glossaries-extra.sty v1.01: documented code

Nicola L.C. Talbot

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

2016-02-02

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

1.1 Package Initialisation and Options

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra}[2016/02/02 v1.01 (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.

```
\define@choicekey{glossaries-extra.sty}{undefaction}[\val\nr]%
 {warn,error}%
 {%
   \ifcase\nr\relax
     \renewcommand*{\glsxtrundefaction}[2]{%
       \@glsxtrundeftag\GlossariesExtraWarning{##1}%
     }%
     \renewcommand*{\glsxtr@warnonexistsordo}[1]{%
      \GlossariesExtraWarning{glossaries-extra}{%
      \string##1\space hasn't been defined, so
      some errors won't be converted to warnings.
      (This most likely means your version of
      glossaries.sty is below version 4.19.)}%
     }%
   \or
     \renewcommand*{\glsxtrundefaction}[2]{%
      \@glsxtrundeftag\PackageError{glossaries-extra}{##1}{##2}%
     }%
     \renewcommand*{\glsxtr@warnonexistsordo}[1]{}%
   \fi
 }
```

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@boolkey{glossaries-extra.sty}[@glsxtr]{docdef}[true]{}
```

indexcrossrefs Automatically index cross references at the end of the document

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

Switch off since this can increase the build time.

```
\@glsxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

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

glsxtrabbrvtype Glossary type for abbreviations.

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

bbreviationsdef Set by abbreviations option.

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

bbreviationsdef

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

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

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

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

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

iationShortcuts Enable shortcut commands for the abbreviations. Unlike the analogous command provided by glossaries, this uses \newcommand instead of \let as a safety feature.

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

Disable this command after it's been used.

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

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

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

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

@setupshortcuts Command used to set the shortcuts option.

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

Provide shortcuts option. Unlike the glossaries version, this is a choice rather than a boolean key but it also provides shortcuts=true and shortcuts=false, which are equivalent to shortcuts=all and shortcuts=none. Multiple use of this option in the *same* option list will override each other.

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

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
 }
```

Pass all other options to glossaries.

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

Process options.

```
\ProcessOptionsX
```

Load glossaries if not already loaded.

```
\RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
\@glsxtr@doaccsupp
```

Define abbreviations glossaries if required.

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

Setup shortcuts if required.

```
\@glsxtr@setupshortcuts
```

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

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

Now define the user command:

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

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

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

1.2 Extra Utilities

rifemptyglossary

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

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

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

1.3 Modifications to Commands Provided by glossaries

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

1.3.1 Existence Checks

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

```
\renewcommand{\glsdoifexists}[2]{%
  \ifglsentryexists{#1}{#2}%
  {%
    \glsxtrundefaction{Glossary entry `\glsdetoklabel{#1}'
    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}}
```

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 and disables the docdef key.

```
\let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
\renewcommand{\makenoidxglossaries}{%
  \glsxtr@orgmakenoidxglossaries
  \@glsxtrdocdeffalse
  \disable@keys{glossaries-extra.sty}{docdef}%
}
```

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

```
\renewcommand*{\gls@defdocnewglossaryentry}{%
  \if@glsxtrdocdef
```

Since the see value is now saved in a field, it can be used by entries that have been defined in the document.

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

```
    \let\newglossaryentry\new@glossaryentry
  \else
    \renewcommand*{\newglossaryentry}[2]{%
      \PackageError{glossaries-extra}{Glossary entries must
       be \MessageBreak defined in the preamble with \MessageBreak
       package option 'docdef=false'}{Move your glossary definitions to
       the preamble. You can also put them in a \MessageBreak separate file
       and load them with \string\loadglsentries.}%
    }%
  \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

```
371 \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

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

\glsxtr

```
373  \newcommand*{\glsxtr}[1][]{%
374   \def\glsxtr@keylist{##1}%
375   \@glsxtr
376  }
```

\@glsxtr

```
377  \newcommand*{\@glsxtr}[2][]{%
378   \ifglsentryexists{##2}%
379   {%
380     \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
381   }%
382   {%
383     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
384       description={\nopostdesc},##1}%
385   }%
386   \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
387  }
```

\Glsxtr

```
388  \newcommand*{\Glsxtr}[1][]{%
389   \def\glsxtr@keylist{##1}%
390   \@Glsxtr
391  }
```

\@Glsxtr

```
392  \newcommand*{\@Glsxtr}[2][]{%
393   \ifglsentryexists{##2}%
394   {%
395     \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
396   }%
397   {%
398     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
399       description={\nopostdesc},##1}%
400   }%
401   \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
402  }
```

\glsxtrpl

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

\@glsxtrpl

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

\Glsxtrpl

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

\@Glsxtrpl

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

\GlsXtrWarning

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

Disable commands after the glossary:

