

Documented Code For glossaries v4.45

Nicola L.C. Talbot

Dickimaw Books

<http://www.dickimaw-books.com/>

2020-02-13

This is the documented code for the glossaries package. This bundle comes with the following documentation:

glossariesbegin.pdf If you are a complete beginner, start with “The glossaries package: a guide for beginners”.

glossary2glossaries.pdf If you are moving over from the obsolete glossary package, read “Upgrading from the glossary package to the glossaries package”.

glossaries-user.pdf For the main user guide, read “glossaries.sty v4.45: L^AT_EX2_ε Package to Assist Generating Glossaries”.

mfirstuc-manual.pdf The commands provided by the mfirstuc package are briefly described in “mfirstuc.sty: uppercasing first letter”.

glossaries-code.pdf This document is for advanced users wishing to know more about the inner workings of the glossaries package.

INSTALL Installation instructions.

CHANGES Change log.

README.md Package summary.

The user level commands described in the user manual (glossaries-user.pdf) may be considered “future-proof”. Even if they become deprecated, they should still work for old documents (although they may not work in a document that also contains new commands introduced since the old commands were deprecated, and you may need to specify a compatibility mode).

The internal commands in *this* document that aren’t documented in the *user manual* should not be considered future-proof and are liable to change. If you want a new user level command, you can post a feature request at <http://www.dickimaw-books.com/feature-request.html>. If you are a package writer wanting to integrate your package with glossaries, it’s better to request a new user level command than to hack these internals.

Contents

1	Main Package Code	4
1.1	Package Definition	4
1.2	Package Options	5
1.3	Predefined Text	39
1.4	Xindy	49
1.5	Loops and conditionals	58
1.6	Defining new glossaries	64
1.7	Defining new entries	69
1.8	Resetting and unsetting entry flags	95
1.9	Keeping Track of How Many Times an Entry Has Been Unset	98
1.10	Loading files containing glossary entries	103
1.11	Using glossary entries in the text	103
1.12	Adding an entry to the glossary without generating text	163
1.13	Creating associated files	164
1.14	Writing information to associated files	185
1.15	Glossary Entry Cross-References	194
1.16	Displaying the glossary	196
1.17	Acronyms	224
1.18	Predefined acronym styles	228
1.19	Predefined Glossary Styles	260
1.20	Debugging Commands	261
1.21	Compatibility with version 2.07 and below	266
2	Prefix Support (glossaries-prefix Code)	267
3	Glossary Styles	274
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	274
3.2	In-line Style (glossary-inline.sty)	276
3.3	List Style (glossary-list.sty)	279
3.4	Glossary Styles using longtable (the glossary-long package)	282
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	288
3.6	Glossary Styles using longtable (the glossary-longragged package)	293
3.7	Glossary Styles using multicol (glossary-mcols.sty)	298
3.8	Glossary Styles using supertabular environment (glossary-super package)	304
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	311
3.10	Tree Styles (glossary-tree.sty)	317

4 Backwards Compatibility	327
4.1 glossaries-compatible-207	327
4.2 glossaries-compatible-307	333
5 Accessibility Support (glossaries-accsupp Code)	347
5.1 Defining Replacement Text	349
5.2 Accessing Replacement Text	353
5.3 Displaying the Glossary	378
5.4 Acronyms	379
5.5 Debugging Commands	393
6 Multi-Lingual Support	396
6.1 Polyglossia Captions	396
Glossary	398
Change History	399
Index	425

1 Main Package Code

1.1 Package Definition

This package requires $\text{\LaTeX}2_{\epsilon}$.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2020/02/13 v4.45 (NLCT)]
```

Required packages:

```
3 \RequirePackage{ifthen}
4 \RequirePackage{xkeyval}[2006/11/18]
5 \RequirePackage{mfirstuc}
```

The textcase package has much better case changing handling, so use `\MakeTextUppercase` instead of `\MakeUppercase`

```
6 \RequirePackage{textcase}
7 \renewcommand*{\mfirstucMakeUppercase}{\MakeTextUppercase}%
8 \RequirePackage{xfor}
```

```
9 \RequirePackage{datatool-base}
```

Need to use `\new@ifnextchar` instead of `\@ifnextchar` in commands that have a final optional argument (such as `\gls`) so require `.` Thanks to Morten Høgholm for suggesting this. (This has replaced using the `xspace` package.)

