, 141, 156, 158, 162, 164
\glshascategoryattribute 94
\glshypernumber 104
\glsifattribute 31, 44, 45, 52, 96, 99–102, 110, 113, 168–176
\glsifcategory 96
\glsifcategoryattribute ... 44, 94, 95, 117
\glsifnotregular 30
\glsifnotregularcategory 95
\glsifplural . 30, 32–35, 38–41, 113, 120–131
\glsifregular 27, 30, 63
\glsifregularcategory 95
\glsignore 26
\glsinlinedescformat 190
\glsinlinesubdescformat 190
\glsinsert 30, 38–41, 120–131
\glskeylisttok 70, 116, 118
\glslabel ... 15, 27, 44, 71, 112, 130, 131, 141
\glslabeltok 70, 116, 118, 135–139, 141, 143, 144, 146, 147, 156–159, 162–164
\glsletentryfield 105
\glslink 70
\glslink options
format 104
hyper 165
noindex 6, 44, 165
\glslinkcheckfirsthyperhook 44
\glslinkvar 46
\glslistdottedwidth 185
\glslongaccessdisplay 88
\glslongdefaultfont 120, 136, 137, 139, 143, 145, 146
\glslongemfont 155–160
\glslongfont 50, 120, 125, 126, 128, 129, 136, 137, 140, 142, 143, 145, 146, 156–158, 160, 163, 164
\glslongfootnotefont 139, 140, 142
\glslongpltok 117, 118, 135–139, 146, 147, 156, 157, 159, 162, 164
\glslongpluralaccessdisplay 89
\glslongtok .. 70, 116, 118, 135–139, 141, 143, 144, 146, 147, 156, 157, 159, 162, 163
\glslonguserfont 162–164
\glsnameaccessdisplay ... 82, 100, 101, 103
\glsnamefont 100, 102, 103
\glsnoidxdisplayloc 75, 76
\glsnoidxdisplayloclisthandler 75
\glsnoidxloclist 76
\glsnoidxnumberlistloophandler 75
\glsnonumberlistfalse 25
\glsnonumberlisttrue 25
\glsnumberlistloop 73
\glsnumlistlastsep 75
\glsnumlistsep 75
\glsopenbrace 79, 80
\glsorder 72
\glspagelistwidth 185, 187–189
\GLSpl 57, 69
\Glspl 22, 57, 69
\glspl 22, 57, 69
\GLSplural 170, 171
\Glsplural 171
\glsplural 170, 171
\glspluralaccessdisplay 83
\glspluralsuffix 116, 120, 135, 137, 140, 142–144, 146, 148, 152, 162
\glspostdescription 111, 185–191
\glspostinline 190
\glspostlinkhook . 28, 29, 38–42, 51, 120–129
\glsprestandardsort 73
\glsreseteentrylist 81
\glsseeformat 17, 75, 76
\glssetabbrvfmt 27, 30, 49, 98–103, 108, 109, 120–124, 126–128