```
\let\@glsxtr@orgprintglossary\@printglossary
```

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

abledflycommand

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

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

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

1.3.3 Existing Glossary Style Modifications

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]{%
      \GlsXtrFormatLocationList{#1}\gls@save@numberlist{#1}}%
  \else
    \glsnonumberlisttrue
    \renewcommand*{\glossaryentrynumbers}[1]{\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}
```

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{%
      \GlsXtrFormatLocationList{##1}%
      \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}%
  {\glsgenentryfmt}%
  {\ifglshasshort{\glslabel}{\glsxtrgenabbrvfmt}{\glsgenentryfmt}}%
}
```

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

@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.

`\@GLStext@` All uppercase version of `\glstext`.

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

`\@Glstext@` First letter uppercase version.

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

`\@glsfirst@` No case changing version.

```
\def\@glsfirst@#1#2[#3]{%
  \@gls@field@link[\let\glsxtrifwasfirstuse\@firstoftwo]{#1}{#2}%
    {\glsentryfirst{#2}#3}%
}
```

`\@Glsfirst@` First letter uppercase version.

```
\def\@Glsfirst@#1#2[#3]{%
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glscapscase\@secondofthree
  ]%
   {#1}{#2}{\Glsentryfirst{#2}#3}%
}
```

`\@GLSfirst@` All uppercase version.

```
\def\@GLSfirst@#1#2[#3]{%
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glscapscase\@thirdofthree
  ]%
   {#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
}
```

`\@glsplural` No case changing version.

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

\@Glsplural First letter uppercase version.

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

\@GLSplural All uppercase version.

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

glsfirstplural@ No case changing version.

```
\def\@glsfirstplural@#1#2[#3]{%
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
  ]%
   {#1}{#2}{\glsentryfirstplural{#2}#3}%
}
```

Glsfirstplural@ First letter uppercase version.

```
\def\@glsfirstplural@#1#2[#3]{%
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
   \let\glscapscase\@secondofthree
  ]%
   {#1}{#2}{\Glsentryfirstplural{#2}#3}%
}
```

Glsfirstplural@ All uppercase version.

```
\def\@glsfirstplural@#1#2[#3]{%
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
   \let\glscapscase\@thirdofthree
  ]%
   {#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
}
```

\@Glsname@ First letter uppercase version.

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

`\@GLSname@` All uppercase version.

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

`\@Glsdesc@` First letter uppercase version.

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

`\@GLSdesc@` All uppercase version.

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

`@glsdescplural@` No case-changing version.

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

`@Glsdescplural@` First letter uppercase version.

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

`@GLSdescplural@` All uppercase version.

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

\@Glssymbol@ First letter uppercase version.

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

\@GLSsymbol@ All uppercase version.

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

lssymbolplural@ No case-changing version.

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

lssymbolplural@ First letter uppercase version.

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

LSsymbolplural@ All uppercase version.

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

\@Glsuseri@ First letter uppercase version.

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

\@GLSuseri@ All uppercase version.

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

\@Glsuserii@ First letter uppercase version.

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

\@GLSuserii@ All uppercase version.

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

\@Glsuseriii@ First letter uppercase version.

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

\@GLSuseriii@ All uppercase version.

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

\@Glsuseriv@ First letter uppercase version.

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

\@GLSuseriv@ All uppercase version.

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

\@Glsuserv@ First letter uppercase version.

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

\@GLSuserv@ All uppercase version.

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

\@Glsuservi@ First letter uppercase version.

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

\@GLSuservi@ All uppercase version.

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

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

\@acrshort No case change.

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

\@Acrshort First letter uppercase.

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

\@ACRshort All uppercase.

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

\@acrshortpl No case change.

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

\@Acrshortpl First letter uppercase.

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

```
}
```

\@ACRshortpl All uppercase.

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

\@acrlong No case change.

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

\@Acrlong First letter uppercase.

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

```
  }%
  \glspostlinkhook
}
```

`\@ACRlong` All uppercase.

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

`\@acrlongpl` No case change.

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

`\@Acrlongpl` First letter uppercase.

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

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

\@ACRlongpl All uppercase.

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

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

\@glsaddkey

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

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

```
    \ifcsdef{@gls@user@#1@}%
    {%
       \PackageError{glossaries}%
       {Can't define `\string#5' as helper command
        `\expandafter\string\csname @gls@user@#1@\endcsname' already
        exists}%
       {}%
    }%
    {%
```

