glossaries-extra.sty v1.15: documented code

Nicola L.C. Talbot

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

2017-05-10

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 . . . 17
1.3 Modifications to Commands Provided by glossaries . . . 23
1.3.1 Existence Checks . . . 28
1.3.2 Document Definitions . . . 31
1.3.3 Existing Glossary Style Modifications . . . 35
1.3.4 Entry Formatting, Hyperlinks and Indexing . . . 39
1.3.5 Entry Counting . . . 72
1.3.6 Acronym Modifications . . . 85
1.3.7 Indexing and Displaying Glossaries . . . 88
1.4 Integration with glossaries-accsupp . . . 110
1.5 Categories . . . 121
1.6 Abbreviations . . . 143
1.6.1 Abbreviation Styles Setup . . . 160
1.6.2 Predefined Styles (Default Font) . . . 163
1.6.3 Predefined Styles (Small Capitals) . . . 176
1.6.4 Predefined Styles (Fake Small Capitals) . . . 180
1.6.5 Predefined Styles (Emphasized) . . . 184
1.6.6 Predefined Styles (User Parentheses Hook) . . . 190
1.7 Using Entries in Headings . . . 198
1.8 Multi-Lingual Support . . . 215

2 Style Adjustments (glossaries-extra-stylemods.sty) 217

2.1 Package Initialisation . . . 217
2.2 List-Like Styles . . . 218
2.3 Longtable Styles . . . 218
2.4 Long Ragged Styles . . . 219
2.5 Supertabular Styles . . . 221
2.6 Super Ragged Styles . . . 222
2.7 Inline Style . . . 223
2.8 Tree Styles . . . 223

Glossary 235

Change History 236

Index 247

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra}[2017/05/10 v1.15 (NLCT)]
```

Requires xkeyval to define package options.

```
\RequirePackage{xkeyval}
```

Requires etoolbox package.

```
\RequirePackage{etoolbox}
```

Has glossaries already been loaded?

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

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

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

Not already loaded, so pass options to glossaries.

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

Set the defaults.

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

Declare package options.

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

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

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

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

`\glsxtrundeftag` Text to display when an entry doesn't exist.

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

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

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

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

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

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

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

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

`f@forglsentries`

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

`f@forglsentries`

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

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

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

`\@glsxtr@record` Does nothing by default.

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

`lsxtr@recordsee` Does nothing by default.

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

`@@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. The third argument is the key family (`glslink` in most cases, `glossadd` for `\glsadd`).

```
\newcommand*{\@@glsxtr@record}[3]{%
 \begingroup
   \def\@glsnumberformat{glsnumberformat}%
   \def\@glsxtr@thevalue{}%
   \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
   \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{#3}{#1}%
```

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

Save the entry counter.

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

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{\let\@glslocref\relax}{\string\glsxtr@record
     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
     {\@glslocref}}%
  \@glsxtr@counterrecordhook
}
```

r@recordcounter

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

p@recordcounter

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

p@recordcounter

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

lsxtr@recordsee Deal with `\glssee` in record mode.

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

srtglossaryunit

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

tr@setup@record Initialise.

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

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

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

addloclistfield

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

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

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

Add a key to store the group heading.

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

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{%
      \let\@glo@autosee\@glsxtr@org@gloautosee
      \renewcommand*{\@do@seeglossary}{\@glsxtr@org@doseeglossary}%
      \renewcommand*{\@glsxtr@record}[3]{}%
      \let\@@do@wrglossary\glsxtr@@do@wrglossary
      \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
      \let\glsxtrundefaction\@glsxtr@err@undefaction
      \let\glsxtr@warnonexistsordo\@gobble
      \let\@@glsxtr@recordcounter\@glsxtr@noop@recordcounter
      \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
      \undef\glsxtrsetaliasnoindex
    }%
  \or
```

Only record (don't index).

```
    \def\glsxtr@setup@record{%
      \ifdef\@glo@autosee{\let\@glo@autosee\relax}{}%
      \let\@do@seeglossary\@glsxtr@recordsee
      \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
      \renewcommand*{\@glsxtr@autoindexcrossrefs}{}%
      \let\@@glsxtr@recordcounter\@glsxtr@op@recordcounter
      \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
```

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

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

Record and index.

```
    \def\glsxtr@setup@record{%
      \let\@glo@autosee\@glsxtr@org@gloautosee
      \renewcommand*{\@do@seeglossary}{\@glsxtr@org@doseeglossary}%
      \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
      \let\@@glsxtr@recordcounter\@glsxtr@op@recordcounter
      \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
      \undef\glsxtrsetaliasnoindex
    }%
  \fi
}
```

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

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}%
}
```

postdot Shortcut for nopostdot=false

```
\@glsxtr@declareoption{postdot}{%
  \glsxtr@dooption{nopostdot=false}%
}
```

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 portability is an issue. (For example, when defining entries in a file that may be input by multiple documents.)

@setupshortcuts Command used to set the shortcuts option.

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

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

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

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]{%
   \let\@glsxtr@shortcutsval\val
   \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
}
```

`lsxtr@doaccsupp`

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

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

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

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

```
\newcommand{\glsxtrNoGlossaryWarning}[1]{%
  \@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
```

g@doseeglossary

Save original definition of `\@do@seeglossary`

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

@org@gloautosee

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

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

Define abbreviations glossaries if required.

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

Setup shortcuts if required.

```
\@glsxtr@setupshortcuts
```

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

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

ariesextrasetup

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

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

Now define the user command:

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

@@do@wrglossary

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

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

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

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

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

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

vide@storagekey Unstarred version.

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

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

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

Provide the no-link command if not already defined.

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

vide@storagekey Starred version.

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

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

`\GlsXtrFmtField`

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

`tDefaultOptions`

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

`\glsxtrfmt` The post-link hook isn't done.

```
\newrobustcmd*{\glsxtrfmt}[3][]{%
  \glsdoifexistsordo{#2}%
  {%
    \ifglshasfield{\GlsXtrFmtField}{#2}%
    {%
      \let\do@gls@link@checkfirsthyper\relax
      \expandafter\@gls@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
        {\csuse{\glscurrentfieldvalue}{#3}}%
    }%
    {#3}%
  }%
  {#3}%
}
```

`\glsxtrentryfmt` No link or indexing.

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

`@glsxtrentryfmt`

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

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

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

trfieldlistgadd Similarly but uses `\listcsgadd`.

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

trfieldlisteadd Similarly but uses `\listcseadd`.

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

trfieldlistxadd Similarly but uses `\listcsxadd`.

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

Now provide commands to iterate over these lists.

fielddolistloop

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

ieldforlistloop

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

List element tests:

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

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

rfieldxifinlist Expands item.

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

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

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

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

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

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

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

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

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

etfieldifexists

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

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

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

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

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

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

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

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

```
\newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
```

```
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csletcs{glo@\glsdetoklabel{#1}@#2}{glo@\glsdetoklabel{#3}@#4}}%
}
```

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

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

xGlsXtrSetField

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

eGlsXtrSetField

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

\glsxtrpageref Like \glsrefentry but references the page number instead (if entry counting is on).

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

lossarypreamble

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

lossarypreamble

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

1.3 Modifications to Commands Provided by glossaries

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

\glsxtralias Provide a key to allow aliases to be defined. The key should be set to the label of the synonymous entry.

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

ryentryposthook Append to the hook to check for the alias key.

```
\appto\@newglossaryentryposthook{%
 \ifcsvoid{glo@\@glo@label @alias}{}%
 {%
```

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

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

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

ewglossaryentry

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

ewglossaryentry Starred version.

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

ewglossaryentry Unstarred version.

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

The following is different from the base glossaries.sty:

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

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

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

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

ignoredglossary Redefine to check for star.

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

ignoredglossary The original definition is patched to check for existence.

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

ignoredglossary Starred form.

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

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

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

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

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

ignoredglossary Unstarred version.

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

ignoredglossary Starred form.

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

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

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

1.3.1 Existence Checks

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

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

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

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

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

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

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

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

arynoexistsordo Similarly for \doifglossarynoexistsordo.

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

```

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:

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

abledflycommand

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

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

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

1.3.3 Existing Glossary Style Modifications

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

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

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

Modify `\setglossarystyle` to set 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@savepreloctag}[2]{%
    \csgdef{@glsxtr@preloctag@##1}{##2}%
  }%
  \renewcommand*{\@glsxtr@doloctag}{%
    \ifcsundef{@glsxtr@preloctag@\glscurrententrylabel}%
    {%
      \GlossariesWarning{Missing pre-location tag for `\glscurrententrylabel'.
        Rerun required}%
    }%
    {%
      \csuse{@glsxtr@preloctag@\glscurrententrylabel}%
    }%
  }%
}
\@onlypreamble\GlsXtrEnablePreLocationTag
```

glsxtrpreloctag

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

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

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

glsxtrpreloctag

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

@glsxtr@pagetag

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

glsxtr@pagestag

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

lsxtrpostloctag

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

lsxtrpostloctag

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

lsxtr@preloctag

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

glsxtr@doloctag

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

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

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

1.3.4 Entry Formatting, Hyperlinks and Indexing

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

```
\renewcommand*{\glsentryfmt}{%
  \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}{}%
  \glsifregular{\glslabel}%
  {\glsxtrregularfont{\glsgenentryfmt}}%
  {%
    \ifglshasshort{\glslabel}%
    {\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}{glslink}%
  \glsdoifexists{#3}%
  {%
```

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

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

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

\@gls@ Save the original definition and redefine.

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

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

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

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

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

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

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

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

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

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

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

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

`\@glsdisp` Save the original definition and redefine. Can't save and restore `\@glsdisp` since it has an optional argument.

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

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

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

sxtrinitwrgloss Set the default if the wrgloss is omitted.

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

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

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

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

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

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

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

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

```
  \glsxtrinitwrgloss
```

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

```
  \@gls@setdefault@glslink@opts
  \do@glsdisablehyperinlist
  \do@gls@link@checkfirsthyper
  \setkeys{glslink}{#1}%
  \glslinkpostsetkeys
  \@gls@saveentrycounter
  \@gls@setsort{\glslabel}%
```

Do write if it should occur before the link text:

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

Do the link text:

```
  \ifKV@glslink@hyper
    \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
  \else
    \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
  \fi
```

Do write if it should occur after the link text:

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

As the original definition:

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

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

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

\glsadd Redefine to include \@glsxtr@record

```
\renewrobustcmd*{\glsadd}[2][]{%
  \@gls@adjustmode
  \@glsxtr@record{#1}{#2}{glossadd}%
  \glsdoifexists{#2}%
  {%
    \def\@glsnumberformat{glsnumberformat}%
    \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
    \def\@glsxtr@thevalue{}%
    \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
    \setkeys{glossadd}{#1}%
    \ifdefempty{\@glsxtr@thevalue}%
    {%
      \@gls@saveentrycounter
    }%
    {%
      \let\theglsentrycounter\@glsxtr@thevalue
      \def\theHglsentrycounter{\@glsxtr@theHvalue}%
    }%
    \@@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
    \@glsxtrsetaliasnoindex
  }
}
{
```

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

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

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

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

setaliasnoindex

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

xtrindexaliased

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

xtrindexaliased

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

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

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

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

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

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

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

\glswriteentry Redefine to test for indexonlyfirst category attribute.

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

```
        {#2}}%
      {#2}%
    \fi
  }%
  {}%
}
```

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

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

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

Similarly for the "noidx" version:

s@noidxglossary

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

xtr@do@@wrindex

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

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

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

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

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

alt@gls@hyp@opt User version

```
\newcommand*{\@alt@gls@hyp@opt}[1][]{%
```

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

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

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

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

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

rSetAltModifier

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

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

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

Check for alias.

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

Redirect link to the alias target.

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

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

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

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. The generated text is scoped.

```
\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@

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

@GLSxtr@p@long@

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

lsxtr@p@longpl@

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

lsxtr@p@longpl@

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

LSxtr@p@longpl@

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

xtr@p@acrshort@

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

xtr@p@acrshort@

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

xtr@p@acrshort@

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

r@p@acrshortpl@

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

r@p@acrshortpl@

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

r@p@acrshortpl@

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

sxtr@p@acrlong@

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

sxtr@p@acrlong@

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

Sxtr@p@acrlong@

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

tr@p@acrlongpl@

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

tr@p@acrlongpl@

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

tr@p@acrlongpl@

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

Commands to minimise conflict.

\@glsxtrp@opt

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

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

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

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

```
\newcommand*{\glossxtrsetpopts}{%
  \glsxtrsetpopts{noindex}%
}
```

\@@glsxtrp

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

Add scope.

```
  {%
    \let\glspostlinkhook\relax
    \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
  }%
}
```

\@glsxtrp

```
\newrobustcmd*{\@glsxtrp}[2]{%
  \ifcsdef{gls#1}%
  {%
    \@@glsxtrp{gls#1}{#2}%
  }%
  {%
    \ifcsdef{glsxtr#1}%
    {%
      \@@glsxtrp{glsxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{`#1' not recognised by
        \string\glsxtrp}{}%
    }%
  }%
}
```

\@Glsxtrp

```
\newrobustcmd*{\@Glsxtrp}[2]{%
  \ifcsdef{Gls#1}%
  {%
    \@@glsxtrp{Gls#1}{#2}%
  }%
  {%
    \ifcsdef{Glsxtr#1}%
    {%
      \@@glsxtrp{Glsxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{`#1' not recognised by
        \string\Glsxtrp}{}%
    }%
  }%
}
```

\@GLSxtrp

```
\newrobustcmd*{\@GLSxtrp}[2]{%
  \ifcsdef{GLS#1}%
  {%
    \@@glsxtrp{GLS#1}{#2}%
  }%
  {%
    \ifcsdef{GLSxtr#1}%
    {%
      \@@glsxtrp{GLSxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{`#1' not recognised by
```

```
        \string\GLSxtrp}{}%
    }%
  }%
}
```

\glsxtr@entry@p

```
\newrobustcmd*{\glsxtr@headentry@p}[2]{%
 \glsifattribute{#1}{headuc}{true}%
 {%
   \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
 }%
 {%
   \@gls@entry@field{#1}{#2}%
 }%
}
```

\glsxtrp Not robust as it needs to expand somewhat.

```
\ifdef\texorpdfstring
{
  \newcommand{\glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{glsxtrhead#1}%
          {%
            {\protect\csuse{glsxtrhead#1}{#2}}%
          }%
          {%
            \glsxtr@headentry@p{#2}{#1}%
          }%
        }%
        {%
          \@glsxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
  }
}
{
  \newcommand{\glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
```

```
        {%
          \ifcsdef{glsxtrhead#1}%
          {%
            {\protect\csuse{glsxtrhead#1}}%
          }%
          {%
            \glsxtr@headentry@p{#2}{#1}%
          }%
        }%
        {%
          \@glsxtrp{#1}{#2}%
        }%
      }%
  }
}
```

Provide short synonyms for the most common option.

\glsps

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

\glspt

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

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

```
\ifdef\texorpdfstring
{
  \newcommand{\Glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{Glsxtrhead#1}%
          {%
            {\protect\csuse{Glsxtrhead#1}{#2}}%
          }%
          {%
            \protect\@Gls@entry@field{#2}{#1}%
          }%
        }%
        {%
          \@Glsxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
  }
}
{
  \newcommand{\Glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
      {%
        \ifcsdef{Glsxtrhead#1}%
        {%
          {\protect\csuse{Glsxtrhead#1}}%
        }%
        {%
          \protect\@Gls@entry@field{#2}{#1}%
        }%
      }%
      {%
        \@Glsxtrp{#1}{#2}%
      }%
    }%
  }
}
```

\GLSxtrp As above but all upper case (but not for the bookmarks, which can't process \uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand{\GLSxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{GLSxtr#1}%
          {%
            {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
          }%
          {%
            \protect\mfirstucMakeUppercase
            {%
              \protect\@gls@entry@field{#2}{#1}%
            }%
          }%
        }%
        {%
          \@GLSxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
  }
}
{
  \newcommand{\GLSxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
      {%
        \ifcsdef{GLSxtr#1}%
        {%
          {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
        }%
        {%
          \protect\mfirstucMakeUppercase
          {%
            \protect\@gls@entry@field{#2}{#1}%
          }%
        }%
      }%
      {%
        \@GLSxtrp{#1}{#2}%
      }%
    }%
  }
}
```

1.3.5 Entry Counting

The 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.

```
\renewcommand*{\@glsunset}[1]{%
  \@@glsunset{#1}%
  \glsxtrpostunset{#1}%
}%
```

`glsxtrpostunset`

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

`\@glslocalunset` Local unset.

```
  \renewcommand*{\@glslocalunset}[1]{%
    \@@glslocalunset{#1}%
    \glsxtrpostlocalunset{#1}%
  }%
```

rpostlocalunset

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

\@glsreset Global reset.

```
\renewcommand*{\@glsreset}[1]{%
  \@@glsreset{#1}%
  \glsxtrpostreset{#1}%
}%
```

glsxtrpostreset

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

\@glslocalreset Local reset.

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

rpostlocalreset

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

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 `\cgls` etc so that it can be used interchangeable with `\gls` etc.

`\@cgls@`

```
\def\@cgls@#1#2[#3]{\@gls@{#1}{#2}[#3]}
```

`\@cGls@`

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

`\@cglspl@`

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

`\@cGlspl@`

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

Add all upper case versions not provided by glossaries.

`\cGLS`

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

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

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

`\@cGLS@`

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

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

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

`\cGLSpl`

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

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

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

\@cGLSpl@

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

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

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

Modify the trigger formats to check for the regular attribute.

\cglsformat

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

\cGlsformat

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

\cglsplformat

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

\cGlsplformat

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

New code similar to above for unit counting.

defunitcounters

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

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

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

@addunitcounter

```
\newcommand*{\@glsxtr@addunitcounter}[1]{%
 \listadd{\@glsxtr@unitcountlist}{#1}%
 \ifcsundef{glsxtr@theunit@#1}
 {%
   \ifcsdef{theH#1}%
   {\csdef{glsxtr@theunit@#1}{\csuse{theH#1}}}%
   {\csdef{glsxtr@theunit@#1}{\csuse{the#1}}}%
 }%
 {}%
}
```

r@ifunitcounter

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

urrentunitcount

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

eviousunitcount

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

t@currunitcount

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

t@currunitcount

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

r@currunitcount

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

r@prevunitcount

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

eentryunitcount

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

Enable new fields:

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

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

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

New commands to access new fields:

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

Access total count:

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

Access max value:

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

Adjust post unset and reset:

```
  \let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
  \renewcommand*{\glsxtrpostunset}[1]{%
    \@glsxtr@entryunitcount@org@unset{##1}%
    \@gls@increment@currunitcount{##1}%
  }%
  \let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
```

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

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

```
\let\@cgls@\@@cgls@
\let\@cglspl@\@@cglspl@
```

```
\let\@cGls@\@@cGls@
\let\@cGlspl@\@@cGlspl@
\let\@cGLS@\@@cGLS@
\let\@cGLSpl@\@@cGLSpl@
```

Write information to the aux file.