\glssetattribute 135, 137–139, 141, 143, 144, 146, 147, 156, 158, 159, 162, 164
\glssetcategoryattribute 57, 69, 71, 93, 94, 96, 97, 109
\glsshortaccessdisplay 87
\glsshortpltok 117, 118, 135–139, 141, 143, 144, 156, 157, 162, 164
\glsshortpluralaccessdisplay 88
\glsshorttok 70, 116–118, 135–139, 141, 143, 144, 147, 156, 157, 159, 162, 163
\glssubentryitem 185–190, 200
\glssymbolaccessdisplay 85
\glssymbolpluralaccessdisplay ... 85, 86
\glstarget 185–190, 200
\GLStext 170
\Glstext 170
\glstext 170
\glstextaccessdisplay 82, 83
\glstextformat 29
\glstextup 148
\glstreeindent 199, 200
\glstreenamebox 200
\glstreenamefmt 191–200
\glstype 38–42, 120–129
\glsunset 18, 60, 61
\glswrite 72
\glswriteentry 6
\Glsxtr 22
\glsxtr 22
\glsxtr@@do@wrglossary 6–8
\glsxtr@addloclistfield 8
\glsxtr@addunused 18
\glsxtr@applyabbrvfmt 130
\glsxtr@applyabbrvstyle 115, 116, 132
\glsxtr@dooption 4, 9, 13, 14
\glsxtr@headentry@p 52, 53
\glsxtr@ifnextpunc 114
\glsxtr@ifpunctoken 114
\glsxtr@indexonly@saveentrycounter 7, 8, 14
\glsxtr@keylist 21, 22
\glsxtr@next 114
\glsxtr@orgmakenoidxglossaries .. 18, 19
\glsxtr@punclist 114
\glsxtr@record 6
\glsxtr@resource 81
\glsxtr@saveentrycounter 6, 7
\glsxtr@setup@record 7, 8, 14
\glsxtr@usesee 17
\glsxtr@warnonexistsordo 5, 7, 8, 16
\glsxtrabbrvfootnote 139–141
\glsxtrabbrvtype 9, 10, 118
\glsxtraddallcrossrefs 17
\glsxtrAltTreeIndent 191
\glsxtralttreeInit 199
\glsxtrAltTreePar 190
\glsxtrAltTreeSetHangIndent ... 191, 200
\glsxtrAltTreeSetSubHangIndent 200
\glsxtralttreeSubSymbolDescLocation 200
\glsxtralttreeSymbolDescLocation 191, 200
\glsxtrassignfieldfont 30–37
\glsxtrcat 21, 22
\glsxtrchecknohyperfirst 31–33
\glsxtrComputeTreeIndent 200
\glsxtrComputeTreeSubIndent 200
\GlsXtrDefineAbbreviationShortcuts 11, 12
\GlsXtrDefineOtherShortcuts 11, 12
\glsxtrdiscardperiod 112
\glsxtrdoautoindexname 45, 46, 104
\glsxtrdopostpunc 141
\glsxtrdowrglossaryhook 45, 46
\GlsXtrEnableEntryCounting 69
\GlsXtrEnableEntryUnitCounting 57
\GlsXtrEnableOnTheFly 20, 23
\glsxtrfieldtitlecase 99–102
\glsxtrfieldtitlecasecs 98
\glsxtrfirstscfont 148–151
\glsxtrfirstsmfont 152–155
\GlsXtrFormatLocationList 25, 27, 197–199
\GLSxtrfull 10, 174, 175
\Glsxtrfull 10, 175, 176
\glsxtrfull 10, 175
\GlsxtrfullformatGlsxtrfullformat 119, 131, 133, 136, 138, 140, 142, 144, 145, 147, 163, 164
\glsxtrfullformat 119, 131, 133, 136–138, 140, 142, 144–146, 163, 164
\GLSxtrfullpl 10, 175, 176
\Glsxtrfullpl 10, 176
\glsxtrfullpl 10, 175
\GlsxtrfullplformatGlsxtrfullplformat 119, 131, 133, 136, 138, 140, 142, 144, 145, 147, 163, 164
\glsxtrfullplformat 131, 133, 136, 138, 140, 142, 144, 145, 147, 163, 164
\glsxtrfullsep 118, 119, 135–138, 140–143, 145, 146, 156, 157, 161
\glsxtrgenabbrvfmt 27