```
    \expandafter\newcommand\expandafter*\expandafter
      {\csname @gls@user@#1\endcsname}[2][]{%
        \new@ifnextchar[%
          {\csuse{@gls@user@#1@}{##1}{##2}}%
          {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
    \csdef{@gls@user@#1@}##1##2[##3]{%
      \@gls@field@link{##1}{##2}{#3{##2}##3}%
    }%
    \newrobustcmd*{#5}{%
      \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
  }%
```

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

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

Finally the all caps version (modified):

```
  \ifcsdef{@GLS@user@#1@}%
  {%
     \PackageError{glossaries}%
     {Can't define `\string#7' as helper command
      `\expandafter\string\csname @GLS@user@#1@\endcsname' already
       exists}%
     {}%
  }%
  {%
    \expandafter\newcommand\expandafter*\expandafter
      {\csname @GLS@user@#1\endcsname}[2][]{%
        \new@ifnextchar[%
          {\csuse{@GLS@user@#1@}{##1}{##2}}%
          {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
```

```
      \csdef{@GLS@user@#1@}##1##2[##3]{%
        \@gls@field@link[\let\glscapscase\@thirdofthree]%
           {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
      }%
      \newrobustcmd*{#7}{%
        \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
    }%
  }%
  {%
    \PackageError{glossaries-extra}{Key '#1' already exists}{}%
  }%
}
```

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

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

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

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

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

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

Store the category label for convenience.

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

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

```
\ifdef\do@glsdisablehyperinlist
{%
  \let\@glsxtr@do@glsdisablehyperinlist\do@glsdisablehyperinlist
  \renewcommand*{\do@glsdisablehyperinlist}{%
```

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

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

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

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

`lt@glslink@opts`

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

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

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

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

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

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

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

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

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

```
      {\glsxtrdoautoindexname{#1}{dualindex}}%
      {#2}}%
     {#2}%
  \fi
 }%
 {}%
}
```

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

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

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

Similarly for the "noidx" version:

s@noidxglossary

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

xtr@do@@wrindex

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

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

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

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

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

alt@gls@hyp@opt User version

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

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

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

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

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

rSetAltModifier

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

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

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

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

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

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

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

Reset \@glslink with patched versions:

```
\ifcsundef{hyperlink}%
{%
```

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

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

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

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

@glsxtr@p@text@

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

@Glsxtr@p@text@

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

@GLSxtr@p@text@

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

lsxtr@p@plural@

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

lsxtr@p@plural@

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

LSxtr@p@plural@

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

glsxtr@p@short@

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

Glsxtr@p@short@

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

GLSxtr@p@short@

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

sxtr@p@shortpl@

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

sxtr@p@shortpl@

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

```
    \glsabbrvfont{\Glsentryshortpl{#2}}#3%
  }%
}

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

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

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

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

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

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

\@cGls@

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

\@cglspl@

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

\@cGlspl@

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

Add all upper case versions not provided by glossaries.

\cGLS

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

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

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

\@cGLS@

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

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

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

\cGLSpl

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

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

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

\@cGLSpl@

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

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

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

Modify the trigger formats to check for the regular attribute.

\cglsformat

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

\cGlsformat

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

\cglsplformat

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

\cGlsplformat

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

New code similar to above for unit counting.

defunitcounters

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

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][]{%
    \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
  \let\@gls@link@checkfirsthyper\@glsxtr@org@checkfirsthyper
  \glssetcategoryattribute{acronym}{regular}{false}%
  \setacronymstyle{long-short}%
}
```

\glsacspace Allow the user to customise the maximum value.

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

\glsacspacemax Value used in the above.

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

1.3.7 Indexing and Displaying Glossaries

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

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

r@reg@glosslist

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

Save the original definition of \makeglossaries:

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

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

\makeglossaries

```
\renewcommand*{\makeglossaries}[1][]{%
 \ifblank{#1}%
 {\@glsxtr@org@makeglossaries}%
 {%
   \edef\@glsxtr@reg@glosslist{#1}%
   \ifundef{\glswrite}{\newwrite\glswrite}{}%
   \protected@write\@auxout{}{\string\providecommand
     \string\@glsorder[1]{}}
   \protected@write\@auxout{}{\string\providecommand
     \string\@istfilename[1]{}}
   \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
   \protected@write\@auxout{}{\string\@glsorder{\glsorder}}
   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
```

Iterate through each supplied glossary type and activate it.

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

New glossaries must be created before \makeglossaries:

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

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

Any subsequence instances of this command should have no effect

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

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