```
\AtEndDocument{\@gls@write@entryunitcounts}%
\renewcommand*{\@gls@entry@unitcount}[3]{%
  \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
  \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
  {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
  {%
    \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
      \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
  }%
  \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
  {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
```

```
    {%
      \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
      \fi
    }%
  }%
  \let\glsenableentryunitcount\relax
  \renewcommand*{\glsenableentrycount}{%
    \PackageError{glossaries-extra}{\string\glsenableentrycount\space
     can't be used with \string\glsenableentryunitcount}%
    {Use one or other but not both commands}%
  }%
}
\@onlypreamble\glsenableentryunitcount
```

entry@unitcount

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

ryunitcounts@do

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

entryunitcounts

```
\newcommand*{\@gls@write@entryunitcounts}{%
  \immediate\write\@auxout
    {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
  \count@=0\relax
  \forallglsentries{\@glsentry}{%
    \glshasattribute{\@glsentry}{unitcount}%
    {%
      \ifglsused{\@glsentry}%
      {%
        \forlistcsloop
          {\@gls@write@entryunitcounts@do}%
          {glo@\glsdetoklabel{\@glsentry}@unitlist}%
      }%
      {}%
      \advance\count@ by \@ne
    }%
    {}%
  }%
  \ifnum\count@=0
    \GlossariesExtraWarningNoLine{Entry counting has been enabled
```

```
      \MessageBreak with \string\glsenableentryunitcount\space but the
      \MessageBreak attribute ‘unitcount’ hasn’t
      \MessageBreak been assigned to any of the defined
      \MessageBreak entries}%
   \fi
}
```

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

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

Enable entry counting:

```
   \glsenableentryunitcount
```

Redefine `\gls` etc:

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

Set the entrycount attribute:

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

In case this command is used again:

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

tcountunsetattr

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

1.3.6 Acronym Modifications

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

nericNewAcronym

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

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

First save the original definitions:

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

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

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

Switch acronyms to abbreviations:

```
\MakeAcronymsAbbreviations
```

RestoreAcronyms Restore acronyms to glossaries interface.

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

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

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

`\glsacspace` Allow the user to customise the maximum value.

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

`\glsacspacemax` Value used in the above.

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

1.3.7 Indexing and Displaying Glossaries

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

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

`r@reg@glosslist`

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

Save the original definition of `\makeglossaries`:

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

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

`\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}}
   \protected@write\@auxout{}{\string\glsxtr@makeglossaries{#1}}
   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
```

Iterate through each supplied glossary type and activate it.

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

New glossaries must be created before \makeglossaries:

```
   \renewcommand*\newglossary[4][]{%
   \PackageError{glossaries}{New glossaries
   must be created before \string\makeglossaries}{You need
   to move \string\makeglossaries\space after all your
   \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
  \let\@makeglossary\relax
  \let\makeglossary\relax
  \renewcommand\makeglossaries[1][]{}%
```

Disable all commands that have no effect after \makeglossaries

```
  \@disable@onlypremakeg
```

Allow see key:

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

Adjust \@do@seeglossary

```
  \renewcommand*{\@do@seeglossary}[2]{%
    \edef\@gls@label{\glsdetoklabel{##1}}%
    \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
    \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
    {\@glsxtr@org@doseeglossary{##1}{##2}}%
    {%
      \protected@write\@auxout{}{%
        \string\@gls@reference
          {\gls@type}{\@gls@label}{\string\glsseeformat##2{}}%
      }%
    }%
  }%
```

Adjust \@@do@@wrglossary

```
  \let\@glsxtr@@do@@wrglossary\@@do@@wrglossary
  \def\@@do@@wrglossary{%
    \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
    \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
    {\@glsxtr@@do@@wrglossary}%
    {\gls@noidxglossary}%
  }%
```

Suppress warning about no \makeglossaries

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

Only warn for glossaries not listed.

```
\renewcommand{\@gls@noref@warn}[1]{%
  \edef\@gls@type{##1}%
  \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
  {%
    \GlossariesExtraWarning{Can’t use
       \string\printnoidxglossary[type={\@gls@type}]
       when ‘\@gls@type’ is listed in the optional argument of
       \string\makeglossaries}%
  }%
  {%
    \GlossariesWarning{Empty glossary for
    \string\printnoidxglossary[type={##1}].
    Rerun may be required (or you may have forgotten to use
    commands like \string\gls)}%
  }%
}%
```

Adjust display number list to check for type:

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

Adjust entry list:

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

Adjust number list loop

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

Only sanitize sort for normal indexing glossaries.

```
\renewcommand*{\glsprestandardsort}[3]{%
  \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
  {%
    \glsdosanitizesort
  }%
  {%
    \ifglssanitizesort
     \@gls@noidx@sanitizesort
    \else
     \@gls@noidx@nosanitizesort
    \fi
  }%
}%
```

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

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

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

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

Need to extract just the type value.

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

Check automake setting:

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

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

rgprintglossary This no longer simply saves `\@printglossary` with `\let` is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes `\provideignoredglossary` to the glstex file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```
\newcommand{\@glsxtr@orgprintglossary}[2]{%
  \def\@glo@type{\glsdefaulttype}%
```

Add check here.

```
  \def\glossarytitle{%
     \ifcsdef{@glotype@\@glo@type @title}%
     {\csuse{@glotype@\@glo@type @title}}%
     {\glossaryname}}%
  \def\glossarytoctitle{\glossarytitle}%
  \let\org@glossarytitle\glossarytitle
  \def\@glossarystyle{%
    \ifx\@glossary@default@style\relax
      \GlossariesWarning{No default glossary style provided \MessageBreak
        for the glossary '\@glo@type'. \MessageBreak
        Using deprecated fallback. \MessageBreak
        To fix this set the style with \MessageBreak
        \string\setglossarystyle\space or use the \MessageBreak
        style key=value option}%
    \fi
  }%
  \def\gls@dotoctitle{\glssettoctitle{\@glo@type}}%
  \let\@org@glossaryentrynumbers\glossaryentrynumbers
  \bgroup
    \@printgloss@setsort
    \setkeys{printgloss}{#1}%
    \ifx\glossarytitle\org@glossarytitle
    \else
      \cslet{@glotype@\@glo@type @title}{\glossarytitle}%
    \fi
    \let\currentglossary\@glo@type
    \let\org@glossaryentrynumbers\glossaryentrynumbers
    \let\glsnonextpages\@glsnonextpages
    \let\glsnextpages\@glsnextpages
    \let\nopostdesc\@nopostdesc
    \gls@dotoctitle
    \@glossarystyle
    \let\gls@org@glossaryentryfield\glossentry
    \let\gls@org@glossarysubentryfield\subglossentry
    \renewcommand{\glossentry}[1]{%
      \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
      \gls@org@glossaryentryfield{##1}%
    }%
    \renewcommand{\subglossentry}[2]{%
      \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
      \gls@org@glossarysubentryfield{##1}{##2}%
    }%
    \@gls@preglossaryhook
    #2%
  \egroup
  \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
  \global\let\warn@noprintglossary\relax
}
```

\@printglossary Redefine.

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

Add a key that switches off the entry targets:

```
\define@choicekey{printgloss}{target}[\val\nr]{true,false}[true]{%
  \ifcase\nr
    \let\@glstarget\glsdohypertarget
  \else
    \let\@glstarget\@secondoftwo
  \fi
}
```

@makeglossaries For the benefit of makeglossaries

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

@glsxtr@gettype Get just the type.

```
\def\@glsxtr@gettype#1,type=#2,#3\@end@glsxtr@gettype{%
  \def\@glo@type{#2}%
}
```

@assign@sortkey Assign the sort key.

```
\newcommand\@glsxtr@mixed@assign@sortkey[1]{%
  \edef\@glo@type{\@glo@type}%
  \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glsxtr@reg@glosslist}%
  {%
    \@glo@no@assign@sortkey{#1}%
  }%
  {%
    \@@glo@assign@sortkey{#1}%
  }%
}%
```

Display number list for the regular version:

splaynumberlist

```
\let\@glsxtr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
\newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \def\@gls@noidxloclist@sep{%
      \def\@gls@noidxloclist@sep{%
        \def\@gls@noidxloclist@sep{%
          \glsnumlistsep
        }%
        \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
      }%
    }%
    \def\@gls@noidxloclist@finalsep{}%
    \def\@gls@noidxloclist@prev{}%
    \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
    \@gls@noidxloclist@finalsep
    \@gls@noidxloclist@prev
  }%
  {%
    ??\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}}
```

x@getgrouptitle Patch.

```
\renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \ifcsundef{\@glsxtr@titlelabel}%
  {%
    \DTLifint{#1}%
    {%
      \ifnum#1<256\relax
        \edef#2{\char#1\relax}%
      \else
        \edef#2{#1}%
      \fi
    }%
    {%
      \ifcsundef{#1groupname}%
      {\def#2{#1}}%
      {\letcs#2{#1groupname}}%
    }%
  }%
  {%
    \letcs#2{\@glsxtr@titlelabel}%
  }%
}
```

`g@getgrouptitle` Save original definition of `\@gls@getgrouptitle`

```
\let\glsxtr@org@getgrouptitle\@gls@getgrouptitle
```

`trgetgrouptitle` Provide a user-level command to fetch the group title. The first argument is the group label. The second argument is a control sequence in which to store the title.

```
\newrobustcmd{\glsxtrgetgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \ifcsdef{\@glsxtr@titlelabel}
  {\letcs{#2}{\@glsxtr@titlelabel}}%
  {\glsxtr@org@getgrouptitle{#1}{#2}}%
}
\let\@gls@getgrouptitle\glsxtrgetgrouptitle
```

`trsetgrouptitle` Sets the title for the given group label.

```
\newcommand{\glsxtrsetgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \csxdef{\@glsxtr@titlelabel}{#2}%
}
```

`\glsnavigation` Redefine to use new user-level command.

```
\renewcommand*{\glsnavigation}{%
  \def\@gls@between{}%
  \ifcsundef{@gls@hypergrouplist@\@glo@type}%
  {%
    \def\@gls@list{}%
  }%
  {%
    \expandafter\let\expandafter\@gls@list
      \csname @gls@hypergrouplist@\@glo@type\endcsname
  }%
  \@for\@gls@tmp:=\@gls@list\do{%
    \@gls@between
    \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
    \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
    \let\@gls@between\glshypernavsep
  }%
}
```

`@noidx@glossary`

```
\renewcommand*{\@print@noidx@glossary}{%
  \ifcsdef{@glsref@\@glo@type}%
  {%
    \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
    {%
      \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
    }%
    {%
      \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
    }%
    \glossarysection[\glossarytoctitle]{\glossarytitle}%
    \glossarypreamble
```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```
    \def\@gls@currentlettergroup{}%
    \begin{theglossary}%
    \glossaryheader
    \glsresetentrylist
    \forlistcsloop{\@gls@noidx@do}{@glsref@\@glo@type}%
    \end{theglossary}%
    \glossarypostamble
  }%
  {%
```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```
    \glsxtrifemptyglossary{\@glo@type}%
    {}%
    {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
    \@gls@noref@warn{\@glo@type}%
  }%
}
```

noidxdisplayloc Patch to check for range formations.

```
\renewcommand*{\glsnoidxdisplayloc}[4]{%
  \setentrycounter[#1]{#2}%
  \@glsxtr@display@loc#3\empty\end@glsxtr@display@loc{#4}%
}
```

xtr@display@loc Patch to check for range formations.

```
\def\@glsxtr@display@loc#1#2\end@glsxtr@display@loc#3{%
  \ifx#1(\relax
    \glsxtrdisplaystartloc{#2}{#3}%
  \else
    \ifx#1)\relax
      \glsxtrdisplayendloc{#2}{#3}%
    \else
      \glsxtrdisplaysingleloc{#1#2}{#3}%
    \fi
  \fi
}
```

isplaysingleloc Single location.

```
\newcommand*{\glsxtrdisplaysingleloc}[2]{%
  \csuse{#1}{#2}%
}
```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glsxtrlocrangefmt`.

displaystartloc Start of a location range.

```
\newcommand*{\glsxtrdisplaystartloc}[2]{%
  \edef\glsxtrlocrangefmt{#1}%
  \ifx\glsxtrlocrangefmt\empty
    \def\glsxtrlocrangefmt{glsnumberformat}%
  \fi
  \expandafter\glsxtrdisplaysingleloc
    \expandafter{\glsxtrlocrangefmt}{#2}%
}
```

trdisplayendloc End of a location range.

```
\newcommand*{\glsxtrdisplayendloc}[2]{%
  \edef\@glsxtr@tmp{#1}%
  \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}%
  \ifx\glsxtrlocrangefmt\@glsxtr@tmp
  \else
    \GlossariesExtraWarning{Mismatched end location range
      (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
  \fi
  \expandafter\glsxtrdisplayendlochook\expandafter{\@glsxtr@tmp}{#2}%
  \expandafter\glsxtrdisplaysingleloc
    \expandafter{\glsxtrlocrangefmt}{#2}%
  \def\glsxtrlocrangefmt{}%
}
```

splayendlochook Allow the user to hook into the end of range command.

```
\newcommand*{\glsxtrdisplayendlochook}[2]{}
```

sxtrlocrangefmt Current range format. Empty if not in a range.

```
\newcommand*{\glsxtrlocrangefmt}{}
```

ls@removespaces Redefine to allow adjustments to location hyperlink.

```
\def\@gls@removespaces#1 #2\@nil{%
 \toks@=\expandafter{\the\toks@#1}%
 \ifx\\#2\\%
   \edef\x{\the\toks@}%
   \ifx\x\empty
   \else
    \glsxtrlocationhyperlink{\glsentrycounter}{\@glo@counterprefix}{\the\toks@}%
   \fi
 \else
   \@gls@ReturnAfterFi{%
     \@gls@removespaces#2\@nil
   }%
 \fi
}
```

cationhyperlink

```
\newcommand*{\glsxtrlocationhyperlink}[3]{%
  \ifdefvoid\glsxtrsupplocationurl
  {%
    \hyperlink{#1#2#3}{#3}%
  }%
  {%
    \hyperref{\glsxtrsupplocationurl}{}{#1#2#3}{#3}%
  }%
}
```

supphypernumber

```
\newcommand*{\glsxtrsupphypernumber}[1]{%
 {%
   \glshasattribute{\glscurrententrylabel}{externallocation}%
   {%
     \def\glsxtrsupplocationurl{%
       \glsgetattribute{\glscurrententrylabel}{externallocation}}%
   }%
   {%
     \def\glsxtrsupplocationurl{}%
   }%
   \glshypernumber{#1}%
 }%
}
```

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

@print@glossary

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

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

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

`oGlsWarningHead` Header message.

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

`rningEmptyStart` No entries have been added to the glossary.

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

`arningEmptyMain` The default “main” glossary is empty.

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

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

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

`arningCheckFile` Advisory message to check the file contents.

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

`WarningAutoMake` Message when automake option has been used.

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

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

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

   \item Run the external (Perl) application:

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

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

`WarningMisMatch` Mismatching `\makenoidxglossaries`.

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

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

arningBuildInfo Build advice.

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

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

    \item Run the external (Lua) application:

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

    \item Run the external (Perl) application:

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

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

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 for use with bib2gls.

xtrresourcefile Since it's dangerous for an external application to create a file with a .tex extension, as from v1.11 this enforces a .glstex extension to avoid conflict.

```
\newcommand*{\glsxtrresourcefile}[2][]{%
  \protected@write\@auxout{}{\string\glsxtr@resource{#1}{#2}}%
  \glsxtr@writefields
  \let\@glsxtr@org@see@noindex\@gls@see@noindex
  \let\@gls@see@noindex\relax
  \IfFileExists{#2.glstex}%
  {%
```

Can't scope `\@input` so save and restore the category code of `@` to allow for internal commands in the location list.

```
    \edef\@bibgls@restoreat{\noexpand\catcode\noexpand`\noexpand\@=\number\catcode`\@}%
    \makeatletter
    \@input{#2.glstex}%
    \@bibgls@restoreat
  }%
  {%
    \GlossariesExtraWarning{No file `#2.glstex'}%
  }%
  \let\@gls@see@noindex\@glsxtr@org@see@noindex
}
\@onlypreamble\glsxtrresourcefile
```

trresourcecount

```
\newcount\glsxtrresourcecount
```

trLoadResources Short cut that uses `\glsxtrresourcefile` with `\jobname` as the mandatory argument.

```
\newcommand*{\GlsXtrLoadResources}[1][]{%
  \ifnum\glsxtrresourcecount=0\relax
    \glsxtrresourcefile[#1]{\jobname}%
  \else
    \glsxtrresourcefile[#1]{\jobname-\the\glsxtrresourcecount}%
  \fi
  \advance\glsxtrresourcecount by 1\relax
}
```

glsxtr@resource

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

\glsxtr@fields

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

xtr@texencoding

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

\glsxtr@langtag

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

@pluralsuffixes

```
\newcommand*{\glsxtr@pluralsuffixes}[4]{}
```

tr@shortcutsval

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

sxtr@linkprefix

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

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
\newcommand*{\glsxtr@writefields}{%
  \protected@write\@auxout{}{\string\glsxtr@fields{\@gls@keymap}}%
```

If any languages have been loaded, the language tag will be available in `\CurrentTrackedLanguageTag` (provided by tracklang). For multilingual documents, the required locale will have to be indicated in the `sort` key when using `\glsxtrresourcefile`.

```
  \ifdef\CurrentTrackedLanguageTag
  {%
     \protected@write\@auxout{}{%
       \string\glsxtr@langtag{\CurrentTrackedLanguageTag}}%
  }%
  {}%
  \protected@write\@auxout{}{\string\glsxtr@pluralsuffixes
    {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
    {\glsxtrabbrvpluralsuffix}}%
  \ifdef\inputencodingname
  {%
     \protected@write\@auxout{}{\string\glsxtr@texencoding{\inputencodingname}}%
  }%
  {%
```

If fontspec has been loaded, assume UTF-8. (The encoding can be changed with `\XeTeXinputencoding`, but I can't work out how to determine the current encoding.)

```
     \@ifpackageloaded{fontspec}%
     {\protected@write\@auxout{}{\string\glsxtr@texencoding{utf8}}}%
     {}%
  }%
  \protected@write\@auxout{}{\string\glsxtr@shortcutsval{\@glsxtr@shortcutsval}}%
```

Prefix deferred until the beginning of the document in case it's redefined later in the preamble. This is picked up by bib2gls when the external option is used.

```
  \AtBeginDocument
    {\protected@write\@auxout{}{\string\glsxtr@linkprefix{\glolinkprefix}}}%
  \let\glsxtr@writefields\relax
```

If the automake option is on, try running bib2gls if the aux file exists.

```
  \ifglsautomake
    \IfFileExists{\jobname.aux}%
    {\immediate\write18{bib2gls "\jobname"}}{}%
```

If \makeglossaries is also used, allow makeindex/xindy to also be run, otherwise disable the error message about requiring \makeglossaries with automake=true.

```
    \ifx\@gls@doautomake\@gls@doautomake@err
      \let\@gls@doautomake\relax
    \fi
  \fi
}
```

do@automake@err

```
\newcommand*{\@gls@doautomake@err}{%
  \PackageError{glossaries}{You must use
  \string\makeglossaries\space with automake=true}
  {%
     Either remove the automake=true setting or
     add \string\makeglossaries\space to your document preamble.%
  }%
}
```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

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

r@counterrecord Aux file command.

```
\newcommand*{\glsxtr@counterrecord}[3]{%
  \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
}
```

unterrecordhook Hook used by \@glsxtr@dorecord.