\Glsxtrheadfirst 167
\glsxtrheadfirst 167
\Glsxtrheadfirstplural 167
\glsxtrheadfirstplural 167
\Glsxtrheadfull 167
\glsxtrheadfull 167
\Glsxtrheadfullpl 167
\glsxtrheadfullpl 167
\Glsxtrheadlong 167
\glsxtrheadlong 167
\Glsxtrheadlongpl 167
\glsxtrheadlongpl 167
\Glsxtrheadplural 167
\glsxtrheadplural 167
\Glsxtrheadshort 167
\glsxtrheadshort 167
\Glsxtrheadshortpl 167
\glsxtrheadshortpl 167
\Glsxtrheadtext 167
\glsxtrheadtext 167
\glsxtrifcounttrigger 60, 61
\glsxtrifemptyglossary 79
\glsxtrifindexing 45
\glsxtrifinmark 53–55, 166, 167
\glsxtrifnextpunc 114, 115
\glsxtrifperiod 113
\glsxtrifwasfirstuse 30–33, 38–41, 44, 71, 112, 113, 121, 123–129, 141
\Glsxtrinlinefullformat 119, 121, 133, 140, 142, 143, 145, 146, 181
\glsxtrinlinefullformat 119–121, 133, 140, 142–146, 181
\Glsxtrinlinefullplformat 119, 122, 133, 140, 142, 143, 145, 146, 182
\glsxtrinlinefullplformat 119, 122, 123, 133, 140, 142, 143, 145, 146, 181
\glsxtrinsertinsidefalse 135
\GLSxtrlong 10, 173, 174
\Glsxtrlong 10, 174
\glsxtrlong 10, 173
\GLSxtrlongpl 10, 173, 174
\Glsxtrlongpl 10, 174
\glsxtrlongpl 10, 173
\glsxtrlongshortdescsort 136
\glsxtrmarkhook 165
\glsxtrnewabbrevpresetkeyhook 117
\glsxtrnewnumber 11
\glsxtrnewsymbol 11
\glsxtrNoGlossaryWarning 12, 76
\GlsXtrNoGlsWarningAutoMake 80
\GlsXtrNoGlsWarningBuildInfo 80
\GlsXtrNoGlsWarningCheckFile 80
\GlsXtrNoGlsWarningEmptyMain 80
\GlsXtrNoGlsWarningEmptyNotMain 80
\GlsXtrNoGlsWarningEmptyStart 80
\GlsXtrNoGlsWarningHead 79
\GlsXtrNoGlsWarningMisMatch 80
\GlsXtrNoGlsWarningNoOut 80
\GlsXtrNoGlsWarningTail 80
\glsxtrorg@ifKV@glslink@hyper 28
\GLSxtrp 52
\Glsxtrp 52
\glsxtrp 51, 54
\Glsxtrpl 22
\glsxtrpl 22
\glsxtrpostdescription 96, 111, 190
\glsxtrpostlink 112
\glsxtrpostlinkendsentence 112
\glsxtrpostlinkhook 112
\glsxtrpostlocalreset 57, 58, 67
\glsxtrpostlocalunset 56, 58, 66
\glsxtrpostnamehook 101–103
\GlsXtrPostNewAbbreviation 118, 133, 135, 137–139, 141, 143, 144, 146, 147, 156, 158, 159, 162, 164
\glsxtrpostreset 56, 58, 66
\glsxtrpostunset 56, 58, 66
\glsxtrprotectlinks 47
\glsxtrregularfont 27, 30
\glsxtrrestoremarkhook 165, 166
\glsxtrscfont 148–151
\glsxtrscsuffix 148–151
\GlsXtrSetActualChar 107
\GlsXtrSetEncapChar 108
\GlsXtrSetEscChar 107
\GlsXtrSetLevelChar 107
\glsxtrsetpopts 51
\glsxtrsetupfulldefs 120–123, 141
\GLSxtrshort 10, 55, 56, 168, 169
\Glsxtrshort 10, 169
\glsxtrshort 10, 168
\GLSxtrshortpl 10, 168, 169
\Glsxtrshortpl 10, 169
\glsxtrshortpl 10, 168, 169
\glsxtrsmfont 151–155
\glsxtrsmsuffix 152–155
\glsxtrtagfont 110
\Glsxtrtitlefirst 166, 167, 179