```
  \@disable@onlypremakeg
```

Allow see key:

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

Suppress warning about no `\makeglossaries`

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

Adjust display number list to check for type:

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

Adjust entry list:

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

Adjust number list loop

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

Only sanitize sort for normal indexing glossaries.

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

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

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

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

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

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

Check automake setting:

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

Display number list for the regular version:

splaynumberlist

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

Display number list for the “noidx” version:

splaynumberlist

```
\newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \def\@gls@noidxloclist@sep{%
      \def\@gls@noidxloclist@sep{%
        \def\@gls@noidxloclist@sep{%
          \glsnumlistsep
        }%
        \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
      }%
    }%
    \def\@gls@noidxloclist@finalsep{}%
    \def\@gls@noidxloclist@prev{}%
    \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
    \@gls@noidxloclist@finalsep
    \@gls@noidxloclist@prev
  }%
  {%
    ??\glsdoifexists{#1}%
    {%
      \GlossariesWarning{Missing location list for ‘#1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
}%
```

And for the number list loop:

@numberlistloop

```
\newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
  \let\@gls@org@glsseeformat\glsseeformat
  \let\glsnoidxdisplayloc#2\relax
  \let\glsseeformat#3\relax
  \ifdef\@gls@loclist
  {%
    \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
  }%
  {%
    ??\glsdoifexists{#1}%
    {%
```

```
      \GlossariesWarning{Missing location list for ‘##1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
  \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
  \let\glsseeformat\@gls@org@glsseeformat
}%
```

Same for entry number list.

entrynumberlist

```
\newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \glsnoidxloclist{\@gls@loclist}%
  }%
  {%
    ??\glsdoifexists{#1}%
    {%
      \GlossariesWarning{Missing location list for ‘#1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
}%
```

entrynumberlist

```
\let\@glsxtr@idx@entrynumberlist\glsentrynumberlist
```

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

@print@glossary

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

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

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

oGlsWarningHead Header message.

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

rningEmptyStart No entries have been added to the glossary.

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

arningEmptyMain The default "main" glossary is empty.

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

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

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

arningCheckFile Advisory message to check the file contents.

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

WarningAutoMake Message when automake option has been used.

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

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

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

    \item Run the external (Perl) application:

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

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

WarningMisMatch Mismatching `\makenoidxglossaries`.

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

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

arningBuildInfo Build advice.

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

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

    \item Run the external (Lua) application:

          \texttt{makeglossaries-lite \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
}
```

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}%
  }
```

`\@glsname@` Redefine to use accessibility support.

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

`\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}%
  }
```

`\@Glsname@` Redefine to use accessibility support.

```
\def\@Glsname@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccessname{#2}#3}%
}
```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessname}[1]{%
    \glsnameaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentryname{#1}}%
    }%
    {#1}%
  }
```

\@GLSname@ Redefine to use accessibility support.

```
\def\@GLSname@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccessname{#2}#3}%
}
```

\glsaccesstext Display the text value (no link and no check for existence).

```
\newcommand*{\glsaccesstext}[1]{%
  \glstextaccessdisplay
  {%
    \glsentrytext{#1}%
  }%
  {#1}%
}
```

\@glstext@ Redefine to use accessibility support.

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

\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}%
}
```

\@Glstext@ Redefine to use accessibility support.

```
\def\@Glstext@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccesstext{#2}#3}%
}
```

\GLSaccesstext Display the text value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccesstext}[1]{%
  \glstextaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrytext{#1}}%
  }%
  {#1}%
}
```

\@GLStext@ Redefine to use accessibility support.

```
\def\@GLStext@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccesstext{#2}#3}%
}
```

glsaccessplural Display the plural value (no link and no check for existence).

```
\newcommand*{\glsaccessplural}[1]{%
  \glspluralaccessdisplay
  {%
    \glsentryplural{#1}%
  }%
  {#1}%
}
```

\@glsplural@ Redefine to use accessibility support.

```
\def\@glsplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsaccessplural{#2}#3}%
}
```

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}%
}
```

\@Glsplural@ Redefine to use accessibility support.

```
\def\@glsplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccessplural{#2}#3}%
}
```

GLSaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessplural}[1]{%
  \glspluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentryplural{#1}}%
  }%
  {#1}%
}
```

\@GLSplural@ Redefine to use accessibility support.

```
\def\@GLSplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccessplural{#2}#3}%
}
```

\glsaccessfirst Display the first value (no link and no check for existence).