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

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
\newcommand*{\GlsXtrRecordCounter}[1]{%
  \@@glsxtr@recordcounter{#1}%
}
\@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
\newcommand*{\@glsxtr@docounterrecord}[1]{%
  \protected@write\@auxout{}{\string\glsxtr@counterrecord
    {\@gls@label}{#1}{\csuse{the#1}}}%
}
```

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

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

ntunsrtglossary Unstarred version.

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

ntunsrtglossary Starred version.

```
\newcommand*{\s@printunsrtglossary}[2][]{%
  \begingroup
    #2%
    \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
  \endgroup
}
```

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

```
\newcommand*{\printunsrtglossaries}{%
  \forallglossaries{\@@glo@type}{\printunsrtglossary[type=\@@glo@type]}%
}
```

@unsrt@glossary

```
\newcommand*{\@print@unsrt@glossary}{%
  \glossarysection[\glossarytoctitle]{\glossarytitle}%
  \glossarypreamble
```

check for empty list

```
  \glsxtrifemptyglossary{\@glo@type}%
  {%
    \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
  }%
  {%
    \key@ifundefined{glossentry}{group}%
    {\let\@gls@getgrouptitle\@glsxtr@noidx@getgrouptitle}%
    {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
    \begin{theglossary}%
    \glossaryheader
    \glsresetentrylist
    \def\@gls@currentlettergroup{}%
    \expandafter\@for\expandafter\glscurrententrylabel\expandafter
      :\expandafter=\csname glolist@\@glo@type\endcsname\do{%
      \ifdefempty{\glscurrententrylabel}
      {}%
      {\printunsrtglossaryhandler\glscurrententrylabel}%
    }%
    \end{theglossary}%
  }%
  \glossarypostamble
}
```

glossaryhandler

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

srtglossaryunit

```
\newcommand{\print@op@unsrtglossaryunit}[2][]{%
  \s@printunsrtglossary[type=\glsdefaulttype,#1]{%
    \printunsrtglossaryunitsetup{#2}%
  }%
}
```

ossaryunitsetup

```
\newcommand*{\printunsrtglossaryunitsetup}[1]{%
  \renewcommand{\printunsrtglossaryhandler}[1]{%
    \glsxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
    {\glsxtrunsrtdo{##1}}%
    {}%
  }%
  \ifcsundef{theH#1}%
  {%
    \renewcommand*{\glolinkprefix}{record.#1.\csuse{the#1}.}%
  }%
  {%
    \renewcommand*{\glolinkprefix}{record.#1.\csuse{theH#1}.}%
  }%
  \renewcommand*{\glossarysection}[2][]{}%
  \appto\glossarypostamble{\glspar\medskip\glspar}%
}
```

srtglossaryunit

```
\newcommand{\print@noop@unsrtglossaryunit}[2][]{%
  \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
   requires the record=only or record=alsoindex package option}{}%
}
```

t@getgrouptitle

```
\newrobustcmd*{\@glsxtr@unsrt@getgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \ifcsdef{\@glsxtr@titlelabel}
  {\letcs{#2}{\@glsxtr@titlelabel}}%
  {\def#2{#1}}%
}
```

\glsxtrunsrtdo Provide a user-level call to \@glsxtr@noidx@do to make it easier to define a new handler.

```
\newcommand{\glsxtrunsrtdo}{\@glsxtr@noidx@do}
```

glsxtr@noidx@do Minor modification of \@gls@noidx@do to check for location field if present.

```
\newcommand{\@glsxtr@noidx@do}[1]{%
  \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@location}%
  \ifglshasparent{#1}%
  {%
    \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
    \ifdefvoid{\@gls@location}%
    {%
      \ifdefvoid{\@gls@loclist}%
      {%
        \subglossentry{\gls@level}{#1}{}%
      }%
      {%
        \subglossentry{\gls@level}{#1}%
        {%
          \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
        }%
      }%
    }%
    {%
      \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
    }%
  }%
  {%
    \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
    \key@ifundefined{glossentry}{group}%
    {%
      \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
    }%
    {%
      \protected@xdef\@glo@thislettergrp{%
        \csname glo@\glsdetoklabel{#1}@group\endcsname}%
    }%
    \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
    {}%
    {%
      \ifdefempty{\@gls@currentlettergroup}{}{\glsgroupskip}%
      \expandafter\glsgroupheading\expandafter
        {\csname glo@\glsdetoklabel{#1}@group\endcsname}%
    }%
    \let\@gls@currentlettergroup\@glo@thislettergrp
    \ifdefvoid{\@gls@location}%
    {%
      \ifdefvoid{\@gls@loclist}
      {%
        \glossentry{#1}{}%
      }%
      {%
        \glossentry{#1}%
        {%
          \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
        }%
      }%
    }%
    {%
      \glossentry{#1}%
      {%
        \glossaryentrynumbers{\@gls@location}%
      }%
    }%
  }%
}
```

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
```

Since the glossaries are automatically scoped, \@glsxtr@org@postdescription shouldn't already be defined, but check anyway just as a precautionary measure.

```
    \ifundef\@glsxtr@org@postdescription
    {%
      \let\@glsxtr@org@postdescription\glspostdescription
      \renewcommand*{\glspostdescription}{%
        \ifglsentryexists{\glscurrententrylabel}%
        {%
          \glsxtrpostdescription
          \@glsxtr@org@postdescription
        }%
        {}%
      }%
    }%
    {}%
```

Enable the options used by \@@glsxtrp:

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

postdescription This command will only be used if \@gls@preglossaryhook is available *and* the glossary style uses \glspostdescription without modifying it. (\nopostdesc will suppress this.) The glossaries-extra-stylemods package will add the post description hook to all the predefined styles that don't include it.

```
\newcommand*{\glsxtrpostdescription}{%
  \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
}
```

postdescgeneral

```
\newcommand*{\glsxtrpostdescgeneral}{}
```

xtrpostdescterm

```
\newcommand*{\glsxtrpostdescterm}{}
```

postdescacronym

```
\newcommand*{\glsxtrpostdescacronym}{}
```

escabbreviation

```
\newcommand*{\glsxtrpostdescabbreviation}{}
```

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

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

glsxtrdopostpunc

```
\glsxtrdopostpunc{⟨code⟩}
```

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

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

@glsxtr@swaptwo

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

1.6 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.

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

Warning should already have been issued.

```
   \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
   \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
   \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
 }%
 {}%
}
```

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

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

Similarly for the long plural form.

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

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```
\newtoks\glsshortpltok
```

\glslongpltok

```
\newtoks\glslongpltok
```

sxtr@insertdots Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to `\newabbreviation`. Note that explicitly using the short or shortplural keys will override this.

```
\newcommand*{\@glsxtr@insertdots}[2]{%
  \def#1{}%
  \@glsxtr@insert@dots#1#2\@nnil
}
```

xtr@insert@dots

```
\newcommand*{\@glsxtr@insert@dots}[2]{%
  \ifx\@nnil#2\relax
   \let\@glsxtr@insert@dots@next\@gobble
  \else
   \ifx\relax#2\relax
```

```
    \else
      \appto#1{#2.}%
    \fi
    \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
  \fi
  \@glsxtr@insert@dots@next#1%
}
```

newabbreviation Define a new generic abbreviation.

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

newabbreviation Internal macro. (bib2gls has an option that needs to temporarily redefine `\newabbreviation`. This is just makes it easier to save and restore the original definition.)

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

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

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

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

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

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

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

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

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

longdefaultfont

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

brvpluralsuffix Default plural suffix. Allow an alternative default suffix for abbreviations.

```
\newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}
```

brvpluralsuffix Default plural suffix.

```
\newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtrfull Full form (no case-change).

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

\@glsxtr@full Low-level macro:

```
\def\@glsxtr@full#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\glsxtrinlinefullformat{#2}{#3}}%
```

What should `\glsxtrifwasfirstuse` be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

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

trsetupfulldefs

```
\newcommand*{\glsxtrsetupfulldefs}{%
  \let\glsxtrifwasfirstuse\@firstoftwo
}
```

\Glsxtrfull Full form (first letter uppercase).

```
\newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
\newcommand*\ns@Glsxtrfull[2][]{%
  \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
                  {\@Glsxtr@full{#1}{#2}[]}%
}
```

\@Glsxtr@full Low-level macro:

```
\def\@Glsxtr@full#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrfull Full form (all uppercase).

```
\newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
\newcommand*\ns@GLSxtrfull[2][]{%
  \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}%
                  {\@GLSxtr@full{#1}{#2}[]}%
}
```

\@GLSxtr@full Low-level macro:

```
\def\@GLSxtr@full#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\glsxtrfullpl Plural full form (no case-change).

```
\newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
\newcommand*\ns@glsxtrfullpl[2][]{%
  \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}%
                  {\@glsxtr@fullpl{#1}{#2}[]}%
}
```

\@glsxtr@fullpl Low-level macro:

```
\def\@glsxtr@fullpl#1#2[#3]{%
  \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\glsabbrvfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

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

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

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    (\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\glsabbrvfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

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

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

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    (\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\glsabbrvfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

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

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

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
```

```
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
  \protect\glsxtrabbrvfootnote{##1}%
    {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
  \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
  \protect\glsxtrabbrvfootnote{##1}%
    {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
}%
\renewcommand*{\Glsxtrfullplformat}[2]{%
  \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
  \protect\glsxtrabbrvfootnote{##1}%
    {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
}%
```

The first use full form and the inline full form use the short (long) style.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  (\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\glsabbrvfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
```

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

```
        \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```
  \renewcommand*{\glsxtrsetupfulldefs}{%
    \let\glsxtrifwasfirstuse\@secondoftwo
  }%
}%
{%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\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}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\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}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    (\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}{\glsxtrabbrvpluralsuffix}%
\renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
\renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
\renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
\renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the 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

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

and for the default short form suffix:

\glsxtrscsuffix

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

long-short-sc

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

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

Mostly as long-short style:

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

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}}%
}
```

g-short-sc-desc

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

Mostly as long-short-desc style:

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

Now the short (long) version

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

Mostly as short-long style:

```
  \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

```
\newcommand*{\glsxtrsmsuffix}{\glsxtrabbrvpluralsuffix}
```

long-short-sm

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

Mostly as long-short style:

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

g-short-sm-desc

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

Mostly as long-short-desc style:

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

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

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

Mostly as short-long style:

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

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

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

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\glsabbrvfont{\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\glsabbrvfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

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

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

```
  \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}{\glsxtrabbrvpluralsuffix}
```

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

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

```
\newabbreviationstyle{long-postshort-user}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
    description={\protect\glslonguserfont{\the\glslongtok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

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

In-line format:

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

short-user-desc Like long-postshort-user but the user supplies the description.

```
\newabbreviationstyle{long-postshort-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsfirstlongfont{\the\glslongtok}%
          \protect\glsxtruserparen
           {\protect\glsabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},
    sort={\the\glslongtok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
}
```

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

```
\newabbreviationstyle{short-postlong-user}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
    description={\protect\glslonguserfont{\the\glslongtok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvuserfont{\glsentrylong{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

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

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

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

In-line format:

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
  }%
}
```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```
\newabbreviationstyle{short-postlong-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}%
           \protect\glsxtruserparen
             {\protect\glsfirstlongfont{\the\glslongpltok}}%
             {\the\glslabeltok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%

    plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvuserfont{\glsentrylong{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
}
```

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\glsabbrvfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

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

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

```
\renewcommand*{\abbrvpluralsuffix}{\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}[2017/05/10 v1.15 (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
}
```

uteTreeSubIndent

\glsxtrComputeTreeSubIndent{⟨*level*⟩}{⟨*label*⟩}{⟨*register*⟩}

Compute the indent for the sub-entries. The first argument is the level, the second argument is the entry label and the third argument is the length register used to store the computed indent.

```
\newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
  \ifcsundef{@glswidestname\romannumeral#1}%
  {%
    \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
  }%
  {%
    \settowidth{#3}{\glstreenamefmt{%
           \csname @glswidestname\romannumeral#1\endcsname\space}}%
  }%
}
```

eeSetHangIndent Set \hangindent for top-level entries:

```
\newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
\newcommand*{\glsxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
\renewglossarystyle{alttree}{%
  \renewenvironment{theglossary}%
    {%
     \glsxtralttreeInit
```

```
   \def\@gls@prevlevel{-1}%
   \mbox{}\par}%
  {\par}%
\renewcommand*{\glossaryheader}{}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand{\glossentry}[2]{%
  \ifnum\@gls@prevlevel=0\relax
  \else
    \glsxtrComputeTreeIndent{##1}%
  \fi
  \parindent\glstreeindent
  \glsxtrAltTreeSetHangIndent
  \makebox[0pt][r]%
  {%
    \glstreenamebox{\glstreeindent}%
    {%
       \glsentryitem{##1}%
       \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
    }%
  }%
  \glsxtralttreeSymbolDescLocation{##1}{##2}%
  \def\@gls@prevlevel{0}%
}
\renewcommand{\subglossentry}[3]{%
  \ifnum##1=1\relax
    \glssubentryitem{##2}%
  \fi
  \ifnum\@gls@prevlevel=##1\relax
  \else
    \glsxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
    \ifnum\@gls@prevlevel<##1\relax
      \setlength\glstreeindent\gls@tmplen
      \addtolength\glstreeindent\parindent
      \parindent\glstreeindent
    \else
      \ifnum\@gls@prevlevel=0\relax
        \glsxtrComputeTreeIndent{##2}%
      \else
        \glsxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
      \fi
      \addtolength\parindent{-\glstreeindent}%
      \setlength\glstreeindent\parindent
    \fi
  \fi
  \glsxtrAltTreeSetSubHangIndent{##1}%
  \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
    \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
  \glsxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
  \def\@gls@prevlevel{##1}%
```

```
    }%
    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
  }
}%
{%
```

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 firstplural 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 210
- \glsfmtshort: new 209
- \Glsfmtshortpl: new 210
- \glsfmtshortpl: new 210
- short: switched inline full form to short (long) 172

0.3 (2015-12-02)

- \@ACRlong: added redefinition 55
- \@ACRlongpl: added redefinition 56
- \@ACRshort: added redefinition 53
- \@ACRshortpl: added redefinition 54
- \@Acrlong: added redefinition 54
- \@Acrlongpl: added redefinition 55
- \@Acrshort: added redefinition 52
- \@Acrshortpl: added redefinition 53
- \@GLSdesc@: added redefinition 48
- \@GLSdescplural@: added redefinition . 49
- \@GLSfirst@: added redefinition 46
- \@GLSfirstplural@: added redefinition 47
- \@GLSname@: added redefinition 48
- \@GLSplural@: added redefinition 46
- \@GLSsymbol@: added redefinition 49
- \@GLSsymbolplural@: added redefinition 50
- \@GLStext@: added redefinition 45
- \@GLSuseri@: added redefinition 50
- \@GLSuserii@: added redefinition 50
- \@GLSuseriii@: added redefinition 51
- \@GLSuseriv@: added redefinition 51
- \@GLSuserv@: added redefinition 51
- \@GLSuservi@: added redefinition 52
- \@Glsdesc@: added redefinition 48
- \@Glsdescplural@: added redefinition . 48
- \@Glsfirst@: added redefinition 46
- \@Glsfirstplural@: added redefinition 47
- \@Glsname@: added redefinition 48
- \@Glsplural@: added redefinition 46
- \@Glssymbol@: added redefinition 49
- \@Glssymbolplural@: added redefinition 50
- \@Glstext@: added redefinition 45
- \@Glsuseri@: added redefinition 50
- \@Glsuserii@: added redefinition 50
- \@Glsuseriii@: added redefinition 51
- \@Glsuseriv@: added redefinition 51
- \@Glsuserv@: added redefinition 51
- \@Glsuservi@: added redefinition 51
- \@acrlong: added redefinition 54
- \@acrlongpl: added redefinition 55
- \@acrshort: added redefinition 52
- \@acrshortpl: added redefinition 53
- \@gls@field@link: added optional argument 40
- \@glsdescplural@: added redefinition . 48
- \@glsfirst@: added redefinition 45
- \@glsfirstplural@: added redefinition 47
- \@glsplural@: added redefinition 46
- \@glssymbolplural@: added redefinition 49
- \@glsxtr@defaultnoglossarywarning: new 102
- \@glsxtr@field@linkdefs: new 44
- \@glsxtr@insertdots: new 144
- \@print@glossary: added redefinition . 99
- \glsabbrvdefaultfont: renamed from \abbrvdefaultfont 148
- \glsaccessdesc: new 114
- \glsaccessdescplural: new 115
- \glsaccessfirst: new 112
- \glsaccessfirstplural: new 112
- \Glsaccesslong: new 117
- \glsaccesslong: new 116
- \glsaccessname: new 110
- \glsaccessplural: new 111
- \Glsaccessshort: new 115
- \glsaccessshort: new 115
- \Glsaccessshortpl: new 116

\glsaccessshortpl: new 116
\glsaccesssymbol: new 113
\glsaccesssymbolplural: new 113
\glsaccesstext: new 111
\glsentryfmt: added check for short . . 39
\glslongpltok: new 144
\glsshortpltok: new 144
\glsxtr@newabbreviation: fixed family name in \setkeys 145
\glsxtrdiscardperiod: added check for plural 141
\GLSxtrlongpl: new 158
\Glsxtrlongpl: new 157
\glsxtrlongpl: new 157
\glsxtrNoGlossaryWarning: new 14
\glsxtrpostlinkAddDescOnFirstUse: new 141
\glsxtrpostlinkAddSymbolOnFirstUse: new 141
\glsxtrpostlinkendsentence: new . 140
\GLSxtrshortpl: new 156
\Glsxtrshortpl: new 155
\glsxtrshortpl: new 155
short-long-desc: fixed name to use \glslabeltok 167
long-short-desc: fixed name to use \glslabeltok 165
0.4 (2015-12-03)
\@glsxtr@doabbreviationsdef: added redefinition of \acronymtype 12
\Glsfmtshort: changed to use \Glsxtrshort 210
\glsfmtshort: changed to use \glsxtrshort 209
\Glsfmtshortpl: changed to use \glsxtrshortpl 210
\glsfmtshortpl: changed to use \glsxtrshortpl 210
\glsxtrifemptyglossary: new 17
\glsxtrnewnumber: added extra argument 125
\glsxtrnewsymbol: added extra argument 125
\MakeAcronymsAbbreviations: set the default type to \acronymtype 87
\newterm: fixed name argument 124
0.5 (2015-12-07)
\@cGLS: new 78
\@cGLS@: new 78
\@cGLSpl: new 78
\@cGLSpl@: new 79
\@glsxtr@setentrycountunsetattr: new 74
\cGLS: new 78
\cGLSformat: new 78
\cGLSpl: new 78
\cGLSplformat: new 79
\GlossariesExtraWarningNoLine: new 11
\glsenableentrycount: new 74
\glsfirstabbrvdefaultfont: new . . 148
\glsfirstlongdefaultfont: new . . . 148
\Glsfmtfirst: new 212
\glsfmtfirst: new 212
\Glsfmtfirstpl: new 212
\glsfmtfirstpl: new 212
\Glsfmtplural: new 211
\glsfmtplural: new 211
\Glsfmtshort: changed to use \Glsxtrtitleshort 210
renamed from \Glsentryfmtshort . 210
\glsfmtshort: changed to use \glsxtrtitleshort 209
renamed from \glsentryfmtshort . 209
\Glsfmtshortpl: changed to use \Glsxtrtitleshortpl 210
renamed from \Glsentryfmtshortpl 210
\glsfmtshortpl: changed to use \glsxtrtitleshortpl 210
renamed from \glsentryfmtshortpl 210
\Glsfmttext: new 211
\glsfmttext: new 210
\glshasattribute: new 122
\glshascategoryattribute: new . . . 121
\GlsXtrEnableEntryCounting: new . . 73
\glsxtrifcounttrigger: new 76
\glsxtrscfont: new 176
\glsxtrscsuffix: new 176
\glsxtrsmfont: new 180
\glsxtrsmsuffix: new 180
short-em: new 187
short-em-desc: new 187
short-em-footnote: new 189
short-em-long: new 185
short-em-long-desc: new 186
short-em-postfootnote: new 189

short-sc-footnote: new 179
short-sc-postfootnote: new 180
short-sm: new 181
short-sm-desc: new 182
short-sm-footnote: new 183
short-sm-long: new 181
short-sm-long-desc: new 181
short-sm-postfootnote: new 183
long-noshort-em: new 188
long-noshort-em-desc: new 188
long-noshort-sm: new 182
long-noshort-sm-desc: new 182
long-short-em: new 184
long-short-em-desc: new 184
long-short-sm: new 180
long-short-sm-desc: new 181