\glsxtrtitlefirst 166, 167, 179
\Glsxtrtitlefirstplural 166, 167, 179, 180
\glsxtrtitlefirstplural 166, 167, 179
\Glsxtrtitlefull 167, 168, 181
\glsxtrtitlefull 166–168, 181
\Glsxtrtitlefullpl 167, 168, 182
\glsxtrtitlefullpl 166–168, 181, 182
\Glsxtrtitlelong 166–168, 180
\glsxtrtitlelong 166, 167, 180
\Glsxtrtitlelongpl 166–168, 181
\glsxtrtitlelongpl 166, 167, 180
\Glsxtrtitleplural 166, 167, 178
\glsxtrtitleplural 166, 167, 178
\Glsxtrtitleshort 166, 167, 177
\glsxtrtitleshort 166, 167, 176
\Glsxtrtitleshortpl 166, 167, 177
\glsxtrtitleshortpl 166, 167, 176, 177
\Glsxtrtitletext 166, 167, 178
\glsxtrtitletext 166, 167, 177, 178
\glsxtrtreetopindent 191, 199
\glsxtrundefaction 5, 7, 8, 15, 16
\glsxtrundeftag 14
\GlsXtrUseAbbrStyleFmts .....
..... 137, 139, 147–161, 163, 165
\GlsXtrUseAbbrStyleSetup 148–161, 163, 164
\glsxtruserfield 161
\glsxtruserparen 162–164
\glsxtrusersuffix 163, 164
\glsxtruseseeformat 17
\GlsXtrWarnDeprecatedAbbrStyle 115, 134
\GlsXtrWarning 21, 22

H

\hangindent 191, 199
\hbox 185
\hfill 185
\hsize 24
\hss 185
\hyperlink 47
\hyperpage 104
hyperref package 47, 104, 165, 176

I

\if 20
\if@glsxtr@format@override 105
\if@glsxtrdocdefrestricted 19
\if@glsxtrindexcrossrefs 9, 17
\ifblank 13, 21, 22, 72
\ifcase 5, 7, 11, 12, 19
\ifcsdef 6, 42, 43, 51–55, 64,
98–101, 103, 112, 115, 130, 133, 185–189
\ifcsempty 81
\ifcsstring 15, 94, 132
\ifcsundef 19, 23, 25,
47, 58, 64–67, 76, 94, 132–134, 184, 192, 199
\ifcsvoid 93
\ifdef 11, 16, 23, 24, 44, 53–55, 75, 76,
96, 97, 107, 108, 111, 161, 176–182, 185, 190
\ifdefempty ... 17, 57, 69, 72, 74, 81, 109, 130
\ifdefequal 80
\ifdefstring 105, 110
\ifdefvoid 17, 18, 63
\ifdim 24, 72, 192–199
\IfFileExists 13, 76, 80, 184
\ifglossaryexists 15, 16
\ifglsacronym 10, 80
\ifglsautomake 74, 80
\ifglsentryexists .....
..... 15, 16, 21, 22, 25, 30, 94, 111, 112
\ifglsfieldeq 92
\ifglshasfield 161
\ifglshaslong 63
\ifglshasparent 192–195
\ifglshasshort 27, 30
\ifglshassymbol 112, 191
\ifglsindexonlyfirst 45
\ifglsnogroupskip 186–190, 201
\ifglsnonumberlist 27
\ifglssanitizesort 74
\ifglsused 18, 44,
45, 59, 68, 71, 130, 192, 193, 195, 197, 198
\ifglsxindy 76, 78
\ifglsxtrinsertinside .....
.... 123–129, 136, 138, 140–147, 163, 164
\ifHy@hyperindex 104
\ifinlistcs 19
\ifKV@glslink@hyper 28
\ifKV@glslink@noindex 6, 7, 45
\ifnum 8, 59, 60, 67, 68, 200
\ifthenelse 80
\IfTrackedLanguageFileExists 182
\ifundef 47, 72, 109, 110
\ifx 23, 25, 105, 106, 108, 114, 116, 201
\immediate 59, 68, 76, 77
\index 105
\indexspace 201
\input 182
\InputIfFileExists 81

\istfilename 72
\item 78, 79, 185

J

\jobname 76, 78–80

K

\key@ifundefined 7, 42
\KV@glslink@hyperfalse 31, 44, 47
\KV@glslink@noindexfalse 44, 45
\KV@glslink@noindextrue 47

L

\LaTeX 78, 79
\leaders 185
\let 4–11, 14, 18, 19, 22, 23, 25, 26, 28–41, 43, 44, 46–48, 51, 57–59, 66, 67, 69–73, 75, 76, 99, 100, 102–106, 109–111, 114–117, 120–129, 141, 165–168, 190, 192
\letabbreviationstyle 141, 142, 144, 145, 147, 149, 150, 153, 158, 159
\letcs 17, 18, 42, 75, 76, 98–103
\levelchar 107
\listadd 63
\listbreak 110
\listcseadd 65
\listcsgadd 19
\listcsxadd 64
\loadglsentries 19, 78