```
10 \RequirePackage{amsgen}
```

As from v3.0, now loading `etoolbox` (this is now redundant as `datatool-base` loads `etoolbox`):

```
11 \RequirePackage{etoolbox}
```

Check if doc has been loaded.

```
f@gls@docloaded
```

```
12 \newif\if@gls@docloaded
13 \@ifpackageloaded{doc}%
14 {%
15   \@gls@docloadedtrue
16 }%
17 {%
18   \@ifclassloaded{nlctdoc}{\@gls@docloadedtrue}{\@gls@docloadedfalse}%
19 }
20 \if@gls@docloaded
```

\doc has been loaded, so some modifications need to be made to ensure both packages can work together. The amount of conflict has been reduced as from v4.11 and no longer involves patching internal commands.

\PrintChanges needs to use doc's version of theglossary, so save that.

org@theglossary

```
21 \let\glsorg@theglossary\theglossary
```

@endtheglossary

```
22 \let\glsorg@endtheglossary\endtheglossary
```

\PrintChanges Now redefine \PrintChanges so that it uses the original theglossary environment.

```
23 \let\glsorg@PrintChanges\PrintChanges
24 \renewcommand{\PrintChanges}{%
25   \begingroup
26     \let\theglossary\glsorg@theglossary
27     \let\endtheglossary\glsorg@endtheglossary
28     \glsorg@PrintChanges
29   \endgroup
30 }
```

End of doc stuff.

```
31 \fi
```

1.2 Package Options

debug Switch on debug mode. This will also cancel the nowarn option. This is now a choice key.

```
32 \newif\if@gls@debug
33 \define@choicekey{glossaries.sty}{debug}{\gls@debug@val\gls@debug@nr}%
34 {true,false,showtargets,showaccsupp}[true]{%
35   \ifcase\gls@debug@nr\relax
36     % debug=true
37     \@gls@debugtrue
38     \renewcommand*\GlossariesWarning[1]{%
39       \PackageWarning{glossaries}{##1}%
40     }%
41     \renewcommand*\GlossariesWarningNoLine[1]{%
42       \PackageWarningNoLine{glossaries}{##1}%
43     }%
44     \let\@glsshowtarget\@gobble
45     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
46   \or
47     % debug=false
48     \@gls@debugfalse
49     \let\@glsshowtarget\@gobble
50     \let\@glsshowaccsupp\@gobblethree
51     \PackageInfo{glossaries}{debug mode OFF}%
```

```

52 \or
53 % debug=showtargets
54 \@gls@debugtrue
55 \renewcommand*{\GlossariesWarning}[1]{%
56   \PackageWarning{glossaries}{##1}%
57 }%
58 \renewcommand*{\GlossariesWarningNoLine}[1]{%
59   \PackageWarningNoLine{glossaries}{##1}%
60 }%
61 \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
62 \renewcommand{\@glsshowtarget}{\@glsshowtarget}%
63 \or
64 % debug=showaccsupp
65 \@gls@debugtrue
66 \renewcommand*{\GlossariesWarning}[1]{%
67   \PackageWarning{glossaries}{##1}%
68 }%
69 \renewcommand*{\GlossariesWarningNoLine}[1]{%
70   \PackageWarningNoLine{glossaries}{##1}%
71 }%
72 \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
73 \renewcommand{\@glsshowaccsupp}{\glsshowaccsupp}%
74 \fi
75 }

```

`\glsshowtarget` If debug=showtargets, show the hyperlink target name in the margin.

```

76 \newcommand*{\glsshowtarget}[1]{%
77   \ifmmode
78     \nfss@text{\glsshowtargetfont [1]}%
79   \else
80     \ifinner

```

Grouping no longer required as new `\@glsshowtarget` adds scoping but retained here in case any existing documents are using `\glsshowtarget` elsewhere.