0.5.1 (2015-12-02)
\Glsaccesstext: new 111

0.5.1 (2015-12-07)
\@glsxtr@doaccsupp: new 14
General: removed \ifglsxtruseuchead 201
\Glsaccessdesc: new 114
\Glsaccessdescplural: new 115
\Glsaccessfirst: new 112
\Glsaccessfirstplural: new 113
\Glsaccessname: new 110
\Glsaccessplural: new 111
\Glsaccesssymbol: new 113
\Glsaccesssymbolplural: new 114
\Glsxtrheadfirst: now uses headuc attribute 205
\glsxtrheadfirst: now uses headuc attribute 204
\Glsxtrheadfirstplural: now uses headuc attribute 205
\glsxtrheadfirstplural: now uses headuc attribute 205
\Glsxtrheadplural: now uses headuc attribute 204
\glsxtrheadplural: now uses headuc attribute 203
\Glsxtrheadshort: now uses headuc attribute 202
\glsxtrheadshort: now uses headuc attribute 201
\Glsxtrheadshortpl: now uses headuc attribute 202
\glsxtrheadshortpl: now uses headuc attribute 201
\Glsxtrheadtext: now uses headuc attribute 203
\glsxtrheadtext: now uses headuc attribute 203
short-long: switch off regular attribute if set 166
short-long-desc: switch off regular attribute if set 167
long-short: switch off regular attribute if set 164
long-short-desc: switch off regular attribute if set 165
footnote: switch off regular attribute if set 168
postfootnote: switch off regular attribute if set 170

0.5.2 (2015-12-08)
\@GLSdesc@: added accessibility support 48
\@GLSdescplural@: added accessibility support 49
\@GLSfirst@: added accessibility support 46
\@GLSfirstplural@: added accessibility support 47
\@GLSname@: added accessibility support 48
\@GLSplural@: added accessibility support 46
\@GLSsymbol@: added accessibility support 49
\@GLSsymbolplural@: added accessibility support 50
\@GLStext@: added accessibility support 45
\@Glsdesc@: added accessibility support 48
\@Glsdescplural@: added accessibility support 48
\@Glsfirst@: added accessibility support 46
\@Glsfirstplural@: added accessibility support 47
\@Glsname@: add accessibility support . . 48
\@Glsplural@: added accessibility support 46
\@Glssymbol@: added accessibility support 49
\@Glssymbolplural@: added accessibility support 50
\@Glstext@: added accessibility support 45
\@glsdesc@: added accessibility support 48

\@glsdescplural@: added accessibility support 48
\@glsfirst@: added accessibility support 45
\@glsfirstplural@: added accessibility support 47
\@glsname@: added accessibility support 47
\@glsplural@: added accessibility support 46
\@glssymbol@: added accessibility support 49
\@glssymbolplural@: added accessibility support 49
\@glstext@: added accessibility support 45
\@glsxtr@activate@initialtagging: new 139
\@glsxtr@do@titlecaps@warn: new . 138
\@glsxtr@tag: new 139
General: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded 110
removed \glsxtrabbrvfmt 158
\glossaryentrynumbers: added 37
\Glossentrydesc: added 136
\Glossentryname: added 131
\Glossentrysymbol: added 137
\glossentrysymbol: added 137
\GLSaccessdesc: new 114, 119
\GLSaccessdescplural: new . . . 115, 119
\GLSaccessfirst: new 112, 118
\GLSaccessfirstplural: new . . 113, 118
\GLSaccesslong: new 117, 120
\GLSaccesslongpl: new 117, 120
\Glsaccesslongpl: new 117
\glsaccesslongpl: new 117
\GLSaccessname: new 110, 118
\GLSaccessplural: new 112, 118
\GLSaccessshort: new 116, 120
\GLSaccessshortpl: new 116, 120
\GLSaccesssymbol: new 113, 119
\GLSaccesssymbolplural: new . 114, 119
\GLSaccesstext: new 111, 118
\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here 39
\GlsXtrEnableInitialTagging: new 137
\glsxtrfieldtitlecase: new 126
\GlsXtrFormatLocationList: new . . . 37
\glsxtrnewabbrevpresetkeyhook: new 146
\glsxtrtagfont: new 139
\KV@printgloss@nonumberlist: added 39
\mfu@checkword@do: added 138
\setabbreviationstyle: added check for post-definition style switch 161

0.5.3 (2015-12-09)
\@glsxtr@autoindex@at: new 134
\@glsxtr@autoindex@encap: new . . . 135
\@glsxtr@autoindex@esc: new 135
\@glsxtr@autoindex@level: new . . . 135
\@glsxtr@autoindex@setname: new . 133
\@glsxtr@doabbreviationsdef: new . 12
General: removed \GlsXtrNoGlsWarningNoAutoMakeMain 101
\glsdescwidth: added 36
\glspagelistwidth: added 37
\glsxtrdoautoindexname: new 133
\glsxtrpostnamehook: new 132
\if@glsxtr@format@override: new . 132
\ProvidesGlossariesExtraLang: new 215
\RequireGlossariesExtraLang: new 215

0.5.4 (2015-12-15)
\@@newglossaryentry@defunitcounters: new 79
\@GLSxtr@p@acrlong@: new 67
\@GLSxtr@p@acrlongpl@: new 67
\@GLSxtr@p@acrshort@: new 66
\@GLSxtr@p@acrshortpl@: new 66
\@GLSxtr@p@long@: new 66
\@GLSxtr@p@longpl@: new 66
\@GLSxtr@p@plural@: new 65
\@GLSxtr@p@short@: new 65
\@GLSxtr@p@shortpl@: new 66
\@GLSxtr@p@text@: new 64
\@GlsXtrEnableOnTheFly: new 33
\@Glsxtr: new 33
\@Glsxtr@p@acrlong@: new 67
\@Glsxtr@p@acrlongpl@: new 67
\@Glsxtr@p@acrshort@: new 66
\@Glsxtr@p@acrshortpl@: new 66
\@Glsxtr@p@long@: new 66
\@Glsxtr@p@longpl@: new 66
\@Glsxtr@p@plural@: new 65
\@Glsxtr@p@short@: new 65
\@Glsxtr@p@shortpl@: new 65
\@Glsxtr@p@text@: new 64

\@Glsxtrpl: new 34
\@alt@gls@hyp@opt: new 61
\@gls@alt@hyp@opt: new 61
\@gls@alt@hyp@opt@char: new 62
\@gls@alt@hyp@opt@keys: new 62
\@gls@increment@currunitcount:
new 80
\@gls@local@increment@currunitcount:
new 81
\@gls@setdefault@glslink@opts:
new 59
\@glsxtr: new 33
\@glsxtr@addunitcounter: new 80
\@glsxtr@currunitcount: new 81
\@glsxtr@ifunitcounter: new 80
\@glsxtr@p@acrlong@: new 67
\@glsxtr@p@acrlongpl@: new 67
\@glsxtr@p@acrshort@: new 66
\@glsxtr@p@acrshortpl@: new 66
\@glsxtr@p@long@: new 66
\@glsxtr@p@longpl@: new 66
\@glsxtr@p@plural@: new 65
\@glsxtr@p@short@: new 65
\@glsxtr@p@shortpl@: new 65
\@glsxtr@p@text@: new 64
\@glsxtr@prevunitcount: new 81
\@glsxtr@setentryunitcountunsetattr:
new 85
\@glsxtr@unitcountlist: new 80
\@glsxtrpl: new 34
\@newglossaryentryposthook: added
empty see value if not set and added
'see' to field key map 29
\@sGlsXtrEnableOnTheFly: new 32
\cGlsformat: added 79
\cglsformat: added 79
\cGlsplformat: added 79
\cglsplformat: added 79
\glsdisablehyper: added 63
\glsdohyperlink: added 62
\glsdonohyperlink: added 63
\glsenableentryunitcount: new 82
\glshasattribute: added check for
entry's existence 122
\glsifattribute: added check for
entry's existence 122
\glspostlinkhook: added existence
check 140
\Glsxtr: new 33
\glsxtr: new 33
\glsxtrcat: new 33
\glsxtrdowrglossaryhook: new 61
\GlsXtrEnableEntryUnitCounting:
new 85
\GlsXtrEnableOnTheFly: new 32
\Glsxtrpl: new 34
\glsxtrpl: new 34
\glsxtrpostlocalreset: new 73
\glsxtrpostlocalunset: new 73
\glsxtrpostreset: new 73
\glsxtrpostunset: new 72
\glsxtrprotectlinks: new 64
\GlsXtrSetAltModifier: new 62
\GlsXtrSetDefaultGlsOpts: new 60
\glsxtrstarflywarn: new 33
\GlsXtrWarning: new 34
\MakeAcronymsAbbreviations: now
disables \setacronymstyle 87
1.0 (2016-01-24)
\@glsxtr@autoindexcrossrefs: new . 11
\@glsxtr@idx@displaynumberlist:
new 93
\@glsxtr@idx@entrynumberlist: new 95
\@glsxtr@noidx@displaynumberlist:
new 94
\@glsxtr@noidx@entrynumberlist:
new 95
\@glsxtr@noidx@numberlistloop:
new 94
\@glsxtr@reg@glosslist: new 88
\makeglossaries: new 88
1.01 (2016-02-02)
\glsxtrdiscardperiod: added check
for first use 141
short-desc: fixed typo in
\glsxtrinlinefullformat and
added missing second argument . . . 173
1.02 (2016-04-25)
\@glsxtr@current@style: new 35
\Glsfmtfull: new 214
\glsfmtfull: new 214
\Glsfmtfullpl: new 215
\glsfmtfullpl: new 214
\Glsfmtlong: new 213
\glsfmtlong: new 213
\Glsfmtlongpl: new 214
\glsfmtlongpl: new 213
\Glsxtrheadfull: new 208

\glsxtrheadfull: new 208
\Glsxtrheadfullpl: new 209
\glsxtrheadfullpl: new 208
\Glsxtrheadlong: new 207
\glsxtrheadlong: new 206
\Glsxtrheadlongpl: new 207
\glsxtrheadlongpl: new 206
\Glsxtrtitlefull: new 209
\glsxtrtitlefull: new 208
\Glsxtrtitlefullpl: new 209
\glsxtrtitlefullpl: new 208
\Glsxtrtitlelong: new 207
\glsxtrtitlelong: new 206
\Glsxtrtitlelongpl: new 207
\glsxtrtitlelongpl: new 207
\ifglsxtrinsertinside: new 164
postfootnote: added redef of \glsxtrsetupfulldefs 170
stylemods: new 15

1.03 (2016-04-27)
\@GLSfirstplural@: bug fix: misspelt cs name 47
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural 46
\@Glsfirstplural@: bug fix: misspelt cs name 47
\@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural 46
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural 46
\glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl . . 207
\glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl 202

1.04 (2015-04-30)
short-em-footnote: renamed from “footnote-em” 189

1.04 (2016-05-02)
\@@glsxtrpostloctag: new 39
\@GLSdesc@: set abbreviation and regular format 48
\@GLSdescplural@: set abbreviation and regular format 49
\@GLSfirst@: set abbreviation format . . 46
\@GLSfirstplural@: set abbreviation and regular format 47
\@GLSname@: set abbreviation and regular format 48
\@GLSplural@: set abbreviation and regular format 46
\@GLSsymbol@: set regular format 49
\@GLSsymbolplural@: set regular format 50
\@GLStext@: set abbreviation and regular format 45
\@GLSuseri@: set regular format 50
\@GLSuserii@: set regular format 50
\@GLSuseriii@: set regular format 51
\@GLSuseriv@: set regular format 51
\@GLSuserv@: set regular format 51
\@GLSuservi@: set regular format 52
\@Glsdesc@: set abbreviation and regular format 48
\@Glsdescplural@: set abbreviation and regular format 48
\@Glsfirst@: set abbreviation and regular format 46
\@Glsfirstplural@: set abbreviation and regular format 47
\@Glsname@: set abbreviation and regular format 48
\@Glsplural@: set abbreviation and regular format 46
\@Glssymbol@: set regular format 49
\@Glssymbolplural@: set regular format 50
\@Glstext@: set abbreviation and regular format 45
\@Glsuseri@: set regular format 50
\@Glsuserii@: set regular format 50
\@Glsuseriii@: set regular format 51
\@Glsuseriv@: set regular format 51
\@Glsuserv@: set regular format 51
\@Glsuservi@: set regular format 51
\@gls@preglossaryhook: added check for entry’s existence 139
\@glsdesc@: set abbreviation and regular format 48
\@glsdescplural@: set abbreviation and regular format 48
\@glsfirst@: set abbreviation and regular format 45
\@glsfirstplural@: set abbreviation and regular format 47
\@glsname@: set abbreviation and regular format 47
\@glsplural@: set abbreviation and regular format 46
\@glssymbol@: set regular format 49

\@glssymbolplural@: set regular format 49
\@glstext@: set abbreviation and regular format . . . 45
\@glsxtr@deprecated@abbrstyle: new . . . 162
\@glsxtr@do@style: new . . . 15
\@glsxtr@doloctag: new . . . 39
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand 95
\@glsxtr@pagestag: new . . . 38
\@glsxtr@pagetag: new . . . 38
\@glsxtr@preloctag: new . . . 39
\@glsxtrpostloctag: new . . . 39
\@glsxtrpreloctag: new . . . 38
\glossentrydesc: added glossdescfont attribute check . . . 127
\Glossentryname: added glossnamefont attribute check . . . 131
\glossentryname: added glossnamefont attribute check . . . 128
  moved post name hook inside condition . . . 130
\glsabbrvemfont: new . . . 184
\glsabbrvuserfont: new . . . 190
\glsfirstabbrvemfont: new . . . 184
\glsfirstabbrvuserfont: new . . . 190
\glsfirstlongemfont: new . . . 184
\glsfirstlonguserfont: new . . . 191
\glsifnotregularcategory: new . . . 123
\glslongdefaultfont: new . . . 148
\glslongemfont: new . . . 184
\glslongfont: new . . . 148
\glslonguserfont: new . . . 190
\glsxtrassignfieldfont: new . . . 44
\GlsXtrEnablePreLocationTag: new . 38
\glsxtrfirstscfont: new . . . 176
\glsxtrfirstsmfont: new . . . 180
\glsxtrlongshortdescsort: new . . . 165
\glsxtrpostnamehook: added category check . . . 132
\glsxtrregularfont: new . . . 40
\glsxtruserfield: new . . . 190
\glsxtruserparen: new . . . 190
\glsxtrusersuffix: new . . . 191
\GlsXtrWarnDeprecatedAbbrStyle: new . . . 163
short-em-long-em: new . . . 186
short-em-long-em-desc: new . . . 187
short-em-nolong: new . . . 187
short-em-nolong-desc: new . . . 187
short-em-postfootnote: renamed from "postfootnote-em" . . . 189
short-footnote: new . . . 169
short-long-user: new . . . 196
short-long-user-desc: new . . . 198
short-nolong: new . . . 173
short-nolong-desc: new . . . 174
short-postfootnote: new . . . 171
short-sc-footnote: renamed from "footnote-sc" . . . 179
short-sc-nolong: new . . . 178
short-sc-nolong-desc: new . . . 178
short-sc-postfootnote: renamed from "postfootnote-sc" . . . 180
short-sm-footnote: renamed from "footnote-sm" . . . 183
short-sm-nolong: new . . . 182
short-sm-nolong-desc: new . . . 182
short-sm-postfootnote: renamed from "postfootnote-sm" . . . 183
\letabbreviationstyle: new . . . 162
\newabbreviationstyle: bug fix: corrected test for existence . . . 161
long-em-noshort-em: new . . . 188
long-em-noshort-em-desc: new . . . 189
long-em-short-em: new . . . 184
long-em-short-em-desc: new . . . 185
long-noshort: new . . . 176
long-noshort-desc: new . . . 176
long-noshort-em: renamed from "long-em" . . . 188
long-noshort-em-desc: renamed from "long-desc-em" . . . 188
long-noshort-sc: renamed from "long-sc" . . . 178
long-noshort-sc-desc: renamed from "long-desc-sc" . . . 179
long-noshort-sm: renamed from "long-sm" . . . 182
long-noshort-sm-desc: renamed from \long-desc-sm . . . 182
long-short-user: new . . . 191
long-short-user-desc: new . . . 196
\renewabbreviationstyle: new . . . 162
style: new . . . 15
1.05 (2016-06-10)
  \eglssetwidest: new . . . 224
  \glsFindWidestAnyName: new . . . 226