```
\newcommand*{\glsaccessfirst}[1]{%
  \glsfirstaccessdisplay
  {%
    \glsentryfirst{#1}%
```

```
    }%
    {#1}%
  }
```

\@glsfirst@ Redefine to use accessibility support.

```
\def\@glsfirst@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsaccessfirst{#2}#3}%
}
```

\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}%
  }
```

\@Glsfirst@ Redefine to use accessibility support.

```
\def\@Glsfirst@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccessfirst{#2}#3}%
}
```

\GLSaccessfirst Display the first value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessfirst}[1]{%
    \glsfirstaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentryfirst{#1}}%
    }%
    {#1}%
  }
```

\@GLSfirst@ Redefine to use accessibility support.

```
\def\@GLSfirst@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccessfirst{#2}#3}%
}
```

cessfirstplural Display the firstplural value (no link and no check for existence).

```
  \newcommand*{\glsaccessfirstplural}[1]{%
    \glsfirstpluralaccessdisplay
    {%
      \glsentryfirstplural{#1}%
    }%
    {#1}%
  }
```

glsfirstplural@ Redefine to use accessibility support.

```
\def\@glsfirstplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsaccessfirstplural{#2}#3}%
}
```

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}%
  }
```

Glsfirstplural@ Redefine to use accessibility support.

```
\def\@Glsfirstplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccessfirstplural{#2}#3}%
}
```

cessfirstplural Display the firstplural value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessfirstplural}[1]{%
    \glsfirstpluralaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
    }%
    {#1}%
  }
```

GLSfirstplural@ Redefine to use accessibility support.

```
\def\@GLSfirstplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccessfirstplural{#2}#3}%
}
```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```
  \newcommand*{\glsaccesssymbol}[1]{%
    \glssymbolaccessdisplay
    {%
      \glsentrysymbol{#1}%
    }%
    {#1}%
  }
```

\@glssymbol@ Redefine to use accessibility support.

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

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}%
}
```

\@Glssymbol@ Redefine to use accessibility support.

```
\def\@Glssymbol@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccesssymbol{#2}#3}%
}
```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccesssymbol}[1]{%
  \glssymbolaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
  }%
  {#1}%
}
```

\@GLSsymbol@ Redefine to use accessibility support.

```
\def\@GLSsymbol@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccesssymbol{#2}#3}%
}
```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```
\newcommand*{\glsaccesssymbolplural}[1]{%
  \glssymbolpluralaccessdisplay
  {%
    \glsentrysymbolplural{#1}%
  }%
  {#1}%
}
```

lssymbolplural@ Redefine to use accessibility support.

```
\def\@glssymbolplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsaccesssymbolplural{#2}#3}%
}
```

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}%
  }
```

lssymbolplural@ Redefine to use accessibility support.

```
\def\@Glssymbolplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccesssymbolplural{#2}#3}%
}
```

esssymbolplural Display the symbolplural value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccesssymbolplural}[1]{%
    \glssymbolpluralaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
    }%
    {#1}%
  }
```

LSsymbolplural@ Redefine to use accessibility support.

```
\def\@GLSsymbolplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccesssymbolplural{#2}#3}%
}
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
  \newcommand*{\glsaccessdesc}[1]{%
    \glsdescriptionaccessdisplay
    {%
      \glsentrydesc{#1}%
    }%
    {#1}%
  }
```

\@glsdesc@ Redefine to use accessibility support.

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

\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}%
  }
```

\@Glsdesc@ Redefine to use accessibility support.

```
\def\@Glsdesc@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccessdesc{#2}#3}%
}
```

\GLSaccessdesc Display the desc value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessdesc}[1]{%
  \glsdescriptionaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrydesc{#1}}%
  }%
  {#1}%
}
```

\@GLSdesc@ Redefine to use accessibility support.

```
\def\@GLSdesc@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccessdesc{#2}#3}%
}
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
\newcommand*{\glsaccessdescplural}[1]{%
  \glsdescriptionpluralaccessdisplay
  {%
    \glsentrydescplural{#1}%
  }%
  {#1}%
}
```

@glsdescplural@ Redefine to use accessibility support.

```
\def\@glsdescplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\glsaccessdescplural{#2}#3}%
}
```

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}%
}
```

@Glsdescplural@ Redefine to use accessibility support.

```
\def\@Glsdescplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\Glsaccessdescplural{#2}#3}%
}
```

ccessdescplural Display the descplural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessdescplural}[1]{%
  \glsdescriptionpluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
  }%
  {#1}%
}
```

@GLSdescplural@ Redefine to use accessibility support.