```

81     {\glsshowtargetfont [1]}%
82   \else
83     \glsshowtargetouter{#1}%
84   \fi
85 \fi
86 }

```

`showtargetouter`

```

87 \newcommand*{\glsshowtargetouter}[1]{%
88   \glsshowtargetsymbolsymbol\marginpar{\glsshowtargetsymbolsymbol\glsshowtargetfont #1}}

```

`howtargetsymbols`

```

89 \newcommand*{\glsshowtargetsymbols}{\tiny$\triangleright$}

```

`sshowtargetfont`

```

90 \newcommand*{\glsshowtargetfont}{\ttfamily\footnotesize}

```

`\@glsshowtarget` debug=showtargets will redefine this.

```

91 \newcommand*{\@glsshowtarget}[1]{%
  92 \newrobustcmd*{\@@glsshowtarget}[1]{%
    93   \begingroup
    94   \protected@edef\@gls@tmp{#1}%
    95   \@onelevel@sanitize\@gls@tmp
    96   \expandafter\glsshowtarget\expandafter{\@gls@tmp}%
    97   \endgroup
    98 }
```

`@@glsshowtarget` Need to detokenize the label in the event that it contains awkward characters like underscores.

`@glsshowaccsupp` debug=showaccsupp will redefine this.

```

99 \newcommand*{\@glsshowaccsupp}[3]{%
100 \newrobustcmd*{\glsshowaccsupp}[3]{%
101   \ifstrepty{#1}%
102   {\@@glsshowtarget{/#2 (#3)}}%
103   {\@@glsshowtarget{/#2 (#3) [#1]}}%
104 }
```

`\glsshowaccsupp` Just use `\@@glsshowtarget` since it basically needs to do the same thing.

Determine what to do if the see key is used before `\makeglossaries`. The default is to produce an error.

`gls@see@noindex`

```

105 \newcommand*{\@gls@see@noindex}{%
106   \PackageError{glossaries}%
107   {'\gls@xr@key' key may only be used after \string\makeglossaries\space
108   or \string\makenoidxglossaries\space (or move
109   \string\newglossaryentry\space
110   definitions into the preamble))%
111   {You must use \string\makeglossaries\space
112   or \string\makenoidxglossaries\space before defining
113   any entries that have a '\gls@xr@key' key. It may
114   be that the 'see' key has been written to the .glsdefs
115   file from the previous run, in which case you need to
116   move your definitions
117   to the preamble if you don't want to use
118   \string\makeglossaries\space
119   or \string\makenoidxglossaries}%
120 }
```

`seenoinindex`

```

121 \define@choicekey{glossaries.sty}{seenoinindex}%
122 [\gls@seenoinindex@val\gls@seenoinindex@nr]{error,warn,ignore}{%
123   \ifcase\gls@seenoinindex@nr
```

```

124 \renewcommand*{\@gls@see@noindex}{%
125 \PackageError{glossaries}%
126 {\@gls@xr@key' key may only be used after \string\makeglossaries\space
127 or \string\makenoidxglossaries}%
128 {You must use \string\makeglossaries\space
129 or \string\makenoidxglossaries\space before defining
130 any entries that have a \@gls@xr@key' key}%
131 }%
132 \or
133 \renewcommand*{\@gls@see@noindex}{%
134 \GlossariesWarning{\@gls@xr@key' key ignored}%
135 }%
136 \or
137 \renewcommand*{\@gls@see@noindex}{}%
138 \fi
139 }

```

toc The toc package option will add the glossaries to the table of contents. This is a boolean key, if the value is omitted it is taken to be true.

```
140 \define@boolkey{glossaries.sty}[gls]{toc}[true]{}

```

numberline The numberline package option adds \numberline to \addcontentsline. Note that this option only has an effect if used in with toc=true.

```
141 \define@boolkey{glossaries.sty}[gls]{numberline}[true]{}

```

\@glossarysec The sectional unit used to start the glossary is stored in \@glossarysec. If chapters are defined, this is initialised to chapter, otherwise it is initialised to section.