\glsFindWidestAnyNameLocation:
new 231
\glsFindWidestAnyNameSymbol: new 229
\glsFindWidestAnyNameSymbolLocation:
new 230
\glsFindWidestLevelTwo: new 227
\glsFindWidestUsedAnyName: new . . 225
\glsFindWidestUsedAnyNameLocation:
new 231
\glsFindWidestUsedAnyNameSymbol:
new 228
\glsFindWidestUsedAnyNameSymbolLocation:
new 229
\glsFindWidestUsedLevelTwo: new . 226
\glsFindWidestUsedTopLevelName:
new 225
\glsfirstlongfootnotefont: new . . 167
\glsgetwidestname: new 224
\glsgetwidestsubname: new 225
\glslongfootnotefont: new 167
\glsxtrAltTreeIndent: new 224
\glsxtralttreeInit: new 224
\glsxtrAltTreePar: new 224
\glsxtrAltTreeSetHangIndent: new 232
\glsxtrAltTreeSetSubHangIndent:
new 232
\glsxtralttreeSubSymbolDescLocation:
new 224
\glsxtralttreeSymbolDescLocation:
new 223
\glsxtrComputeTreeIndent: new . . . 232
\glsxtrComputeTreeSubIndent: new 232
\glsxtrtreetopindent: new 224
short-em-long: fixed incorrect font used
by long form 185
\xglssetwidest: new 224
1.06 (2016-06-18)
\@glsdoifexistsorwarn: new 11
\@glsxtr@docdefval: new 10
\@glsxtr@usesee: new 30
General: disabled docdef key at the start
of the document 17
docdef option changed to choice 10
\glsxtr@usesee: new 29
\glsxtrusesee: new 29
\glsxtruseseeformat: new 30
\if@glsxtrdocdefrestricted: new . . 11
1.07 (2016-08-15)
\@@glsxtrp: new 67
\@GLSfirst@: added check for
nohyperfirst attribute 46
\@GLSfirstplural@: added check for
nohyperfirst attribute 47
\@GLSxtrp: new 68
\@Glsfirst@: added check for
nohyperfirst attribute 46
\@Glsfirstplural@: added check for
nohyperfirst attribute 47
\@Glsxtrp: new 68
\@gls@preglossaryhook: added
\glossxtrsetpopts 139
\@glsfirst@: added check for
nohyperfirst attribute 45
\@glsfirstplural@: added check for
nohyperfirst attribute 47
\@glsxtrinmark: new 199
\@glsxtrnotinmark: new 199
\@glsxtrp: new 68
\@glsxtrp@opt: new 67
\glossxtrsetpopts: new 67
\glsps: new 70
\glspt: new 70
\glsxtr@entry@p: new 69
\glsxtrabbrvfootnote: new 168
\glsxtrchecknohyperfirst: new 45
\glsxtrfieldtitlecasecs: new 126
\glsxtrifinmark: new 199
\GLSxtrp: new 71
\Glsxtrp: new 70
\glsxtrp: new 69
\glsxtrsetpopts: new 67
short-long-desc: added text key 167
fixed misspelling of \glsabbrvfont in
plural key 167
long-short-desc: added missing text
key 165
fixed misspelling of \glsabbrvfont . 165
footnote: changed first forms to use
\glsfirstlongfootnotefont . . . 168
postfootnote: removed \footnote
from first keys 169
switched from \glsfirstlongfont to
\glsfirstlongfootnotefont . . . 171
\RestoreAcronyms: modified
\@gls@link@checkfirsthyper to
set \glsxtrifwasfirstuse 88
1.08 (2016-12-13)
\@@glsxtr@record: new 6

\@GLS@: added \@glsxtr@record 41
\@GLSpl@: added \@glsxtr@record ... 41
\@Gls@: added \@glsxtr@record 41
\@Glspl@: added \@glsxtr@record ... 41
\@gls@: added \@glsxtr@record 40
\@gls@@link@: added \@glsxtr@record 42
\@gls@field@link: added \@glsxtr@record 40
\@gls@saveentrycounter: new 17
\@glsdisp: added \@glsxtr@record .. 41
\@glspl@: added \@glsxtr@record ... 41
\@glsxtr@dorecord: new 7
\@glsxtr@err@undefaction: new 5
\@glsxtr@record: new 6
\@glsxtr@warn@onexistsordo: new ... 5
\@glsxtr@warn@undefaction: new 5
\@print@unsrt@glossary: new 107
General: added record package option 9
\glsadd: added \@glsxtr@record 44
\glsdoifexists: now defines \glslabel 28
\glsxtr@@do@wrglossary: new 17
\glsxtr@addloclistfield: new 8
\glsxtr@indexonly@saveentrycounter: new 8
\glsxtr@record: new 106
\glsxtr@resource: new 104
\glsxtr@saveentrycounter: new 17
\glsxtr@setup@record: new 8
\glsxtrassignfieldfont: added check for existence 44
\glsxtrresourcefile: new 104
\printunsrtglossaries: new 107
\printunsrtglossary: new 106

1.09 (2016-12-16)
\@glsxtr@gettype: new 93
\@glsxtr@mixed@assign@sortkey: new 93
\@printglossary: redefined to save options 93
\glsxtr@makeglossaries: new 93

1.10 (2016-12-17)
\@GLSpl@: fixed bug caused by typo in command name 41

1.11 (2017-01-19)
\@glsxtr@do@redef@forglsentries: new 5
\@glsxtr@noidx@do: new 108
\@glsxtr@redef@forglsentries: new . 5
\@glsxtr@shortcutsval: new 13
\@glsxtr@unsrt@getgrouptitle: new 108
\@print@noidx@glossary: added redefinition 96
\glsxtr@addloclistfield: added group key 8
added location key 8
\glsxtr@fields: new 104
\glsxtr@linkprefix: new 105
\glsxtr@org@newignoredglossary: new 25
\glsxtr@s@newignoredglossary: new 25
\glsxtr@shortcutsval: new 105
\glsxtr@texencoding: new 104
\glsxtr@writefields: new 105
\GlsXtrLoadResources: new 104
\glsxtrresourcefile: changed extension to .glstex 104
\newignoredglossary: added starred version 24

1.12 (2017-02-03)
\@@glsxtr@recordcounter: new 7
\@gls@preglossaryhook: check for definition 139
\@glsxtr@counterrecordhook: new . 106
\@glsxtr@display@loc: new 97
\@glsxtr@docounterrecord: new ... 106
\@glsxtr@longnewglossaryentry: new 24
\@glsxtr@noop@recordcounter: new .. 7
\@glsxtr@op@recordcounter: new 7
\@glsxtr@provide@storagekey: new . 18
\@glsxtr@s@longnewglossaryentry: new 24
\@glsxtrentryfmt: new 19
\@glsxtrindexaliased: new 60
\@glsxtrsetaliasnoindex: new 59
\@newglossaryentryposthook: added check for alias key 23
\@no@glsxtrindexaliased: new 60
\@printunsrtglossary: new 107
General: added target key to printgloss family 93
\apptoglossarypreamble: new 22
\csGlsXtrLetField: new 21
\eGlsXtrSetField: new 22
\gGlsXtrSetField: new 22

\glsdohyperlink: added check for alias field . . . 63
\glsnoidxdisplayloc: added redefinition . . . 97
\glssettoctitle: added patch . . . 26
\glsxtr@counterrecord: new . . . 106
\glsxtr@langtag: new . . . 104
\glsxtr@newabbreviation: new . . . 145
\glsxtr@org@newignoredglossary: Added check for existence . . . 25
\glsxtr@pluralsuffixes: new . . . 105
\glsxtr@provideignoredglossary: new . . . 26
\glsxtr@s@newignoredglossary: Added check for existence . . . 25
\glsxtr@s@provideignoredglossary: new . . . 27
\glsxtrabbrvpluralsuffix: new . . . 148
\glsxtralias: new . . . 23
\glsxtrcopytoglossary: new . . . 27
\glsxtrdeffield: new . . . 21
\glsxtrdisplayendloc: new . . . 98
\glsxtrdisplayendlochook: new . . . 98
\glsxtrdisplaysingleloc: new . . . 97
\glsxtrdisplaystartloc: new . . . 98
\glsxtredeffield: new . . . 21
\glsxtrentryfmt: new . . . 19
\glsxtrfielddolistloop: new . . . 20
\glsxtrfieldforlistloop: new . . . 20
\glsxtrfieldifinlist: new . . . 20
\glsxtrfieldlistadd: new . . . 20
\glsxtrfieldlisteadd: new . . . 20
\glsxtrfieldlistgadd: new . . . 20
\glsxtrfieldlistxadd: new . . . 20
\glsxtrfieldxifinlist: new . . . 21
\glsxtrfmt: new . . . 19
\GlsXtrFmtDefaultOptions: new . . . 19
\GlsXtrFmtField: new . . . 19
\glsxtrifkeydefined: new . . . 18
\glsxtrindexaliased: new . . . 60
\GlsXtrLetField: new . . . 21
\GlsXtrLetFieldToField: new . . . 21
\GlsXtrLoadResources: removed restriction on only one per document 104
\glsxtrlocrangefmt: new . . . 98
\glsxtrpostlongdescription: new . . . 24
\glsxtrprovidestoragekey: new . . . 18
\GlsXtrRecordCounter: new . . . 106
\glsxtrresourcecount: new . . . 104
\glsxtrresourcefile: added catcode change for @ . . . 104
\glsxtrsetaliasnoindex: new . . . 59
\GlsXtrSetField: new . . . 21
\glsxtrsetfieldifexists: new . . . 21
\glsxtrunsrtdo: new . . . 108
\Glsxtrusefield: new . . . 21
\glsxtrusefield: new . . . 21
short-postlong-user: new . . . 194
short-postlong-user-desc: new . . . 196
\longnewglossaryentry: added starred version . . . 23
long-postshort-user: new . . . 192
long-postshort-user-desc: new . . . 193
postdot: new . . . 11
\pretoglossarypreamble: new . . . 23
\print@noop@unsrtglossaryunit: new . . . 108
\print@op@unsrtglossaryunit: new 108
\printunsrtglossary: added starred form . . . 106
\printunsrtglossaryhandler: new . 107
\printunsrtglossaryunit: new . . . 8
\printunsrtglossaryunitsetup: new 108
\provideignoredglossary: new . . . 26
\s@glsxtr@provide@storagekey: new 18
\s@printunsrtglossary: new . . . 107
\xGlsXtrSetField: new . . . 22

1.13 (2017-02-07)
\@glsdisp: removed \@glsxtr@org@glsdisp . . . 41
\glsxtrsetaliasnoindex: switched to \providecommand . . . 59

1.14 (2017-04-18)
\@gls@link: added redefinition . . . 43
\@gls@noidx@getgrouptitle: new . . . 95
\@gls@removespaces: new . . . 98
\@glsxtr@do@automake@err: new . . . 106
\@glsxtr@org@gloautosee: new . . . 16
\@glsxtr@record: added third arg . . . 6
\@glsxtr@recordsee: new . . . 7
General: added \glsadd option theHvalue . . . 44
added \glsadd option thevalue . . . 43
\glsdisablehyper: added redefinition . 63
\glsenableentrycount: fixed assignment of \@cGls@ . . . 75
\glsenableentryunitcount: fixed assignment of \@cGls@ . . . 83

\glsnavigation: new 96
\glsxtr@org@getgrouptitle: new . . . 95
\glsxtr@recordsee: new 6
\glsxtr@writefields: added check for automake 105
\glsxtrdisplayendloc: added check for empty format 98
\glsxtrgetgrouptitle: new 96
\glsxtrinitwrgloss: new 42
\glsxtrlocationhyperlink: new 99
\glsxtrsetgrouptitle: new 96
\glsxtrsupphypernumber: new 99
\ifglsxtrwrglossbefore: new 42

1.15 (2017-05-10)

\@glsxtr@dorecord: corrected premature expansion of \@glslocref 7
short-em-long-em: fixed spelling of \glsabbrvfont 186
short-long: fixed spelling of \glsabbrvfont 166
short-long-user: fixed spelling of \glsabbrvfont 197
short-postlong-user: fixed spelling of \glsabbrvfont 194
short-postlong-user-desc: fixed spelling of \glsabbrvfont 196
long-em-short-em: fixed spelling of \glsabbrvfont 185
long-postshort-user: fixed spelling of \glsabbrvfont 192
long-postshort-user-desc: fixed spelling of \glsabbrvfont 194
long-short: fixed spelling of \glsabbrvfont 164
long-short-user: fixed spelling of \glsabbrvfont 191
footnote: fixed spelling of \glsabbrvfont 168
postfootnote: fixed spelling of \glsabbrvfont 170

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

\. . . . 140, 141, 223
\@ 104
\@@cGLS@ 75, 83
\@@cGLSpl@ 75, 83
\@@cGls@ 75, 83
\@@cGlspl@ 75, 83
\@@cgls@ 75, 83
\@@cglspl@ 75, 76, 83
\@@do@@wrglossary 7, 89
\@@do@wrglossary 9, 10, 17, 44, 60
\@@glo@assign@sortkey 93
\@@glo@list 5
\@@glo@type 107
\@@gls@expand@field 18
\@@glslocalreset 73
\@@glslocalunset 73
\@@glsreset 73
\@@glsunset 72
\@@glsxtr@autoindex@escspch 135, 136
\@@glsxtr@checkspch 133, 134, 136
\@@glsxtr@disabledflycommand 35
\@@glsxtr@record 9, 10
\@@glsxtr@recordcounter 9, 10, 106
\@@glsxtrp 68
\@@glsxtrpostloctag 38
\@@glsxtrpreloctag 38
\@@newglossaryentry@defcounters 74
\@@newglossaryentry@defunitcounters 82
\@@par 224
\@ACRlong 64
\@ACRlongpl 64
\@ACRshort 64
\@ACRshortpl 64
\@Acrlong 64
\@Acrlongpl 64
\@Acrshort 64
\@Acrshortpl 64
\@GLS@ 64, 77, 78
\@GLSdesc@ 49
\@GLSpl@ 64, 78, 79
\@GLSplural@ 65
\@GLSsymbol@ 50
\@GLStext@ 64
\@GLSxtr@full 150
\@GLSxtr@fullpl 151
\@GLSxtr@p@acrlong@ 64
\@GLSxtr@p@acrlongpl@ 64
\@GLSxtr@p@acrshort@ 64
\@GLSxtr@p@acrshortpl@ 64
\@GLSxtr@p@long@ 64
\@GLSxtr@p@longpl@ 64
\@GLSxtr@p@plural@ 64
\@GLSxtr@p@short@ 64
\@GLSxtr@p@shortpl@ 64
\@GLSxtr@p@text@ 64
\@GLSxtrlong 64, 154, 155
\@GLSxtrlongpl 64, 158
\@GLSxtrp 71, 72
\@GLSxtrshort 64, 153
\@GLSxtrshortpl 64, 156
\@Gls@ 64, 77, 78
\@Gls@acrentryname 86
\@Gls@entry@field 56, 70, 71
\@Gls@entryname 86
\@GlsXtrEnableOnTheFly 32
\@Glspl@ 64, 77, 78
\@Glsplural@ 65
\@Glstext@ 64
\@Glsxtr 33, 35
\@Glsxtr@full 149
\@Glsxtr@fullpl 151
\@Glsxtr@p@acrlong@ 64
\@Glsxtr@p@acrlongpl@ 64

\@Glsxtr@p@acrshort@ 64
\@Glsxtr@p@acrshortpl@ 64
\@Glsxtr@p@long@ 64
\@Glsxtr@p@longpl@ 64
\@Glsxtr@p@plural@ 64
\@Glsxtr@p@short@ 64
\@Glsxtr@p@shortpl@ 64
\@Glsxtr@p@text@ 64
\@Glsxtrlong 64, 154
\@Glsxtrlongpl 64, 157
\@Glsxtrp 70, 71
\@Glsxtrpl 34, 35
\@Glsxtrshort 64, 152
\@Glsxtrshortpl 64, 156
\@acrlong 64
\@acrlongpl 64
\@acrshort 64
\@acrshortpl 64
\@alt@gls@hyp@opt 61
\@auxout 7, 8, 39, 75, 84, 88, 89, 99, 100, 104–106
\@bibgls@restoreat 104
\@cGLS 78
\@cGLS@ 75, 78, 83
\@cGLSpl 78
\@cGLSpl@ 75, 78, 83
\@cGls@ 75, 83
\@cGlspl@ 75, 83
\@cgls@ 75, 83
\@cglspl@ 75, 83
\@disable@onlypremakeg 89
\@do@auxoutstuff 99, 100
\@do@glssee 23
\@do@newglossaryentry 86, 146
\@do@seeglossary 9, 10, 16, 89
\@do@wrglossary 43
\@empty 44, 52–56, 133, 134, 149–158
\@end@glsxtr@addunused 30
\@end@glsxtr@gettype 91, 93
\@end@glsxtr@usesee 29, 30
\@endfortrue 161
\@firstofone 44, 45, 127, 128, 132, 138
\@firstofthree 41, 44, 52–55, 61, 62, 149, 150, 152, 154, 155, 157
\@firstoftwo 45–50, 53–56, 58, 61, 88, 142, 143, 149–151, 155–158, 199, 200
\@for 5, 15, 31, 74, 85, 89, 91, 96, 107, 126, 137
\@glo@assign@sortkey 91
\@glo@autosee 9, 10, 16
\@glo@category 79
\@glo@counterprefix 7, 98
\@glo@countunit 79, 80
\@glo@default@sorttype 91
\@glo@desc 24
\@glo@descplural 24
\@glo@group 9
\@glo@label 8, 9, 18, 23, 29, 30, 56, 63, 225–232
\@glo@location 8
\@glo@loclist 8
\@glo@name 133
\@glo@no@assign@sortkey 93
\@glo@parent 226–228
\@glo@see 23, 29, 30
\@glo@sort 133
\@glo@sorttype 91, 96, 97
\@glo@text 42
\@glo@thislettergrp 109
\@glo@thisvalue 190
\@glo@tmp 18, 56
\@glo@type 30, 86, 89, 92, 93, 96, 97, 99, 100, 102, 103, 107
\@glo@types 124, 225–231
\@glossary@default@style 35, 36, 92, 234
\@glossarystyle 92
\@gls@ 64, 76, 78
\@gls@@link 42
\@gls@ReturnAfterFi 98
\@gls@actualchar 134
\@gls@adjustmode 44
\@gls@alt@hyp@opt 62
\@gls@alt@hyp@opt@char 61, 62
\@gls@alt@hyp@opt@keys 62
\@gls@automake 91
\@gls@between 96
\@gls@checkedmkidx 133, 134, 136
\@gls@checkmkidxchars 133
\@gls@codepage 100
\@gls@counter 6, 7, 43, 44, 60
\@gls@currentlettergroup 97, 107, 109
\@gls@declareoption 4
\@gls@doautomake 91, 106
\@gls@doautomake@err 106
\@gls@encapchar 134
\@gls@entry@count 75
\@gls@entry@field 18, 21, 56, 69–72, 74
\@gls@entry@unitcount 83, 84
\@gls@field@font 44–52
\@gls@field@link 45–52, 57, 58
\@gls@getgrouptitle 96, 107