```
\def\@GLSdescplural@#1#2[#3]{%
  \@gls@field@link{#1}{#2}{\GLSaccessdescplural{#2}#3}%
}
```

\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.

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

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

Short cut to determine if an entry has a regular category.

```
\newcommand{\glsifregular}[3]{%
  \glsifregularcategory{\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\xcapitalisewords\expandafter
    {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
}
```

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}}%
      \glsifattribute{#1}{glossdesc}{firstuc}%
```

```
      {%
        \Glsaccessdesc{#1}%
      }%
      {%
        \glsifattribute{#1}{glossdesc}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \glsdescriptionaccessdisplay
          {%
            \glsxtrfieldtitlecase{#1}{desc}%
          }%
          {#1}%
        }%
        {%
          \glsaccessdesc{#1}%
        }%
      }%
    }%
  }
}
{
  \renewcommand*{\glossentrydesc}[1]{%
    \glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glsifattribute{#1}{glossdesc}{firstuc}%
      {%
        \Glsentrydesc{#1}%
      }%
      {%
        \glsifattribute{#1}{glossdesc}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \glsxtrfieldtitlecase{#1}{desc}%
        }%
        {%
          \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}}%
      \glsifattribute{#1}{glossname}{firstuc}%
      {%
        \glsnameaccessdisplay
        {%
          \glsnamefont{\Glsentryname{#1}}%
        }%
        {#1}%
      }%
      {%
        \glsifattribute{#1}{glossname}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \glsnameaccessdisplay
          {%
            \glsnamefont{\glsxtrfieldtitlecase{#1}{name}}%
          }%
          {#1}%
        }%
        {%
          \glsifattribute{#1}{glossname}{uc}%
          {%
            \glsnameaccessdisplay
            {%
```

Hide the label from the upper-casing command.

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

Do post-name hook:

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
{
  \renewcommand*{\glossentryname}[1]{%
```

```
    \glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glsifattribute{#1}{glossname}{firstuc}%
      {%
        \glsnamefont{\Glsentryname{#1}}%
      }%
      {%
        \glsifattribute{#1}{glossname}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \glsnamefont{\glsxtrfieldtitlecase{#1}{name}}%
        }%
        {%
          \glsifattribute{#1}{glossname}{uc}%
          {%
```

Hide the label from the upper-casing command.

```
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \glsnamefont{\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\glsnamefont\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}}%
      \glsnameaccessdisplay
      {%
        \glsnamefont{\Glsentryname{#1}}%
      }%
      {#1}%
```

Do post-name hook:

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
{
  \renewcommand*{\Glossentryname}[1]{%
    \glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glsnamefont{\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}%
}
```

`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.

```
\ifdef\@gls@preglossaryhook
{
  \renewcommand*{\@gls@preglossaryhook}{%
    \@glsxtr@activate@initialtagging
    \let\@glsxtr@org@postdescription\glspostdescription
    \renewcommand*{\glspostdescription}{%
      \glsxtrpostdescription
      \@glsxtr@org@postdescription
    }%
  }%
}
{}
```

`postdescription` This command will only be used if `\@gls@preglossaryhook` is available *and* the glossary style uses `\glspostdescription` without modifying it. (The nopostdesc option will suppress this.)

```
\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 spacefactor.

```
   \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}%
 {%
   \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
 }%
 {}%
}
```

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][]{%
  \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}
```

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

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

longdefaultfont

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

brvpluralsuffix Default plural suffix.

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

\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}}#3%
    }%
    \@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}}#3%
    }%
    \@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}}#3}%
    }%
    \@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{\glsaccesslong{#2}#3}%
    \@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{\Glsaccesslong{#2}#3}%
    \@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{\glsaccesslong{#2}#3}}%
    \@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}}#3%
    }%
    \@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}}#3%
    }%
    \@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}}#3}%
    }%
    \@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{\glsaccesslongpl{#2}#3}%
    \@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{\Glsaccesslongpl{#2}#3}%
    \@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{\glsaccesslongpl{#2}#3}}%
    \@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

```
3640          \glscapscase
3641          {%
```

Subsequent singular form, don't adjust case:

```
3642            \glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert
3643          }%
3644          {%
```

Subsequent singular form, make first letter upper case:

```
3645            \glsabbrvfont{\Glsaccessshort{\glslabel}}\glsinsert
3646          }%
3647          {%
```

Subsequent singular form, all caps:

```
3648            \mfirstucMakeUppercase
3649              {\glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert}%
3650          }%
3651        }%
3652      }%
3653      {%
```