```

142 \ifcsundef{chapter}%
143 {\newcommand*\@glossarysec{section}}%
144 {\newcommand*\@glossarysec{chapter}}

```

section The section key can be used to set the sectional unit. If no unit is specified, use section as the default. The starred form of the named sectional unit will be used. If you want some other way to start the glossary section (e.g. a numbered section) you will have to redefine \glossarysection.

```

145 \define@choicekey{glossaries.sty}{section}{part,chapter,section,%
146 subsection,subsubsection,paragraph,subparagraph}[section]{}
147 \renewcommand*\@glossarysec{#1}

```

Determine whether or not to use numbered sections.

glossarysecstar

```
148 \newcommand*\@glossarysecstar{*}

```

lossaryseclabel

```
149 \newcommand*\@glossaryseclabel{}

```

\glsautoprefix Prefix to add before label if automatically generated:

```
150 \newcommand*\glsautoprefix{}

```


numberedsection

```

151 \define@choicekey{glossaries.sty}{numberedsection}%
152 [\gls@numberedsection@val\gls@numberedsection@nr]{%
153 false,nolabel,autolabel,nameref}[nolabel]{%
154 \ifcase\gls@numberedsection@nr\relax
155 \renewcommand*{\@@glossarysecstar}{*}%
156 \renewcommand*{\@@glossaryseclabel}{}%
157 \or
158 \renewcommand*{\@@glossarysecstar}{}%
159 \renewcommand*{\@@glossaryseclabel}{}%
160 \or
161 \renewcommand*{\@@glossarysecstar}{}%
162 \renewcommand*{\@@glossaryseclabel}{%
163 \label{\glsautoprefix\@glo@type}}%
164 \or
165 \renewcommand*{\@@glossarysecstar}{*}%
166 \renewcommand*{\@@glossaryseclabel}{%
167 \protected@edef\@currentlabelname{\glossarytoctitle}%
168 \label{\glsautoprefix\@glo@type}}%
169 \fi
170 }

```

The default glossary style is stored in `\@glossary@default@style`. This is initialised to `list`. (The `list` style is defined in the accompanying package described in [section 1.19](#).) Note that the `list` style is incompatible with `classicthesis` so change the default to `index` if that package has been loaded.

y@default@style

```

171 \ifpackageloaded{classicthesis}
172 {\newcommand*{\@glossary@default@style}{index}}
173 {\newcommand*{\@glossary@default@style}{list}}

```

style The default glossary style can be changed using the `style` package option. The value can be the name of any defined glossary style. The glossary style is set at the beginning of the document, so you can still use the `style` key to set a style that is defined in another package. This package comes with some predefined styles that are defined in [section 1.19](#). This now uses `\def` instead of `\renewcommand` as `\@glossary@default@style` may have been set to `\relax`.

```

174 \define@key{glossaries.sty}{style}{%
175 \def\@glossary@default@style{#1}%
176 }

```

Each `\DeclareOptionX` needs a corresponding `\DeclareOption` so that it can be passed as a document class option, so define a command that will implement both.

s@declareoption

```

177 \newcommand*{\@gls@declareoption}[2]{%
178 \DeclareOptionX{#1}{#2}%
179 \DeclareOption{#1}{#2}%
180 }

```

Each entry within a given glossary will have an associated number list. By default, this refers to the page numbers on which that entry has been used, but it can also refer to any counter used in the document (such as the section or equation counters). The default number list format displays the number list “as is”:

aryentrynumbers

```
181 \newcommand*{\glossaryentrynumbers}[1]{#1\gls@save@numberlist{#1}}
```

nonumberlist

Note that the entire number list for a given entry will be passed to `\glossaryentrynumbers` so any font changes will also be applied to the delimiters. The `nonumberlist` package option suppresses the number lists (this simply redefines `\glossaryentrynumbers` to ignore its argument).