\@gls@grptitle 96
\@gls@hyp@opt 57, 58, 62, 78, 149–158
\@gls@hyp@opt@cs 61, 62
\@gls@increment@currcount 74
\@gls@increment@currunitcount 82
\@gls@keymap 8, 9, 18, 29, 56, 105
\@gls@label 7, 61, 89, 106, 161
\@gls@levelchar 134
\@gls@link 19, 40, 42, 52–56, 149–158
\@gls@link@checkfirsthyper 41, 88
\@gls@link@label 43
\@gls@link@nocheckfirsthyper . . . . . .
. . . . 40, 52–56, 149–158
\@gls@link@opts 43
\@gls@list 96
\@gls@local@increment@currcount 74
\@gls@local@increment@currunitcount 83
\@gls@location 109, 110
\@gls@loclist 94, 95, 109, 110
\@gls@longpl 144–146
\@gls@nohyperlist 25, 27
\@gls@noidx@do 97
\@gls@noidx@nosanitizesort 91
\@gls@noidx@sanitizesort 91
\@gls@noidxloclist@finalsep 94
\@gls@noidxloclist@prev 94
\@gls@noidxloclist@sep 94
\@gls@noref@warn 90, 97
\@gls@org@glsnoidxdisplayloc 94
\@gls@org@glsseeformat 94, 95
\@gls@preglossaryhook 93, 138
\@gls@prevlevel 233
\@gls@quotechar 133
\@gls@reference 31, 89
\@gls@saveentrycounter . . . 9, 10, 17, 43, 44
\@gls@see@noindex 104
\@gls@setdefault@glslink@opts . . . 43, 60
\@gls@setsort 43
\@gls@short 145
\@gls@shortpl 144–146
\@gls@sort 109
\@gls@tmp 96
\@gls@tmpb 136
\@gls@type 89–91, 161, 225–231
\@gls@write@entrycounts 75
\@gls@write@entryunitcounts 83
\@gls@write@entryunitcounts@do 84
\@gls@xref 8
\@glsabbrv@current@abbreviation 145, 158
\@glsacronymlists 86
\@glsdoifexistsorwarn . . . 10, 128, 129, 131
\@glsentry 75, 84
\@glslink 43, 63, 64
\@glslocref 7
\@glsnextpages 92
\@glsnonextpages 92
\@glsnumberformat . . 6, 7, 43, 44, 60, 132, 133
\@glsorder 88, 89
\@glsspl@ 64, 77, 78
\@glsplural@ 65
\@glspunc@token 142
\@glsstyle@alttree 223
\@glsstyle@inline 223
\@glsstyle@listdotted 218
\@glstarget 63, 93
\@glstext@ 64
\@glswidestname 224, 225, 232
\@glsxtr 33, 35
\@glsxtr@@do@@wrglossary 89
\@glsxtr@abbreviationsdef 12, 16
\@glsxtr@activate@initialtagging . .
. . . . 138, 139
\@glsxtr@addunitcounter 80
\@glsxtr@addunusedxrefs 30, 31
\@glsxtr@attrval . . . 127, 128, 130, 131, 133
\@glsxtr@autoindex@at 133–135
\@glsxtr@autoindex@doextra@esc 133
\@glsxtr@autoindex@encap 133–135
\@glsxtr@autoindex@esc 133–136
\@glsxtr@autoindex@escat 134, 135
\@glsxtr@autoindex@escencap . . . 134, 135
\@glsxtr@autoindex@esclevel . . . 134, 135
\@glsxtr@autoindex@escquote . . . 133, 135
\@glsxtr@autoindex@level 134, 135
\@glsxtr@autoindex@setname 133
\@glsxtr@autoindexcrossrefs . . . 9, 11, 29
\@glsxtr@cat 74, 85, 137
\@glsxtr@counterrecordhook 7
\@glsxtr@csname 80, 81, 83
\@glsxtr@current@style 36, 234
\@glsxtr@currentunitcount 80, 81, 83
\@glsxtr@currunitcount 82, 84
\@glsxtr@declareoption 4, 11, 12, 14
\@glsxtr@defaultnoglossarywarning 14, 15
\@glsxtr@deprecated@abbrstyle . . . . .
. . . . 179, 180, 182, 183, 188–190
\@glsxtr@disabledflycommand 35
\@glsxtr@display@loc 97

\@glsxtr@do@@wrindex 61
\@glsxtr@do@glsdisablehyperinlist 58, 59
\@glsxtr@do@redef@forglsentries 6
\@glsxtr@do@style 15, 216
\@glsxtr@do@titlecaps@warn . 127–130, 138
\@glsxtr@doabbreviationsdef 12
\@glsxtr@doaccsupp 14, 16
\@glsxtr@docdefval 10, 11, 31
\@glsxtr@docounterrecord 7
\@glsxtr@doloctag 38
\@glsxtr@dorecord 7
\@glsxtr@dostylewarn 161
\@glsxtr@enabletagging 137
\@glsxtr@end@ 32
\@glsxtr@endescspch 134–136
\@glsxtr@entrycount@org@localreset . 75
\@glsxtr@entrycount@org@localunset . 74
\@glsxtr@entrycount@org@reset 75
\@glsxtr@entrycount@org@unset 74
\@glsxtr@entryunitcount@org@localreset
. . . . 83
\@glsxtr@entryunitcount@org@localunset
. . . . 82, 83
\@glsxtr@entryunitcount@org@reset . . 83
\@glsxtr@entryunitcount@org@unset . . 82
\@glsxtr@err@undefaction 6, 9
\@glsxtr@field@linkdefs 40
\@glsxtr@format@overridefalse 132
\@glsxtr@format@overridetrue . . 132, 133
\@glsxtr@foundinlist 143
\@glsxtr@full 149
\@glsxtr@fullpl 150
\@glsxtr@gettype 91
\@glsxtr@glossdescfont 127, 128
\@glsxtr@glossnamefont 128–132
\@glsxtr@gobbleto@endescspch 136
\@glsxtr@idx@displaynumberlist 90
\@glsxtr@idx@entrynumberlist 90
\@glsxtr@ifcsstart 32
\@glsxtr@ifpunctoken 143
\@glsxtr@ifunitcounter 79
\@glsxtr@insert@dots 144
\@glsxtr@insert@dots@next 144, 145
\@glsxtr@insertdots 145
\@glsxtr@label 31, 126
\@glsxtr@loadstyles 217
\@glsxtr@longnewglossaryentry 23
\@glsxtr@mixed@assign@sortkey 91
\@glsxtr@noidx@displaynumberlist . . . 90
\@glsxtr@noidx@do 108
\@glsxtr@noidx@entrynumberlist 90
\@glsxtr@noidx@getgrouptitle 107
\@glsxtr@noidx@numberlistloop 90
\@glsxtr@noop@recordcounter 7, 9
\@glsxtr@notfoundinlist 143
\@glsxtr@op@recordcounter 9, 10
\@glsxtr@optlist 34, 35
\@glsxtr@org@GLS@ 41
\@glsxtr@org@GLSpl@ 41
\@glsxtr@org@Gls@ 41
\@glsxtr@org@Glspl@ 41
\@glsxtr@org@Glsxtrtitlefirst . 199, 201
\@glsxtr@org@Glsxtrtitlefirstplural
. . . . 200, 201
\@glsxtr@org@Glsxtrtitlefull . . 200, 201
\@glsxtr@org@Glsxtrtitlefullpl 200, 201
\@glsxtr@org@Glsxtrtitlelong . . 200, 201
\@glsxtr@org@Glsxtrtitlelongpl 200, 201
\@glsxtr@org@Glsxtrtitleplural 199, 201
\@glsxtr@org@Glsxtrtitleshort . 199, 200
\@glsxtr@org@Glsxtrtitleshortpl 199, 200
\@glsxtr@org@Glsxtrtitletext . . 199, 200
\@glsxtr@org@MakeUppercase 199, 200
\@glsxtr@org@checkfirsthyper 58, 88
\@glsxtr@org@delimN 38, 39
\@glsxtr@org@delimR 38, 39
\@glsxtr@org@doseeglossary 9, 10, 89
\@glsxtr@org@gloautosee 9, 10
\@glsxtr@org@gls@ 40, 41
\@glsxtr@org@glsignore 38, 39
\@glsxtr@org@glspl@ 41
\@glsxtr@org@glsxtrtitlefirst . 199, 201
\@glsxtr@org@glsxtrtitlefirstplural
. . . . 200, 201
\@glsxtr@org@glsxtrtitlefull . . 200, 201
\@glsxtr@org@glsxtrtitlefullpl 200, 201
\@glsxtr@org@glsxtrtitlelong . . 200, 201
\@glsxtr@org@glsxtrtitlelongpl 200, 201
\@glsxtr@org@glsxtrtitleplural 199, 201
\@glsxtr@org@glsxtrtitleshort . 199, 200
\@glsxtr@org@glsxtrtitleshortpl 199, 200
\@glsxtr@org@glsxtrtitletext . . 199, 200
\@glsxtr@org@makeglossaries 88
\@glsxtr@org@markboth 199
\@glsxtr@org@markright 198, 199
\@glsxtr@org@newacronymstyle 87
\@glsxtr@org@postdescription 139
\@glsxtr@org@see@noindex 104

\@glsxtr@org@setacronymstyle 87
\@glsxtr@orgprintglossary 35, 93
\@glsxtr@orgwarndep 144
\@glsxtr@p@acrlong@ 64
\@glsxtr@p@acrlongpl@ 64
\@glsxtr@p@acrshort@ 64
\@glsxtr@p@acrshortpl@ 64
\@glsxtr@p@long@ 64
\@glsxtr@p@longpl@ 64
\@glsxtr@p@plural@ 64
\@glsxtr@p@short@ 64
\@glsxtr@p@shortpl@ 64
\@glsxtr@p@text@ 64
\@glsxtr@pagestag 38
\@glsxtr@pagetag 38
\@glsxtr@prevunitcount 82
\@glsxtr@printglossopts 35, 91, 93
\@glsxtr@provide@addstoragekey 19
\@glsxtr@provide@storagekey 18
\@glsxtr@record 9, 10, 40–42, 44
\@glsxtr@recordsee 9
\@glsxtr@redef@forglsentries . . 6, 16, 17
\@glsxtr@redefstyles 15, 216
\@glsxtr@reg@glosslist 88–91, 93
\@glsxtr@s@longnewglossaryentry 23
\@glsxtr@savepreloctag 38, 39
\@glsxtr@setentrycountunsetattr 73
\@glsxtr@setentryunitcountunsetattr 85
\@glsxtr@setupshortcuts 13, 14, 16, 17
\@glsxtr@shortcutsval 13, 105
\@glsxtr@swaptwo 143
\@glsxtr@tag 138
\@glsxtr@taggingcs 138
\@glsxtr@theHvalue 6, 7, 44
\@glsxtr@thevalue 6, 7, 43, 44
\@glsxtr@thisloctag 38, 39
\@glsxtr@titlelabel 95, 96, 108
\@glsxtr@tmp 15, 98
\@glsxtr@type 126
\@glsxtr@unitcountlist 80
\@glsxtr@unsrt@getgrouptitle 107
\@glsxtr@usesee 30
\@glsxtr@warn@onexistsordo 6, 9, 10
\@glsxtr@warn@undefaction 6, 9, 10
\@glsxtrdocdeffalse 31
\@glsxtrentryfmt 19
\@glsxtrindexaliased 59
\@glsxtrindexcrossrefsfalse 11
\@glsxtrindexcrossrefstrue 11
\@glsxtrinmark 198, 199
\@glsxtrlong 64, 153
\@glsxtrlongpl 64, 157
\@glsxtrnotinmark 198, 199
\@glsxtrp 69, 70
\@glsxtrp@opt 67
\@glsxtrpl 34, 35
\@glsxtrpostloctag 37–39
\@glsxtrpreloctag 37–39
\@glsxtrsetaliasnoindex 59, 60
\@glsxtrshort 64, 152
\@glsxtrshortpl 64, 155
\@glsxtrundeftag 5, 17
\@gobble 6, 9, 11, 45, 144
\@gobbletwo 144
\@ifnextchar 61
\@ifpackageloaded . . . .
. . . 4, 12, 105, 110, 126, 128, 131, 132, 215
\@ifstar 18, 23, 24, 26, 32, 61, 106, 137
\@ifundefined 215
\@ignored@glossaries 25–27
\@input 104
\@input@ 99
\@istfilename 89
\@makeglossary 89
\@mfu@domakefirstuc 138
\@mfu@nocaplist 138
\@ne 75, 84
\@newglossaryentry@defcounters . . 74, 82
\@newglossaryentryposthook . . . 8, 9, 18, 56
\@newglossaryentryprehook . 8, 9, 18, 24, 56
\@nil 98, 109
\@nnil 133, 134, 136, 143, 144
\@no@glsxtrindexaliased 59, 60
\@no@makeglossaries 103
\@nopostdesc 92
\@onelevel@sanitize 8, 34, 95, 96, 108
\@onlypreamble . . . .
. . . . 35, 38, 84, 104, 106, 133, 135, 137
\@org@glossaryentrynumbers 92, 93
\@org@newglossaryentryprehook 24
\@print@unsrt@glossary 107
\@printgloss@setsort 91, 92
\@printglossary 35, 107
\@printunsrtglossary 106
\@sGlsXtrEnableOnTheFly 32
\@secondofthree 45–
47, 52–55, 57, 150, 151, 153, 154, 156, 157

\@secondoftwo 41, 44, 48–56, 58, 63, 88, 93, 143, 149, 150, 152–158, 170, 199, 200
\@sglsxtr@provide@storagekey 18
\@thirdofthree 45–47, 53–56, 58, 150, 151, 153, 155, 156, 158
\@thirdoftwo 48–52
\@warn@nomakeglossaries 100
\@xdy@main@language 99
\@xdylanguage 99, 100
\\ 98

\␣ 101, 102

A

\AB 12
\Ab 12
\ab 12
abbreviation styles:
  long-noshort 188
  long-postshort-user 193
  long-short-user 192
  short 173
  short-long-user 194
  short-postlong-user 196
\abbreviationsname 12
\abbrvpluralsuffix 105, 145, 146, 164, 166, 168, 170, 172, 173, 175, 177–183, 191, 192, 195, 197
\ABP 12
\Abp 12
\abp 12
\ACRfullfmt 86
\Acrfullfmt 86
\acrfullfmt 86
\ACRfullplfmt 87
\Acrfullplfmt 86
\acrfullplfmt 86
\acronymentry 86
\acronymfont 52–56, 66, 87
\acronymname 12
\acronymsort 86
\acronymtype 12, 86, 87
\acrpluralsuffix 86, 105
\actualchar 136
\addtolength 233
\advance 75, 84, 104
\AF 13
\Af 12
\af 12
\AFP 13
\Afp 12
\afp 12
\AL 12
\Al 12
\al 12
\ALP 12
\Alp 12
\alp 12
\AnyTrackedLanguages 215
\appto 8, 9, 15, 18, 23, 29, 56, 61, 74, 82, 108, 132, 142, 145
\AS 12
\As 12
\as 12
\ASP 12
\Asp 12
\asp 12
\AtBeginDocument 17, 36, 37, 105
\AtEndDocument 30, 75, 83, 99, 100

B

babel package 133, 134, 142
\begin 97, 101, 102, 107, 218–222
\begingroup 6, 60, 107
\bgroup 24, 92

C

\catcode 104
category attributes:
  discardperiod 141
  entrycount 72–75, 85
  firstuc 130
  glossdesc 126
  glossdescfont 127
  glossname 128
  glossnamefont 128, 131
  headuc 201
  indexname 133
  indexonlyfirst 60
  insertdots 145, 146
  nohyper 58
  nohyperfirst 45–47
  regular 39, 79, 163–168, 170, 172, 174, 175, 185, 186, 191, 197
\cGLS 12, 73, 85
\cGls 12, 73, 85
\cgls 12, 73, 85
\cGLSformat 77
\cGlsformat 77

\cglsformat 76, 78
\cGLSpl 12, 73, 85
\cGlspl 12, 73, 85
\cglspl 12, 73, 85
\cGLSplformat 77
\cGlsplformat 77
\cglsplformat 76, 79
\char 95
\columnwidth 36, 37
\count@ 75, 76, 84
\csappto 22
\csdef 18, 21–23, 56–58, 75, 80, 81, 83, 121, 161–163, 170, 192, 194, 196, 218
\cseappto 27
\csedef 21, 81
\csgdef 22, 25–27, 31, 38, 75, 80, 83, 84
\cslet 21, 24, 92
\csletcs 21, 22, 162, 163
\csname 5, 6, 18, 26, 36, 39, 42–44, 52–58, 60, 67, 81, 89, 96, 99, 100, 102, 103, 107, 109, 126, 144, 149–158, 163, 232
\cspreto 23
\csuse 19, 20, 23, 26, 38, 57, 69–71, 79–84, 92, 96, 97, 106, 108, 109, 121, 132, 140, 161, 163, 225–228
\csxdef 29, 81, 83, 96
\currentglossary 92
\CurrentOption 16, 217
\CurrentTrackedLanguageTag 105
\CurrentTrackedTag 216
\CustomAbbreviationFields 146, 164–168, 170, 171, 173, 174, 176, 184, 186, 188, 191–194, 196, 197

D

\DeclareAcronymList 86
\DeclareOption 4, 217
\DeclareOptionX 4, 16
\def . 6–10, 17, 24, 26, 30, 32–35, 37, 39–56, 60, 62–67, 76–79, 86, 89–99, 107, 108, 132–136, 138, 142–146, 149–158, 161, 233
\defglsentryfmt 25–27
\define@boolkey 11, 59
\define@choicekey 6, 9, 10, 13, 14, 42, 93
\define@key 8, 9, 15, 18, 43, 44, 56, 144
\DefineAcronymSynonyms 13, 14
\delimN 38, 39
\delimR 38, 39
\detokenize 32
\dimen@ 88, 225–232
\dimen@i 226–228
\dimen@ii 226–228
\dimexpr 36, 37, 224
\disable@keys 12, 17, 31
\do . . . 5, 15, 31, 74, 85, 89, 91, 96, 107, 126, 137
\do@gls@link@checkfirsthyper 19, 40–43, 52–56, 149–158
\do@glsdisablehyperinlist 43, 59
doc package 136
\dolistcsloop 20
\DTLifinlist 89–91, 93
\DTLifint 95

E

\eappto 7, 15, 25–27, 133, 217
\edef . 5, 6, 23, 25–28, 43, 44, 58, 60, 63, 79–81, 83, 88–90, 93, 95, 98–100, 104, 127, 128, 130, 131, 133, 134, 136, 144, 226–228
\eglssetwidest 225–232
\egroup 24, 93
\else 7, 8, 11–15, 22, 31, 32, 37, 39, 42, 43, 60, 76, 88, 91–93, 95, 97, 98, 101, 103, 104, 132–134, 136, 143–145, 152–158, 164–175, 191–193, 195, 197, 219–223, 233, 234
\emph 184
\empty 97, 98
\encapchar 136
\end 97, 101, 102, 107, 218–222
\end@glsxtr@display@loc 97
\endcsname 5, 6, 18, 26, 36, 39, 42–44, 52–58, 60, 67, 81, 89, 96, 99, 100, 102, 103, 107, 109, 126, 144, 149–158, 163, 232
\endgroup 7, 60, 107
entry categories:
  abbreviation 158
  general 121, 123
  index 124
\epreto 133
\equal 103
etoolbox package 4
\expandafter 16, 18, 19, 29, 30, 32–34, 56–58, 61, 62, 67, 78–80, 89–91, 93, 96, 98, 107, 109, 126, 129, 130, 133, 136, 143, 145, 146
\expandonce 86, 133, 134

F

\fi 6–8, 10–15, 22, 30–32, 36, 37, 39, 42, 43, 60, 61, 76, 84, 85, 88, 91–

93, 95, 97, 98, 100, 101, 103, 104, 106, 132–134, 136, 143, 145, 152–158, 164–175, 191–193, 195, 197, 219–223, 225–234
first use . . . 235
  flag . . . 235
  text . . . 235
\firstacronymfont . . . 87, 88
fontspec package . . . 105
\footnote . . . 168
\forallglossaries 30, 107, 124, 126, 225–231
\forallglsentries . . . 75, 84
\ForEachTrackedDialect . . . 216
\forglsentries . . . 5, 30, 124, 126, 225–231
\forlistcsloop . . . 20, 84, 97
\forlistloop . . . 94, 138
\futurelet . . . 142