First use:

```
3654        \glsifplural
3655        {%
```

First use plural form:

```
3656          \glscapscase
3657          {%
```

First use plural form, don't adjust case:

```
3658            \glsxtrfullplformat{\glslabel}{\glsinsert}%
3659          }%
3660          {%
```

First use plural form, make first letter upper case:

```
3661            \Glsxtrfullplformat{\glslabel}{\glsinsert}%
3662          }%
3663          {%
```

First use plural form, all caps:

```
3664            \mfirstucMakeUppercase
3665              {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
3666          }%
3667        }%
3668        {%
```

First use singular form

```
3669          \glscapscase
3670          {%
```

First use singular form, don't adjust case:

```
3671            \glsxtrfullformat{\glslabel}{\glsinsert}%
```

```
3672          }%
3673          {%
```

First use singular form, make first letter upper case:

```
3674            \Glsxtrfullformat{\glslabel}{\glsinsert}%
3675          }%
3676          {%
```

First use singular form, all caps:

```
3677            \mfirstucMakeUppercase
3678             {\glsxtrfullformat{\glslabel}{\glsinsert}}%
3679          }%
3680        }%
3681      }%
3682    }%
3683    {%
```

User supplied text.

```
3684      \glscustomtext
3685    }%
3686 }
```

1.6.1 Abbreviation Styles Setup

breviationstyle

```
3687 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
3688   \ifcsundef{@glsabbrv@dispstyle@setup@#2}
3689   {%
3690     \PackageError{glossaries-extra}{Undefined abbreviation style `#2'}{}%
3691   }%
3692   {%
```

Have abbreviations already been defined for this category?

```
3693      \ifcsstring{@glsabbrv@current@#1}{#2}%
3694      {%
```

Style already set.

```
3695      }%
3696      {%
3697        \def\@glsxtr@dostylewarn{}%
3698        \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
3699        {%
3700          \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
3701            style has been switched \MessageBreak
3702            for category `#1', \MessageBreak
3703            but there have already been entries \MessageBreak
3704            defined for this category. Unwanted \MessageBreak
3705            side-effects may result}}%
3706          \@endfortrue
3707        }%
3708        \@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@#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}%
  }%
}
```

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.

long-short

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

Unset the regular attribute if it has been set.

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

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

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

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

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

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

Set this as the default style for general abbreviations:

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

`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={\the\glsshorttok},%
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
    plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

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

```
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-short}%
}
```

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

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

Unset the regular attribute if it has been set.

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

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

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

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

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
```

```
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}}##2\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})},%
    plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-long}%
}
```

footnote Short form followed by long form in footnote on first use. Take care about using `\glsfirst` as this won't suppress the hyperlink. (Perhaps modify `\glsfirst` to reflect nohyperfirst attribute?)

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

```
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\footnote{\protect\glsfirstlongfont{\the\glslongpltok}}},%
    plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
```

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

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

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

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

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

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}}##2%
    \protect\footnote{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2%
    \protect\footnote{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}}##2%
    \protect\footnote{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}}##2%
    \protect\footnote{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
```

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

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
```

```
    \glsfirstabbrvfont{\Glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
}
```

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. This deferment won't occur with \glsfirst.

```
\newabbreviationstyle{postfootnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\footnote{\protect\glsfirstlongfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\footnote{\protect\glsfirstlongfont{\the\glslongpltok}}},%
    plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
```

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

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse{\glsxtrdopostpunc{\protect\footnote
        {\glsfirstlongfont{\glsentrylong{\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}{\glspluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
```

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

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}}##2%
  }%
```

```
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}}##2%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}}##2%
  }%
```

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

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
}
```

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

```
\newabbreviationstyle{short}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsabbrvfont{\the\glsshorttok}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
    text={\protect\glsabbrvfont{\the\glsshorttok}},
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

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

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

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

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

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\Glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}##2\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}}##2%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}}##2%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2%
  }%
}
```

Set this as the default style for acronyms:

```
\setabbreviationstyle[acronym]{short}
```

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

```
\newabbreviationstyle{short-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\glsxtrinlinefullformat{\the\glslabeltok}{}},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
```

```
    text={\protect\glsabbrvfont{\the\glsshorttok}},
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

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

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

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

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslongpl{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}}##2\glsxtrfullsep{##1}%
    (\glsfirstlongfont{\glsaccesslong{##1}})%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}}##2\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}}##2%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}}##2%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}}##2%
  }%
}
```

`long-desc` Provide a style that only displays the long form, but the long and short form can be displayed with the "full" commands that use the inline format. The predefined glossary styles won't