```
182 \@gls@declareoption{nonumberlist}{%
183   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
184 }
```

savenumberlist

Provide means to store the number list for entries.

```
185 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{%
186 \gls@savenumberlistfalse}
```

eautionumberlist

```
187 \newcommand*{\@glo@seeautonumberlist{}}
```

eautionumberlist

Automatically activates number list for entries containing the see key.

```
188 \@gls@declareoption{seeautonumberlist}{%
189   \renewcommand*{\@glo@seeautonumberlist}{%
190     \def\@glo@prefix{\glsnextpages}%
191   }%
192 }
```

esclocations

When using `makeindex` or `xindy`, the locations may need to be adjusted to ensure they’re in a format that’s allowed by the indexing application. This involves a bit of hackery and isn’t needed if the locations are all guaranteed to be in the correct form (or if the user is prepared to post-process the glossary file before calling the relevant indexing application) so `esclocations=false` will switch off this mechanism allowing for a faster and more stable approach.

```
193 \define@boolkey{glossaries.sty}[gls]{esclocations}[true]{%
194 \gls@esclocationstrue}
```

\@gls@loadlong

```
195 \newcommand*{\@gls@loadlong}{\RequirePackage{glossary-long}}
```

nolong

This option prevents from being loaded. This means that the glossary styles that use the longtable environment will not be available. This option is provided to reduce overhead caused by loading unrequired packages.

```
196 \@gls@declareoption{nolong}{\renewcommand*{\@gls@loadlong}{}}
```

`\@gls@loadsuper` The package isn't loaded if isn't installed.

```

197 \IfFileExists{supertabular.sty}{%
198   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
199   \newcommand*{\@gls@loadsuper}{}

```

`nosuper` This option prevents from being loaded. This means that the glossary styles that use the supertabular environment will not be available. This option is provided to reduce overhead caused by loading unrequired packages.

```

200 \@gls@declareoption{nosuper}{\renewcommand*{\@gls@loadsuper}{}

```

`\@gls@loadlist`

```

201 \newcommand*{\@gls@loadlist}{\RequirePackage{glossary-list}}

```

`nolist` This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used. If the style is still set to list, the default must be set to `\relax`.

```

202 \@gls@declareoption{nolist}{%
203   \renewcommand*{\@gls@loadlist}{%
204     \ifdefstring{\@glossary@default@style}{list}%
205     {\let\@glossary@default@style\relax}%
206     }%
207   }%
208 }

```

`\@gls@loadtree`

```

209 \newcommand*{\@gls@loadtree}{\RequirePackage{glossary-tree}}

```

`notree` This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used.

```

210 \@gls@declareoption{notree}{\renewcommand*{\@gls@loadtree}{}

```

`nostyles` Provide an option to suppress all the predefined styles (in the event that the user has custom styles that are not dependent on the predefined styles).

```

211 \@gls@declareoption{nostyles}{%
212   \renewcommand*{\@gls@loadlong}{}%
213   \renewcommand*{\@gls@loadsuper}{}%
214   \renewcommand*{\@gls@loadlist}{}%
215   \renewcommand*{\@gls@loadtree}{}%
216   \let\@glossary@default@style\relax
217 }

```

`postdescription` The description terminator is given by `\glspostdescription` (except for the 3 and 4 column styles). This is a full stop by default. The spacefactor is adjusted in case the description ends with an upper case letter. (Patch provided by Michael Pock.)

```

218 \newcommand*{\glspostdescription}{%
219   \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
220 }

```

nopostdot Boolean option to suppress post description dot

```
221 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
222 \glsnopostdotfalse
```

nogroupskip Boolean option to suppress vertical space between groups in the pre-defined styles.