G

\gdef . . . 38, 135
\Genacrfullformat . . . 86, 87
\genacrfullformat . . . 86, 87
\GenericAcronymFields . . . 86
\Genplacrfullformat . . . 86, 87
\genplacrfullformat . . . 86, 87
\glo@grabfirst . . . 109
\glo@name . . . 129, 130
\gloaliaslabel . . . 63
\global . . . 24, 93, 109
\glolinkprefix . . . 43, 63, 105, 108
glossaries package . . . 16, 217
glossaries-accsupp package . . . 14, 16, 110
glossaries-extra package . . . 2
glossaries-extra-stylemods package . 15, 139, 216
glossaries.sty package . . . 24
\GlossariesExtraWarning . . . 5, 11, 22, 23, 33, 35, 87, 90, 98, 101, 104, 107, 127, 128, 130, 131, 138, 163
\GlossariesExtraWarningNoLine . 11, 76, 84
\GlossariesWarning . . 38, 90, 92, 94, 95, 161
\GlossariesWarningNoLine . . . 90, 100
glossary styles:
  alttree . . . 223, 224, 232
  inline . . . 223
  listdotted . . . 218
  listdottedstyle . . . 218
  sublistdotted . . . 218
glossary-long package . . . 219
glossary-longbooktabs package . . . 219
glossary-mcols package . . . 217
glossary-tree package . . . 217
\glossaryentrynumbers . 39, 92, 93, 109, 110
\glossaryheader . . . 97, 107, 218–222, 233
\glossaryname . . . 92
\glossarypostamble . . . 97, 107, 108
\glossarypreamble . . . 97, 107
\glossarysection . . . 97, 102, 107, 108
\glossarytitle . . . 26, 92, 97, 102, 107
\glossarytoctitle . . . 26, 92, 97, 102, 107
\glossentry . . . 92, 109, 110, 218–222, 233
\glossentrydesc . . . 218–224
\glossentryname . . . 218–222, 233
\glossentrysymbol . . . 219–224
\glossxtrsetpopts . . . 139
\GLS . . . 73, 85
\Gls . . . 34, 73, 85
\gls . . . 22, 33, 35, 73, 85, 90, 101
\gls@assign@desc . . . 24
\gls@assign@field . . . 8, 9, 18, 56
\gls@checkseeallowed . . . 32, 89
\gls@codepage . . . 100
\gls@defdocnewglossaryentry . . . 74, 82
\gls@defglossaryentry . . . 24, 33, 34
\gls@dotoctitle . . . 92
\gls@level . . . 109
\gls@noidxglossary . . . 89
\gls@org@glossaryentryfield . . . 92
\gls@org@glossarysubentryfield . . . 92
\gls@save@numberlist . . . 37, 39
\gls@tmplen . . . 225–231, 233
\gls@type . . . 89
\glsabbrvdefaultfont . . . 148, 164, 166, 168, 170, 172, 173, 175
\glsabbrvemfont . . . 184–190
\glsabbrvfont . . 65, 66, 87, 148, 152, 153, 155, 156, 159, 164–168, 170–173, 175–197
\glsabbrvuserfont . . . 190–192, 195, 197
\GLSaccessdesc . . . 48
\Glsaccessdesc . . . 48, 127, 137
\glsaccessdesc . . . 48, 127, 141
\GLSaccessdescplural . . . 49
\Glsaccessdescplural . . . 49
\glsaccessdescplural . . . 48
\GLSaccessfirst . . . 46
\Glsaccessfirst . . . 46
\glsaccessfirst . . . 45
\GLSaccessfirstplural . . . 47
\Glsaccessfirstplural . . . 47
\glsaccessfirstplural . . . 47

\Glsaccesslong 54, 147, 154, 165, 172, 175, 192, 193
\glsaccesslong 54, 55, 147, 154, 155, 164, 166–169, 171–175, 191, 193, 195, 197
\Glsaccesslongpl 55, 147, 157, 165, 172, 175, 192, 193
\glsaccesslongpl 55, 56, 147, 157, 158, 164, 166, 167, 169, 171, 172, 174, 175, 191, 193, 195, 197
\GLSaccessname 48
\Glsaccessname 48
\glsaccessname 47
\GLSaccessplural 47
\Glsaccessplural 46
\glsaccessplural 46
\Glsaccessshort 52, 153, 159, 166, 169, 171, 174, 195, 197
\glsaccessshort . . . 52, 53, 147, 152, 153, 159, 164–166, 168–175, 191–193, 195, 197
\Glsaccessshortpl 53, 156, 159, 167, 169, 171, 174, 195, 197
\glsaccessshortpl 53, 54, 147, 155, 156, 159, 164–166, 169, 171–175, 192, 193, 195, 197
\GLSaccesssymbol 49
\Glsaccesssymbol 49, 137
\glsaccesssymbol 49, 137, 141
\GLSaccesssymbolplural 50
\Glsaccesssymbolplural 50
\glsaccesssymbolplural 49
\GLSaccesstext 45
\Glsaccesstext 45
\glsaccesstext 45
\glsacrshortcutstrue 13, 14
\glsacspacemax 88
\glsadd 31, 101
\glsaddstoragekey 23, 121
\glsbackslash 32
\glscapscase 41, 44–58, 149–160
\glscategory 40, 44, 58, 65, 66, 122, 123, 126–128, 130–132, 137, 140, 149–153, 155, 156
\glscategorylabel 58, 144–146, 170, 192, 194, 196
\glsclosebrace 102, 103
\glscurrententrylabel 37–39, 92, 99, 107, 132, 139, 140
\glscurrentfieldvalue 19, 20, 190
\glscustomtext . . 40, 41, 52–56, 149–158, 160
\glsdefaulttype 5, 11, 22, 23, 91, 92, 101, 107, 108
\glsdescriptionaccessdisplay 114, 115, 127
\glsdescriptionpluralaccessdisplay 115
\glsdescwidth 218, 220–222
\glsdetoklabel 20–22, 24, 28–32, 43, 44, 60, 63, 74, 75, 80–84, 89, 92, 94, 95, 109, 126, 129, 130, 226–228
\glsdisplaynumberlist 90, 93
\glsdohyperlink 64
\glsdohypertarget 93
\glsdoifexists 10, 21, 27, 29, 40, 41, 44, 52–56, 94, 95, 149–158
\glsdoifexistsordo 19, 42
\glsdoifexistsorwarn 11, 126, 127, 136, 137
\glsdoifnoexists 24
\glsdonohyperlink 43, 63
\glsdosanitizesort 91
\glsenableentrycount 73, 76, 84
\glsenableentryunitcount 75, 85
\glsentrycounter 98
\glsentrycurrcount 74, 75, 82
\Glsentrydesc 114, 119, 128
\glsentrydesc 114, 115, 119, 128
\Glsentrydescplural 115, 119
\glsentrydescplural 115, 119
\Glsentryfirst 79, 112, 118
\glsentryfirst 79, 112, 118, 212
\Glsentryfirstplural 79, 113, 118
\glsentryfirstplural 79, 112, 113, 118, 212, 213
\glsentryfmt 25–27
\Glsentryfull 87
\glsentryfull 87
\Glsentryfullpl 87
\glsentryfullpl 87
\glsentryitem 218–222, 233
\Glsentrylong 66, 67, 79, 117, 120
\glsentrylong 66, 67, 79, 116, 117, 120, 170, 194, 196, 213
\Glsentrylongpl 66, 67, 79, 117, 120
\glsentrylongpl 66, 67, 79, 117, 120, 213, 214
\Glsentryname 110, 117, 129–132
\glsentryname 110, 111, 117, 118, 133, 225–232
\glsentrynumberlist 90, 95, 230–232
\Glsentryplural 111, 118
\glsentryplural 111, 112, 118, 211
\glsentryprevcount 74, 76, 82
\glsentryprevmaxcount 82

\glsentryprevtotalcount 82
\Glsentryshort 65, 66, 116, 120
\glsentryshort 65, 66, 88, 115, 116, 119, 120, 192, 194, 209, 210
\Glsentryshortpl 65, 66, 116, 120
\glsentryshortpl 65, 66, 116, 120, 210
\Glsentrysymbol 113, 119
\glsentrysymbol 113, 119, 229, 230
\Glsentrysymbolplural 114, 119
\glsentrysymbolplural 114, 119
\Glsentrytext 111, 118
\glsentrytext 63, 111, 118, 211
\Glsentryuseri 50
\glsentryuseri 50
\Glsentryuserii 50
\glsentryuserii 50
\Glsentryuseriii 51
\glsentryuseriii 51
\Glsentryuseriv 51
\glsentryuseriv 51
\Glsentryuserv 51
\glsentryuserv 51
\Glsentryuservi 52
\glsentryuservi 52
\glsfieldfetch 63
\glsfieldxdef 126
\glsfindwidesttoplevelname 225
\GLSfirst 204, 205
\Glsfirst 205
\glsfirst 204, 205
\glsfirstabbrvdefaultfont 148, 164, 166, 168, 170, 172, 173, 175
\glsfirstabbrvemfont 185–190
\glsfirstabbrvfont 87, 147, 164–175, 177–183, 185–193, 195, 197
\glsfirstabbrvuserfont . 191, 192, 194–197
\glsfirstaccessdisplay 112
\glsfirstlongdefaultfont 164, 166, 172, 173, 175
\glsfirstlongemfont 185–189
\glsfirstlongfont 147, 164–168, 170, 172–176, 184–189, 191–197
\glsfirstlongfootnotefont 168–171
\glsfirstlonguserfont . . 191, 192, 195, 197
\GLSfirstplural 205, 206
\Glsfirstplural 206
\glsfirstplural 205
\glsfirstpluralaccessdisplay . . 112, 113
\glsforeachincategory 161
\glsgenentryfmt 40
\glsgetattribute 62, 76, 80–82, 99, 127, 128, 130, 131, 133
\glsgetcategoryattribute 122
\glsgetwidestname 224
\glsgroupheading 109, 218–222, 233
\glsgroupskip 109, 219–223, 234
\glshasattribute 62, 75, 76, 80, 81, 83, 84, 99, 127, 128, 130, 131, 133, 164–168, 170, 185, 186, 191, 192, 194, 196, 197
\glshascategoryattribute 122
\glshyperlink 63
\glshypernavsep 96
\glshypernumber 99, 132
\glsifattribute 42, 45, 59, 60, 69, 124, 127–130, 139, 141, 142, 201–209
\glsifcategory 124
\glsifcategoryattribute 58, 122, 123, 145, 146
\glsifnotregular 44
\glsifnotregularcategory 123
\glsifplural 41, 44, 46–50, 52–56, 141, 142, 149–159
\glsifregular 40, 44, 79
\glsifregularcategory 123
\glsifusetranslator 26
\glsignore 38, 39
\glsinlinedescformat 223
\glsinlinesubdescformat 223
\glsinsert 42, 44, 52–56, 149–160
\glskeylisttok 86, 145, 146
\glslabel 28, 40, 42, 43, 58–60, 62, 63, 88, 140, 141, 158–160, 170, 192, 194, 196
\glslabeltok 86, 145, 146, 164–168, 170, 172, 173, 175, 176, 184–186, 188, 191–194, 196, 197
\glsletentryfield 133
\glslink 86, 87
\glslink options
  format 132
  hyper 198
  noindex 6, 59, 198
  wrgloss 42
\glslinkcheckfirsthyperhook 58
\glslinkpostsetkeys 43
\glslinkvar 61, 62
\glslistdottedwidth 218
\glslocalunset 42
\glslongaccessdisplay 116, 117

\glslongdefaultfont . . . 148, 164, 166, 167, 172, 173, 175
\glslongemfont . . . 184–189
\glslongfont . . . 66, 148, 154, 155, 157, 158, 164, 166, 168, 170, 172, 173, 175, 185–189, 191, 192, 195, 197
\glslongfootnotefont . . . 167, 168, 170
\glslongpltok . . . 146, 164–168, 174, 176, 185, 186, 188, 191, 192, 194, 196, 197
\glslongpluralaccessdisplay . . . 117
\glslongtok 86, 145, 146, 164–168, 170, 172–174, 176, 184–186, 188, 191–194, 196, 197
\glslonguserfont . . . 191, 192, 194, 195, 197
\glsnameaccessdisplay . . . 110, 129, 131
\glsnamefont . . . 128–132
\glsnavhyperlink . . . 96
\glsnextpages . . . 92
\glsnoidxdisplayloc . . . 94
\glsnoidxdisplayloclisthandler . . . 94
\glsnoidxloclist . . . 95, 109, 110
\glsnoidxnumberlistloophandler . . . 94
\glsnonextpages . . . 92
\glsnonumberlistfalse . . . 37
\glsnonumberlisttrue . . . 37
\glsnumberlistloop . . . 90
\glsnumlistlastsep . . . 94
\glsnumlistsep . . . 94
\glsopenbrace . . . 102, 103
\glsorder . . . 89
\glspagelistwidth . . . 218, 220–222
\glspar . . . 108
\GLSpl . . . 73, 85
\Glspl . . . 34, 73, 85
\glspl . . . 34, 73, 85
\GLSplural . . . 204
\Glsplural . . . 204
\glsplural . . . 204
\glspluralaccessdisplay . . . 111, 112
\glspluralsuffix . . . 105, 145, 148
\glspostdescription . . . 139, 218–224
\glspostinline . . . 223
\glspostlinkhook . 40, 42, 52–56, 67, 149–158
\glsprestandardsort . . . 91
\glsresetentrylist . . . 97, 107
\glssee . . . 23
\glsseeformat . . . 30, 89, 94, 95
\glssetabbrvfmt . . . 40, 44, 65, 66, 126–128, 130, 131, 137, 149–153, 155, 156
\glssetattribute . . . 164–168, 170, 172, 173, 175, 176, 185, 186, 188, 191, 192, 194, 196, 197
\glssetcategoryattribute . . . 74, 85, 88, 122, 123, 125, 137
\glssetnoexpandfield . . . 8, 9
\glssettoctitle . . . 92
\glsshortaccessdisplay . . . 115, 116
\glsshortpltok . . . 146, 164–168, 170, 172, 173, 185, 186, 191, 192, 194, 196, 197
\glsshortpluralaccessdisplay . . . 116
\glsshorttok . 86, 145, 146, 164–168, 170–173, 176, 184–186, 188, 191–194, 196, 197
\glssubentryitem . . . 218–223, 233
\glssymbolaccessdisplay . . . 113
\glssymbolpluralaccessdisplay . . . 114
\glstarget . . . 218–223, 233
\GLStext . . . 203
\Glstext . . . 203
\glstext . . . 203
\glstextaccessdisplay . . . 111
\glstextformat . . . 42, 43
\glstextup . . . 176
\glstreeindent . . . 232, 233
\glstreenamebox . . . 233
\glstreenamefmt . . . 224–233
\GlstrLetField . . . 21
\glstype . . . 42, 43, 52–56, 149–158
\glsunset . . . 31, 42, 76, 77
\glswrite . . . 88
\glswriteentry . . . 7
\Glsxtr . . . 35
\glsxtr . . . 35
\glsxtr@@do@wrglossary . . . 7, 9, 10
\glsxtr@addloclistfield . . . 9, 10
\glsxtr@addunused . . . 30
\glsxtr@applyabbrvfmt . . . 158
\glsxtr@applyabbrvstyle . . . 144, 145, 161
\glsxtr@counterrecord . . . 106
\glsxtr@dooption . . . 4, 11, 16
\glsxtr@fields . . . 105
\glsxtr@headentry@p . . . 69, 70
\glsxtr@ifnextpunc . . . 142
\glsxtr@ifpunctoken . . . 142
\glsxtr@indexonly@saveentrycounter . . . 9, 10, 17
\glsxtr@keylist . . . 33, 34
\glsxtr@langtag . . . 105
\glsxtr@linkprefix . . . 105

\glsxtr@makeglossaries 89
\glsxtr@newabbreviation 87, 145
\glsxtr@next 143
\glsxtr@org@getgrouptitle 96
\glsxtr@org@newignoredglossary 24
\glsxtr@orgmakenoidxglossaries 31
\glsxtr@pluralsuffixes 105
\glsxtr@provideignoredglossary 26
\glsxtr@punclist 142, 143
\glsxtr@record 7
\glsxtr@recordsee 8
\glsxtr@resource 104
\glsxtr@s@newignoredglossary 24
\glsxtr@s@provideignoredglossary . . . 26
\glsxtr@saveentrycounter 7, 8, 60
\glsxtr@setup@record 9, 10, 16, 17
\glsxtr@shortcutsval 105
\glsxtr@texencoding 105
\glsxtr@usesee 29
\glsxtr@warnonexistsordo . . 6, 9, 10, 28, 29
\glsxtr@writefields 104
\glsxtrabbrvfootnote 168–170
\glsxtrabbrvpluralsuffix 105, 149, 164, 166, 168, 170, 172, 173, 175, 176, 180, 191
\glsxtrabbrvtype 12, 146
\glsxtraddallcrossrefs 30
\glsxtralias 60
\glsxtrAltTreeIndent 224
\glsxtralttreeInit 232
\glsxtrAltTreePar 223
\glsxtrAltTreeSetHangIndent . . . 224, 233
\glsxtrAltTreeSetSubHangIndent 233
\glsxtralttreeSubSymbolDescLocation 233
\glsxtralttreeSymbolDescLocation 224, 233
\glsxtrassignfieldfont 45–52
\glsxtrcat 33, 34
\glsxtrchecknohyperfirst 45–47
\glsxtrComputeTreeIndent 233
\glsxtrComputeTreeSubIndent 233
\GlsXtrDefineAbbreviationShortcuts . 14
\GlsXtrDefineOtherShortcuts 14
\glsxtrdiscardperiod 140
\glsxtrdisplayendloc 97
\glsxtrdisplayendlochook 98
\glsxtrdisplaysingleloc 97, 98
\glsxtrdisplaystartloc 97
\glsxtrdoautoindexname 60, 61, 132
\glsxtrdopostpunc 170
\glsxtrdowrglossaryhook 61
\GlsXtrEnableEntryCounting 85
\GlsXtrEnableEntryUnitCounting 73
\GlsXtrEnableOnTheFly 33, 35
\glsxtrfieldlistgadd 106
\glsxtrfieldtitlecase 127–130
\glsxtrfieldtitlecasecs 126
\glsxtrfieldxifinlist 108
\glsxtrfirstscfont 177–180
\glsxtrfirstsmfont 180–183
\GlsXtrFmtDefaultOptions 19
\GlsXtrFmtField 19
\GlsXtrFormatLocationList 37, 39, 230–232
\GLSxtrfull 13, 208
\Glsxtrfull 12, 209
\glsxtrfull 12, 208
\Glsxtrfullformat . . . 147, 160, 162, 165, 166, 169, 171, 173–175, 192, 193, 195, 197
\glsxtrfullformat . . 147, 160, 162, 164–168, 170, 172, 174, 175, 191, 193, 195, 197
\GLSxtrfullpl 13, 208, 209
\Glsxtrfullpl 12, 209
\glsxtrfullpl 12, 208
\Glsxtrfullplformat . 148, 160, 162, 165, 167, 169, 171, 173–175, 192, 193, 195, 197
\glsxtrfullplformat . 160, 162, 164, 166, 169, 171, 172, 174, 175, 191, 193, 195, 197
\glsxtrfullsep 147, 164–167, 169, 171–175, 184–186, 190
\glsxtrgenabbrvfmt 40
\glsxtrgetgrouptitle 96
\Glsxtrheadfirst 200
\glsxtrheadfirst 200
\Glsxtrheadfirstplural 200
\glsxtrheadfirstplural 200
\Glsxtrheadfull 200
\glsxtrheadfull 200
\Glsxtrheadfullpl 200
\glsxtrheadfullpl 200
\Glsxtrheadlong 200
\glsxtrheadlong 200
\Glsxtrheadlongpl 200
\glsxtrheadlongpl 200
\Glsxtrheadplural 200
\glsxtrheadplural 200
\Glsxtrheadshort 200
\glsxtrheadshort 200
\Glsxtrheadshortpl 200
\glsxtrheadshortpl 200