M

\MakeAcronymsAbbreviations 71
\makeatletter 76, 106
\makeatother 106
\makebox 185, 200
\makefirstuc 110
\makeglossaries 72, 77–80
\makeglossary 73
makeindex **202**
makeindex 72
\makenoidxglossaries 78
\MakeTextUppercase 167
\MakeUppercase 166, 167
\marg 199
\markboth 166
\markright 166
\maxdimen 24
\mbox 200
\medskip 80
\MessageBreak 19, 23, 59, 68, 74, 132
mfirstuc package 109, 110
\mfirstucMakeUppercase 31–41, 43, 49–52, 55, 56, 62, 70, 82–92, 101, 102, 121, 123, 124, 126, 128–131
\mfu@checkword@arg 110
\mfu@checkword@do 110

N

\NeedsTeXFormat 4, 184
\new@glossaryentry 19, 74
\new@ifnextchar 42, 43, 62, 113, 120–129
\newabbr 10
\newabbreviation 10, 71
\newabbreviationhook 118
\newabbreviationstyle 135–139, 141, 143, 144, 146–164
\newacronym 69, 71
\newacronymhook 70
\newacronymstyle 71
\newcommand 4–18, 20–23, 25–27, 30, 31, 42, 43, 45–47, 51, 53–58, 60, 62–66, 68, 69, 71, 72, 75–79, 81–98, 104–116, 118–130, 132–134, 136, 139, 147, 148, 151, 152, 155, 161, 162, 166–182, 184, 190–192, 199
\newcount 8
\newentry 11
\newglossary 9, 72, 73
\newglossaryentry 11, 19, 58, 65, 70, 96, 97, 118
\newglossaryentry options
  desc 86, 91
  descplural 86, 87, 91
  first ... 47, 83, 84, 90, 135, 171–173, 178, 202
  firstplural 84, 90, 135, 172, 179, 202
  long 88, 92, 180
  longplural 89, 92, 180
  name 82, 89, 105
  plural 83, 90, 135, 170, 171, 178
  see 9, 17, 19, 73
  short 87, 91, 116
  shortplural 88, 92, 116
  symbol 85, 90
  symbolplural 85, 86, 91
  text 47, 82, 83, 89, 90, 135, 136, 169, 170, 177
\newif 104, 135
\newlength 191
\newnum 11
\newrobustcmd 42, 43, 51, 52, 62, 109, 110, 120–129, 166, 168–176, 192–198
\newsym 11

\newterm 96
\newtoks 116
\newwrite 72
\NoCaseChange 53–55, 168–176
\noexpand ... 13, 70, 76, 77, 105, 106, 118, 184
\nofiles 79
\noindent 80
\nopostdesc 21, 22, 96
\nr 5, 7, 8, 11, 12
\ns@GLSxtrfull 121
\ns@Glsxtrfull 121
\ns@glsxtrfull 120
\ns@GLSxtrfullpl 123
\ns@Glsxtrfullpl 122
\ns@glsxtrfullpl 122
\ns@GLSxtrlong 126
\ns@Glsxtrlong 125
\ns@glsxtrlong 125
\ns@GLSxtrlongpl 129
\ns@Glsxtrlongpl 128, 129
\ns@glsxtrlongpl 128
\ns@GLSxtrshort 124
\ns@Glsxtrshort 124
\ns@glsxtrshort 123
\ns@GLSxtrshortpl 127
\ns@Glsxtrshortpl 127
\ns@glsxtrshortpl 126
\null 12
\number 64–67
\numexpr 64, 65, 67