```
223 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
224 \glsnogroupskipfalse
```

ucmark Boolean option to determine whether or not to use upper case in definition of `\glsglossarymark`

```
225 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}

226 \@ifclassloaded{memoir}
227 {%
228   \glsucmarktrue
229 }%
230 {%
231   \glsucmarkfalse
232 }
```

glossaryentry If the `entrycounter` package option has been used, define a counter to number each level 0 entry. This is now defined by an internal command for consistency.

aryentrycounter

```
233 \newcommand*{\@gls@define@glossaryentrycounter}{%
234   \ifglsentrycounter
235     Define the glossaryentry counter if it doesn't already exist.
236     \ifundef\c@glossaryentry
237       {%
238         \ifx\@gls@counterwithin\@empty
239           \newcounter{glossaryentry}%
240         \else
241           \newcounter{glossaryentry}[\@gls@counterwithin]%
242         \fi
243         \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
244       }%
245     \fi
246 }
```

entrycounter Defines a counter that can be used in the standard glossary styles to number each (main) entry. If true, this will define a counter called `glossaryentry`.

```
247 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
248 \glsentrycounterfalse
```

counterwithin This option can be used to set a parent counter for `glossaryentry`. This option automatically sets `entrycounter=true`.

```

249 \define@key{glossaries.sty}{counterwithin}{%
250   \renewcommand*{\@gls@counterwithin}{#1}%
251   \glentrycountertrue
252   \@gls@define@glossaryentrycounter
253 }

s@counterwithin The default value is no parent counter:
254 \newcommand*{\@gls@counterwithin}{}

lossarysubentry If the subentrycounter package option has been used, define a counter to number each level 1
entry. This is now defined by an internal command for consistency.

subentrycounter
255 \newcommand{\@gls@define@glossarysubentrycounter}{%
    Check if counter already defined.
256   \ifundef\c@glossarysubentry
257   {%
258     \ifglssubentrycounter
259     \ifglentrycounter
260     \newcounter{glossarysubentry}[glossaryentry]%
261     \else
262     \newcounter{glossarysubentry}%
263     \fi
    As with \theHglossaryentry, this starts with \currentglossary. to help avoid duplicate
    hyper targets.
264     \def\theHglossarysubentry{\currentglossary.\currentglssubentry.\theglossarysubentry}%
265     \fi
266   }%
267   {}%
268 }

subentrycounter Define a counter that can be used in the standard glossary styles to number each level 1 entry.
If true, this will define a counter called glossarysubentry.
269 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
270 \glssubentrycounterfalse

efault@sorttype Initialise default sort for \printnoidxglossary
271 \newcommand*{\@glo@default@sorttype}{standard}

sort Define the sort method: sort=standard (default), sort=def (order of definition) or sort=use (or-
der of use). If no indexing required, use sort=none.
272 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
273   \renewcommand*{\@glo@default@sorttype}{#1}%
274   \csname @gls@setupsort@#1\endcsname
275 }

```

prestandardsort

```
\glsprestandardsort{<sort cs>}{<type>}{<label>}
```

Allow user to hook into sort mechanism. The first argument *<sort cs>* is the temporary control sequence containing the sort value before it has been sanitized and had *makeindex/xindy* special characters escaped.

```
276 \newcommand*{\glsprestandardsort}[3]{%
277   \glsdosanitizesort
278 }
```

check@sortallowed

```
279 \newcommand*{\@glo@check@sortallowed}[1]{}
```

upsort@standard

Set up the macros for default sorting.

```
280 \newcommand*{\@gls@setupsort@standard}{%
```

Store entry information when it's defined.

```
281   \def\do@glo@storeentry{\@glo@storeentry}%
```

No count register required for standard sort.

```
282   \def\@gls@defsortcount##1{}%
```

Sort according to sort key (*\@glo@sort*) if provided otherwise sort according to the entry's name (*\@glo@name*). (First argument glossary type, second argument entry label.)

```
283   \def\@gls@defsort##1##2{%
```

```
284     \ifx\@glo@sort\@gls@defaultsort
```

```
285       \let\@glo@sort\@glo@name
```

```
286     \fi
```

```
287     \let\glsdosanitizesort\@gls@sanitizesort
```

```
288     \glsprestandardsort{\@glo@sort}{##1}{##2}%
```

```
289     \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
```

```
290   }%
```

Don't need to do anything when the entry is used.