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

```
\newabbreviationstyle{long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\protect\protect\glsfirstlongfont{\the\glslongtok}},
    sort={\the\glslongtok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
    text={\the\glslongtok},
    plural={\the\glslongpltok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

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

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

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

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

```
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}}##2%
  }%
}
```

`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}%
}
```

1.6.3 Predefined Styles (Small Capitals)

These styles use:

`\glsxtrscfont`

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

and for the default short form suffix:

`\glsxtrscsuffix`

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

`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}}%
}
```

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}}%
}
```

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}}%
}
```

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}}%
}
```

short-sc

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

Mostly as short style:

```
  \GlsXtrUseAbbrStyleFmts{short}%
```

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

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

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}}%
}
```

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

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

Mostly as long style:

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

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

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

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

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

```
}%
{%
```

Mostly as long style:

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

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

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

footnote-sc

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

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{footnote}%
```

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

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

postfootnote-sc

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

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{postfootnote}%
```

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

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

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}}
```

and for the default short form suffix:

\glsxtrsmsuffix

```
4259 \newcommand*{\glsxtrsmsuffix}{\glspluralsuffix}
```

long-short-sm

```
4260 \newabbreviationstyle{long-short-sm}%
4261 {%
4262   \GlsXtrUseAbbrStyleSetup{long-short}%
4263 }%
4264 {%
```

Mostly as long-short style:

```
4265   \GlsXtrUseAbbrStyleFmts{long-short}%
4266   \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
4267   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
4268 }
```

g-short-sm-desc

```
4269 \newabbreviationstyle{long-short-sm-desc}%
4270 {%
4271   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
4272 }%
4273 {%
```

Mostly as long-short-desc style:

```
4274   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
4275   \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
4276   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
4277 }
```

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

```
4278 \newabbreviationstyle{short-sm-long}%
4279 {%
4280   \GlsXtrUseAbbrStyleSetup{short-long}%
4281 }%
4282 {%
```

Mostly as short-long style:

```
4283   \GlsXtrUseAbbrStyleFmts{short-long}%
4284   \renewcommand*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%
4285   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
4286 }
```

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

```
4287 \newabbreviationstyle{short-sm-long-desc}%
4288 {%
4289   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
4290 }%
4291 {%
```

Mostly as short-long-desc style:

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

`short-sm`

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

Mostly as short style:

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

`short-sm-desc`

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

Mostly as short style:

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

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

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

Mostly as long style:

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

`long-desc-sm` The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

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

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

Mostly as long style:

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

footnote-sm

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

Mostly as long style:

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

postfootnote-sm

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

Mostly as long style:

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

1.6.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

long-short-em

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

Mostly as long-short style:

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

```
    \renewcommand*\glsabbrvfont[1]{\emph{##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]{\emph{##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]{\emph{##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]{\emph{##1}}%
}
```

short-em

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

Mostly as short style:

```
  \GlsXtrUseAbbrStyleFmts{short}%
  \renewcommand*\glsabbrvfont[1]{\emph{##1}}%
}
```

short-em-desc

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

Mostly as short style:

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

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

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

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{long}%
  \renewcommand*\glsabbrvfont[1]{\emph{##1}}%
}
```

long-desc-em The emphasized font will only be used if the short form is explicitly invoked through commands like \glsshort.

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

Mostly as long style:

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

footnote-em

```
\newabbreviationstyle{footnote-em}%
{%
  \GlsXtrUseAbbrStyleSetup{footnote}%
}%
{%
```

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{footnote}%
  \renewcommand*\glsabbrvfont[1]{\emph{##1}}%
}
```

postfootnote-em

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

Mostly as long style:

```
  \GlsXtrUseAbbrStyleFmts{postfootnote}%
  \renewcommand*\glsabbrvfont[1]{\emph{##1}}%
}
```

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

\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
```

New definitions

```
  \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
}
```

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\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
}
```

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]{%
  \glsxtrshort[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}[]%
}
```

\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}}
}
```

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

Glossary

This document is incomplete. The external file associated with the glossary 'main' (which should be called `glossaries-extra-code.gls2`) hasn't been created.

Check the contents of the file `glossaries-extra-code.glo2`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

If you don't want this glossary, add `nomain` to your package option list when you load `glossaries-extra.sty`. For example:

```
\usepackage[nomain]{glossaries-extra}
```

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite "glossaries-extra-code"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "glossaries-extra-code"
  ```

Then rerun LaTeX on this document.
This message will be removed once the problem has been fixed.

Change History

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

A

G

H

I

O

P