O

\or 5, 7, 8, 11, 12, 19
\org@glossaryentrynumbers 24, 25

P

\p@gls@hyp@opt 46
package options:
  abbreviations 9, 10
  accsupp 12, 82
  acronym 10
  automake 74, 78
  docdef 8, 18, 19, 58, 65
    false 19
    restricted 9
    true 19
  nonumberlist 24
  numbers 10
  record 6, 7, 28, 80
  shortcuts 11
    all 11
    false 11
    none 11
    true 11
  style 13
  stylemods 13
  symbols 10, 97
  undefaction 15, 16
    error 5
    warn 5
\PackageError . 5, 13, 19, 22, 23, 42, 43, 51, 52, 57–59, 65, 67, 69, 71–74, 132–134, 184
\PackageWarning 9
\PackageWarningNoLine 9
\par 79, 80, 190, 191, 200
\parindent 191, 200
\PassOptionsToPackage 4
\preto 45
\printabbreviations 9
\printglossaries 73, 79
\printglossary 9, 73, 79
\printglossary options
  nonumberlist 27
\printnoidxglossaries 79
\printnoidxglossary 79
\printnumbers 11, 97
\printsymbols 11, 97
\printunsrtglossary 81
\ProcessOptions 184
\ProcessOptionsX 13
\protect . 53–56, 89–92, 118, 119, 135–141, 143, 144, 146–157, 159, 162–164, 168–176
\protected@csedef 191
\protected@csxdef 191
\protected@edef 23, 70, 105, 118
\protected@write 6, 26, 72, 81
\providecommand 9, 26, 43, 59, 68, 72, 76, 77, 184
\ProvidesFile 182
\ProvidesPackage 4, 184

Q

\quotechar 107

R

\raggedright 187, 189
\relax 5, 7, 8, 10–12, 14, 19, 23, 24, 26, 29, 46, 51, 59, 60, 67, 68, 73, 75, 105, 106, 108, 110, 112, 116, 117, 192–201
relsize package 151

\renewcommand 7–16, 19, 20, 22, 23, 25–29, 42, 44, 45, 47, 51, 56–59, 63, 65–67, 69–74, 76, 77, 96, 98–103, 108–112, 119, 133, 135–165, 185–190, 200, 201
\renewenvironment 185–189, 199
\renewglossarystyle 185–189, 199
\renewrobustcmd 29
\RequireGlossariesExtraLang 182
\RequirePackage 4, 12–14, 184
\reserved@a 114
\reserved@b 114
\reserved@d 114
\RestoreAcronyms 71
\romannumeral 191, 192, 199

S

\s@gls@hyp@opt 46
\s@glsxtr@enabletagging 109
\seename 17
\setabbreviationstyle 71, 136, 144
\setacronymstyle 70, 71
\SetGenericNewAcronym 71
\setglossarystyle 13, 185, 201
\setkeys 6, 13, 14, 30, 45, 69, 116, 117
\setlength 24, 191, 200
\settowidth 72, 191–199
\setupglossaries 4, 14
\sfcode 112, 190
\space 5, 20, 23, 57–59, 65, 67–69, 71–73, 77, 80, 112, 119, 136, 190, 191, 199
\spacefactor 112, 117, 190
\string 5, 6, 19, 20, 23, 26, 42, 43, 51, 52, 57–59, 65, 67–69, 71–74, 76–81, 100, 102, 103, 105
\strut 185–190
\subglossentry 185–190, 200

T

\tablehead 188, 189
\tabletail 188, 189
\tabularnewline 185–190
\TeX 78
\texorpdfstring 53–55, 176–182
textcase package 165
\textsc 147
\textsmaller 151
\texttt 77–80
\the 70, 108, 118, 135–139, 141, 143, 144, 146, 147, 156–159, 162–164
\theindex 104
\this@dialect 182
\toks@ 108

U

\undef 109
\underline 110
\unskip 18, 185
\usepackage 79, 80

V

\val 5, 7, 8, 11, 12

W

\warn@nomakeglossaries 73
\warn@noprintglossary 73
\write 59, 68, 72, 76, 77

X

\xcapitalisewords 98
\xifinlist 64
xindy **202**
xindy 72
xkeyval package 4
\XKV@checkchoice 27
\XKV@plfalse 27
\XKV@resa 27
\XKV@sttrue 27