```
291   \def\@gls@setsort##1{}%
```

This sort option is allowed with *\makeglossaries* and *\makenoidxglossaries*.

```
292   \let\@glo@check@sortallowed\@gobble
```

```
293 }
```

Set standard sort as the default:

```
294 \@gls@setupsort@standard
```

lssortnumberfmt

Format the number used as the sort key by *sort=def* and *sort=use*. Defaults to six digit numbering.

```
295 \newcommand*{\glssortnumberfmt}[1]{%
```

```
296   \ifnum#1<100000 0\fi
```

```
297   \ifnum#1<10000 0\fi
```

```
298   \ifnum#1<1000 0\fi
```

```
299   \ifnum#1<100 0\fi
```

```

300 \ifnum#1<10 0\fi
301 \number#1%
302 }

s@setupsort@def Set up the macros for order of definition sorting.
303 \newcommand*{\@gls@setupsort@def}{%
  Store entry information when it's defined.
304 \def\do@glo@storeentry{\@glo@storeentry}%
  Defined count register associated with the glossary.
305 \def\@gls@defsortcount##1{%
306 \expandafter\global
307 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
308 }%
  Increment count register associated with the glossary and use as the sort key.
309 \def\@gls@defsort##1##2{%
  It may be that the sort order was changed after the glossary was defined, so check if the count
  register has been defined.
310 \ifcsundef{glossary@##1@sortcount}%
311 {\@gls@defsortcount{##1}}%
312 {}%
313 \expandafter\global\expandafter
314 \advance\csname glossary@##1@sortcount\endcsname by 1\relax
315 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
316 \expandafter\glssortnumberfmt
317 {\csname glossary@##1@sortcount\endcsname}}%
318 }%
  Don't need to do anything when the entry is used.
319 \def\@gls@setsort##1{%
  This sort option is allowed with \makeglossaries and \makenoidxglossaries.
320 \let\@glo@check@sortallowed\@gobble
321 }

s@setupsort@use Set up the macros for order of use sorting.
322 \newcommand*{\@gls@setupsort@use}{%
  Don't store entry information when it's defined.
323 \let\do@glo@storeentry\@gobble
  Defined count register associated with the glossary.
324 \def\@gls@defsortcount##1{%
325 \expandafter\global
326 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
327 }%
  Initialise the sort key to empty.
328 \def\@gls@defsort##1##2{%
329 \expandafter\gdef\csname glo@##2@sort\endcsname{%
330 }%

```

If the sort key hasn't been set, increment the counter associated with the glossary and set the sort key.

```
331 \def\@gls@setsort##1{%
```

Get the parent, if one exists

```
332 \edef\@glo@parent{\csname glo@##1@parent\endcsname}%
```

Set the information for the parent entry if not already done.

```
333 \ifx\@glo@parent\@empty
```

```
334 \else
```

```
335 \expandafter\@gls@setsort\expandafter{\@glo@parent}%
```

```
336 \fi
```

Set index information for this entry

```
337 \edef\@glo@type{\csname glo@##1@type\endcsname}%
```

```
338 \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
```

```
339 \ifx\@gls@tmp\@empty
```

```
340 \expandafter\global\expandafter
```

```
341 \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
```

```
342 \expandafter\protected@xdef\csname glo@##1@sort\endcsname{%
```

```
343 \expandafter\glssortnumberfmt
```

```
344 {\csname glossary@\@glo@type @sortcount\endcsname}}%
```

```
345 \@glo@storeentry{##1}%
```

```
346 \fi
```

```
347 }%
```

This sort option is allowed with `\makeglossaries` and `\makenoidxglossaries`.

```
348 \let\@glo@check@sortallowed\@gobble
```

```
349 }
```

`@setupsort@none` Slightly improves efficiency in the event that no indexing is required.

```
350 \newcommand*{\@gls@setupsort@none}{%
```

Don't store entry index information.

```
351 \def\do@glo@storeentry##1{%
```

No count register required for standard sort.

```
352 \def\@gls@defsortcount##1{%
```

Don't modify sort value.