\Glsxtrheadtext 200
\glsxtrheadtext 200
\glsxtrifcounttrigger 76, 77
\glsxtrifemptyglossary 97, 102, 107
\glsxtrifindexing 60
\glsxtrifinmark 69–72, 199, 200
\glsxtrifnextpunc 142, 143
\glsxtrifperiod 141, 142
\glsxtrifwasfirstuse . . . 44–47, 52–56, 58, 88, 141, 149, 152–158, 170, 192, 194, 196
\glsxtrindexaliased 59, 60
\glsxtrinitwrgloss 43
\glsxtrinitwrglossbeforefalse 42
\glsxtrinitwrglossbeforetrue 42
\Glsxtrinlinefullformat 148, 150, 162, 169, 171, 172, 174, 175, 193, 195, 214
\glsxtrinlinefullformat 148–150, 162, 169, 171–173, 175, 193, 195, 214
\Glsxtrinlinefullplformat . . 148, 151, 162, 169, 171, 172, 174, 175, 193, 195, 215
\glsxtrinlinefullplformat . . 147, 148, 151, 162, 169, 171–173, 175, 193, 195, 215
\glsxtrinsertinsidefalse 164
\glsxtrlocationhyperlink 98
\glsxtrlocrangefmt 98
\GLSxtrlong 12, 206, 207
\Glsxtrlong 12, 207
\glsxtrlong 12, 206
\GLSxtrlongpl 12, 206, 207
\Glsxtrlongpl 12, 207
\glsxtrlongpl 12, 206, 207
\glsxtrlongshortdescsort 165
\glsxtrmarkhook 198, 199
\glsxtrnewabbrevpresetkeyhook 146
\glsxtrnewnumber 13
\glsxtrnewsymbol 13
\glsxtrNoGlossaryWarning 14, 15, 99
\GlsXtrNoGlsWarningAutoMake 103
\GlsXtrNoGlsWarningBuildInfo 103
\GlsXtrNoGlsWarningCheckFile 103
\GlsXtrNoGlsWarningEmptyMain 103
\GlsXtrNoGlsWarningEmptyNotMain . . . 103
\GlsXtrNoGlsWarningEmptyStart 103
\GlsXtrNoGlsWarningHead 102
\GlsXtrNoGlsWarningMisMatch 103
\GlsXtrNoGlsWarningNoOut 103
\GlsXtrNoGlsWarningTail 103
\glsxtrorg@ifKV@glslink@hyper 40
\GLSxtrp 69
\Glsxtrp 68
\glsxtrp 68, 70
\Glsxtrpl 35
\glsxtrpl 35
\glsxtrpostdescription 125, 139, 223
\glsxtrpostlink 140
\glsxtrpostlinkendsentence 140
\glsxtrpostlinkhook 140
\glsxtrpostlocalreset 73, 75, 83
\glsxtrpostlocalunset 73, 74, 82, 83
\glsxtrpostlongdescription 24
\glsxtrpostnamehook 129–132
\GlsXtrPostNewAbbreviation 146, 162, 164–168, 170, 172, 173, 175, 176, 185, 186, 188, 191, 192, 194, 196, 197
\glsxtrpostreset 73, 75, 83
\glsxtrpostunset 72, 74, 82
\glsxtrprotectlinks 62, 63
\GlsXtrRecordCounter 7
\glsxtrregularfont 40, 44, 45
\glsxtrresourcecount 104
\glsxtrresourcefile 104
\glsxtrrestoremarkhook 198, 199
\glsxtrscfont 176–180
\glsxtrscsuffix 177–180
\GlsXtrSetActualChar 136
\glsxtrsetaliasnoindex 9, 10, 59, 60
\GlsXtrSetEncapChar 136
\GlsXtrSetEscChar 136
\glsxtrsetfieldifexists 21, 22
\GlsXtrSetLevelChar 136
\glsxtrsetpopts 67
\glsxtrsetupfulldefs 149–151, 170
\GLSxtrshort 12, 71, 72, 201, 202
\Glsxtrshort 12, 202
\glsxtrshort 12, 201
\GLSxtrshortpl 12, 201, 202
\Glsxtrshortpl 12, 202, 203
\glsxtrshortpl 12, 202
\glsxtrsmfont 180–183
\glsxtrsmsuffix 180–183
\glsxtrsupplocationurl 99
\glsxtrtagfont 139
\Glsxtrtitlefirst 199–201, 212
\glsxtrtitlefirst 199–201, 212
\Glsxtrtitlefirstplural 200, 201, 213
\glsxtrtitlefirstplural 200, 201, 212
\Glsxtrtitlefull 200, 201, 214
\glsxtrtitlefull 200, 201, 214

\Glsxtrtitlefullpl 200, 201, 215
\glsxtrtitlefullpl 200, 201, 215
\Glsxtrtitlelong 200, 201, 213
\glsxtrtitlelong 200, 201, 213
\Glsxtrtitlelongpl 200, 201, 214
\glsxtrtitlelongpl 200, 201, 213, 214
\Glsxtrtitleplural 199–201, 211, 212
\glsxtrtitleplural 199–201, 211
\Glsxtrtitleshort 199, 200, 210
\glsxtrtitleshort 199, 200, 209
\Glsxtrtitleshortpl 199, 200, 210
\glsxtrtitleshortpl 199, 200, 210
\Glsxtrtitletext 199, 200, 211
\glsxtrtitletext 199, 200, 211
\glsxtrtreetopindent 224, 232
\glsxtrundefaction . . . 6, 9, 10, 17, 25, 27–29
\glsxtrundeftag 17
\glsxtrunsrtdo 108
\GlsXtrUseAbbrStyleFmts 165, 167, 176–190, 194, 196, 198
\GlsXtrUseAbbrStyleSetup 177–189, 196, 198
\glsxtruserfield 190
\glsxtruserparen 191–197
\glsxtrusersuffix 191, 192, 195, 197
\glsxtruseseeformat 30
\GlsXtrWarnDeprecatedAbbrStyle 144, 163
\GlsXtrWarning 33, 34

H

\hangindent 224, 232
\hbox 218
\hfill 218
\href 62
\hsize 36, 37
\hss 218
\hyperlink 63, 99
\hyperpage 132
\hyperref 62, 99
hyperref package 63, 132, 198, 209

I

\if 32
\if@glsxtr@format@override 133
\if@glsxtrdocdefrestricted 31
\if@glsxtrindexcrossrefs 11, 30
\ifblank 15, 18, 33, 34, 88
\ifcase 6, 9, 13, 14, 31, 42, 93
\ifcsdef 6, 17, 22, 23, 25–27, 56, 57, 68–72, 80, 92, 96, 108, 127, 128, 130, 131, 140, 144, 158, 161, 218–222
\ifcsstring 17, 122, 161
\ifcsundef 22, 23, 25–27, 31, 35, 38, 63, 74, 80–83, 95, 96, 99, 108, 122, 160, 162, 163, 217, 225, 232
\ifcsvoid 23, 121
\ifdef 9, 13, 19, 28, 29, 36, 37, 58, 59, 69–71, 94, 95, 105, 124, 125, 136, 139, 190, 209–215, 218, 223
\ifdefempty 6, 7, 25–27, 29, 44, 74, 85, 86, 89, 91, 98, 107, 109, 137, 158
\ifdefequal 103, 109
\ifdefstring 5, 133, 138
\ifdefvoid 23, 29, 30, 63, 79, 99, 109
\ifdim 36, 37, 88, 225–232
\IfFileExists 15, 99, 103–105, 217
\ifglossaryexists 29
\ifglsacronym 12, 103
\ifglsacrshortcuts 13
\ifglsautomake 91, 103, 105
\ifglsentrycounter 22
\ifglsentryexists 28, 33, 34, 37, 44, 122, 139, 140
\ifglsfieldeq 121
\ifglshasfield 19, 59, 190
\ifglshaslong 79
\ifglshasparent 109, 225–228
\ifglshasshort 40, 44
\ifglshassymbol 141, 224
\ifglsindexonlyfirst 60
\ifglsnogroupskip 219–223, 234
\ifglsnonumberlist 39
\ifglssanitizesort 91
\ifglssubentrycounter 22
\ifglsused 30, 31, 58, 60, 75, 84, 88, 158, 225, 226, 228, 230, 231
\ifglsxindy 99, 101
\ifglsxtrinitwrglossbefore 42, 43
\ifglsxtrinsertinside 152–158, 164–175, 191–193, 195, 197
\ifHy@hyperindex 132
\ifinlistcs 20, 31
\ifKV@glslink@hyper 40, 43
\ifKV@glslink@local 42
\ifKV@glslink@noindex 7, 8, 60
\ifnum 10, 11, 76, 84, 95, 104, 233
\ifthenelse 103
\IfTrackedLanguageFileExists 216
\ifundef 63, 88, 138, 139

\ifx 36, 37, 92, 97, 98, 106, 133, 134, 136, 143, 144, 234
\immediate 75, 84, 99, 100, 105
\index 133
\indexspace 234
\input 215
\inputencodingname 105
\istfilename 89
\item 101, 102, 218

J

\jobname 99, 101–105

K

\key@ifundefined 8, 9, 18, 56, 107, 109
\KV@glslink@hyperfalse . . 45, 58, 59, 63, 64
\KV@glslink@noindexfalse 59
\KV@glslink@noindextrue 59, 64

L

\LaTeX 101, 102
\leaders 218
\leavevmode 24, 43
\let 4, 6, 7, 9–13, 16, 17, 19, 24, 31, 32, 35, 36, 38–64, 67, 73–75, 82–96, 104–107, 109, 127–134, 138, 139, 142–145, 149–158, 170, 198–201, 223, 225
\letabbreviationstyle 169, 171, 173, 174, 176, 178, 182, 187
\letcs . 18, 29, 30, 56, 94–96, 108, 109, 127–131
\levelchar 136
\listadd 80
\listbreak 138
\listcsadd 20
\listcseadd 20, 81
\listcsgadd 20, 31
\listcsxadd 20, 80
\loadglsentries 32, 101
\long 24

M

\MakeAcronymsAbbreviations 87
\makeatletter 99, 104, 135
\makeatother 135
\makebox 218, 233
\makefirstuc 138
makeglossaries 93
\makeglossaries 88, 100–103, 106
\makeglossary 89
makeindex **235**
makeindex 88
\makenoidxglossaries 101
\MakeTextUppercase 200
\MakeUppercase 199, 200
\markboth 199
\markright 199
\maxdimen 36, 37
\mbox 233
\medskip 103, 108
\MessageBreak 32, 35, 76, 85, 91, 92, 161
mfirstuc package 138
\mfirstucMakeUppercase . 45–56, 58, 65–67, 69, 71, 72, 78, 79, 86, 87, 111–120, 129, 130, 150, 151, 153, 155, 156, 158–160
\mfu@checkword@arg 138
\mfu@checkword@do 138

N

\NeedsTeXFormat 4, 217
\new@glossaryentry 32, 91
\new@ifnextchar 57, 78, 142, 149–158
\newabbr 13
\newabbreviation 13
\newabbreviationhook 146
\newabbreviationstyle 164–169, 171, 173, 174, 176–189, 191–194, 196, 198
\newacronym 86, 87
\newacronymhook 86
\newacronymstyle 87
\newcommand . 4–8, 10–35, 37–40, 42, 44, 45, 56, 57, 59–62, 64, 67, 69–74, 76, 78–82, 84, 85, 87, 88, 92–102, 104–126, 132–158, 160–163, 165, 167, 168, 176, 180, 184, 190, 191, 199–215, 217, 223–225, 232
\newcount 10, 104
\newentry 13
\newglossary 12, 89
\newglossaryentry 13, 31, 32, 74, 82, 86, 124, 125, 146
\newglossaryentry options
  alias 23
  desc 114, 119
  descplural 115, 119
  first 62, 112, 118, 163, 204–206, 212, 235
  firstplural . . 112, 113, 118, 163, 205, 212, 235
  loclist 20
  long 117, 120, 213
  longplural 117, 120, 213
  name 110, 117, 118, 133

plural 111, 112, 118, 163, 203, 204, 211
see 11, 29, 32, 89
short 116, 120, 144
shortplural 116, 120, 144
symbol 113, 119
symbolplural 113, 114, 119
text 62, 111, 118, 163, 165, 203, 210
\newif 42, 132, 164
\newlength 224
\newnum 13
\newrobustcmd 19, 21, 22, 57, 58, 67–69, 78, 96, 108, 138, 139, 149–158, 199, 201–209, 225–231
\newsym 13
\newterm 124
\newtoks 144
\newwrite 88
\NoCaseChange 69–72, 201–209
\noexpand 7, 15, 23, 86, 99, 100, 104, 133, 134, 146, 217
\nofiles 102
\noindent 103
\nopostdesc 24, 33, 34, 92, 125
\nr 6, 9, 10, 13, 14, 42, 93
\ns@GLSxtrfull 150
\ns@Glsxtrfull 149
\ns@glsxtrfull 149
\ns@GLSxtrfullpl 151
\ns@Glsxtrfullpl 151
\ns@glsxtrfullpl 150
\ns@GLSxtrlong 154
\ns@Glsxtrlong 154
\ns@glsxtrlong 153
\ns@GLSxtrlongpl 158
\ns@Glsxtrlongpl 157
\ns@glsxtrlongpl 157
\ns@GLSxtrshort 153
\ns@Glsxtrshort 152
\ns@glsxtrshort 152
\ns@GLSxtrshortpl 156
\ns@Glsxtrshortpl 155, 156
\ns@glsxtrshortpl 155
\null 15
\number 81–83, 104
\numexpr 81, 83

O

\or 6, 9, 10, 13, 14, 32, 42
\org@glossaryentrynumbers 37, 92
\org@glossarytitle 92
\org@ifKV@glslink@hyper 43

P

\p@gls@hyp@opt 61
package options:
abbreviations 11, 12
accsupp 14, 110
acronym 12
automake 91, 101, 105
true 106
docdef 10, 31, 74, 82
false 31
restricted 11
true 32
nonumberlist 37
nopostdot
false 11
numbers 13
record 6, 9, 40, 103, 244
shortcuts 13
all 13
false 13
none 13
true 13
style 15
stylemods 15
symbols 13, 125
undefaction 28
error 5
warn 5
\PackageError 5, 7, 15, 31, 35, 56–58, 60, 68, 73–75, 82, 84, 85, 87, 89–91, 97, 106, 108, 161–163, 217
\PackageWarning 11
\PackageWarningNoLine 11
\pageref 22
\par 102, 103, 223, 224, 233
\parindent 224, 233
\PassOptionsToPackage 4
\preglossarypreamble 23
\preto 59
\print@noop@unsrtglossaryunit 8, 9
\print@op@unsrtglossaryunit 9, 10
\printabbreviations 12
\printglossaries 90, 102
\printglossary 12, 90, 102
\printglossary options
nonumberlist 39

type 91
\printnoidxglossaries 102
\printnoidxglossary 90, 102
\printnumbers 13, 125
\printsymbols 13, 125
\printunsrtglossary 107
\printunsrtglossaryhandler 107, 108
\printunsrtglossaryunit 9, 10, 108
\printunsrtglossaryunitsetup 108
\ProcessOptions 217
\ProcessOptionsX 16
\protect 69–72, 118–120, 147, 164–186, 188, 191–194, 196, 197, 201–209
\protected@csedef 22, 224
\protected@csxdef 22, 224
\protected@edef 36, 86, 95, 96, 108, 133, 146
\protected@write 7, 8, 39, 88, 89, 104–106
\protected@xdef 109
\providecommand 12, 18, 39, 58, 59, 75, 84, 88, 89, 99, 100, 217
\ProvidesFile 215
\ProvidesPackage 4, 217

Q

\quotechar 136

R

\raggedright 220, 222
\relax 6, 7, 9, 10, 13, 14, 16, 19, 32, 35–38, 42, 61, 67, 75, 76, 84, 89, 90, 92–95, 97, 104–106, 109, 134, 136, 138, 140, 141, 144, 145, 225–234
relsize package 180
\renewcommand 5, 9–16, 23–29, 31, 32, 35, 37–42, 56, 58–60, 62, 63, 67, 72–75, 79, 82–93, 95–97, 99, 100, 108, 124, 126–129, 131, 136–140, 148, 162, 164–199, 218–223, 233, 234
\renewenvironment 218–222, 232
\renewglossarystyle 218–222, 232
\renewrobustcmd 44, 63
\RequireGlossariesExtraLang 216
\RequirePackage 4, 14–16, 217
\reserved@a 142
\reserved@b 142
\reserved@d 143
\RestoreAcronyms 87
\romannumeral 224, 225, 232

S

\s@gls@hyp@opt 61
\s@glsxtr@enabletagging 137
\s@printunsrtglossary 106, 108
\seename 30
\setabbreviationstyle 87, 165, 173
\setacronymstyle 87, 88
\setentrycounter 97
\SetGenericNewAcronym 87
\setglossarystyle 16, 92, 218, 234
\setkeys 6, 16, 43, 44, 60, 86, 92, 145, 146
\setlength 36, 37, 224, 233
\settowidth 88, 224–232
\setupglossaries 4, 16
\sfcode 140, 141, 223
\space 5, 7, 33, 35, 60, 73–76, 82, 84, 85, 87–90, 92, 100, 103, 106, 108, 141, 147, 165, 223, 224, 232
\spacefactor 140, 141, 145, 223
\string 5, 7, 8, 32, 33, 35, 39, 56, 57, 60, 68, 69, 73–76, 82, 84, 85, 87–92, 99–106, 108, 128, 130–133
\strut 218–223
\subglossentry 92, 109, 218–223, 233

T

\tablehead 221, 222
\tabletail 221, 222
\tabularnewline 218–223
\TeX 101
\texorpdfstring 19, 69–71, 209–215
textcase package 198
\textsc 176
\textsmaller 180
\texttt 100–103
\the 86, 98, 104, 136, 146, 164–168, 170–176, 184–186, 188, 191–194, 196, 197
\theglsentrycounter 7, 44
\theHglsentrycounter 7, 44
\theindex 132
\this@dialect 216
\toks@ 98, 136
tracklang package 105

U

\undef 9, 10, 137
\underline 139
\unskip 24, 31, 218
\usepackage 102, 103

V

\val 6, 9, 10, 13, 14, 42, 93

W

\warn@nomakeglossaries 90
\warn@noprintglossary 90, 93
\write 75, 84, 89, 99, 100, 105

X

\x 98
\xcapitaliseword s 126
\xdef 92
\xifinlist 80
\xifinlistcs 21
xindy **235**
xindy 88
xkeyval package 4
\XKV@checkchoice 39
\XKV@plfalse 39
\XKV@resa 39
\XKV@sttrue 39