```
353 \def\@gls@defsort##1##2{%
```

```
354 \expandafter\global\expandafter\let\csname glo@##2@sort\endcsname\@glo@sort
```

```
355 }%
```

Don't need to do anything when the entry is used.

```
356 \def\@gls@setsort##1{%
```

This sort option isn't allowed with `\makeglossaries` or `\makenoidxglossaries`.

```
357 \renewcommand\@glo@check@sortallowed[1]{\PackageError{glossaries}
```

```
358 {Option sort=none not allowed with \string##1}%
```

```
359 {(Use sort=def instead)}}%
```

```
360 }
```


`\glsdefmain` Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`. The default extensions conflict if used with doc, so provide different extensions if doc loaded. (If these extensions are inappropriate, use `nomain` and manually define the main glossary with the desired extensions.)

```
361 \newcommand*{\glsdefmain}{%
362   \if@gls@docloaded
363     \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
364   \else
365     \newglossary{main}{gls}{glo}{\glossaryname}%
366   \fi
```

Define hook to set the toc title when translator is in use.

```
367 \newcommand*{\gls@tr@set@main@toctitle}{%
368   \translatelet{\glossarytoctitle}{Glossary}%
369 }%
370 }
```

Keep track of the default glossary. This is initialised to the main glossary, but can be changed if for some reason you want to make a secondary glossary the main glossary. This affects any commands that can optionally take a glossary name as an argument (or as the value of the type key in a key-value list). This was mainly done so that `\loadglsentries` can temporarily change `\glsdefaulttype` while it loads a file containing new glossary entries (see [section 1.10](#)).

`\glsdefaulttype`

```
371 \newcommand*{\glsdefaulttype}{main}
```

Keep track of which glossary the acronyms are in. This is initialised to `\glsdefaulttype`, but is changed by the acronym package option.

`\acronymtype`

```
372 \newcommand*{\acronymtype}{\glsdefaulttype}
```

`nomain` The `nomain` option suppress the creation of the main glossary.

```
373 \@gls@declareoption{nomain}{%
374   \let\glsdefaulttype\relax
375   \renewcommand*{\glsdefmain}{}%
376 }
```

`acronym` The `acronym` option sets an associated conditional which is used in [section 1.17](#) to determine whether or not to define a separate glossary for acronyms.

```
377 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
378   \ifglsacronym
379     \renewcommand{\@gls@do@acronymsdef}{%
380       \DeclareAcronymList{acronym}%
381       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
382       \renewcommand*{\acronymtype}{acronym}%
383     }
```

Define hook to set the toc title when translator is in use.

```

383     \newcommand*{\gls@tr@set@acronym@toctitle}{%
384         \translatelet{\glossarytoctitle}{Acronyms}%
385     }%
386 }%
387 \else
388     \let\@gls@do@acronymsdef\relax
389 \fi
390 }

```

`\printacronyms` Define `\printacronyms` at the start of the document if acronym is set and compatibility mode isn't on and `\printacronyms` hasn't already been defined.

```

391 \AtBeginDocument{%
392     \ifglsacronym
393     \ifbool{glscompatible-3.07}%
394     {}%
395     {%
396         \providecommand*{\printacronyms}[1][1]{%
397             \printglossary[type=\acronymtype,#1]}%
398     }%
399 \fi
400 }

```

`@do@acronymsdef` Set default value

```

401 \newcommand*{\@gls@do@acronymsdef}{}

```

`acronyms` Provide a synonym for `acronym=true` that can be passed via the document class options.

```

402 \@gls@declareoption{acronyms}{%
403     \glsacronymtrue

404     \def\@gls@do@acronymsdef{%
405         \DeclareAcronymList{acronym}%
406         \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
407         \renewcommand*{\acronymtype}{acronym}%

```

Define hook to set the toc title when translator is in use.

```

408     \newcommand*{\gls@tr@set@acronym@toctitle}{%
409         \translatelet{\glossarytoctitle}{Acronyms}%
410     }%
411 }%
412 }

```

`glsacronymlists` Comma-separated list of glossary labels indicating which glossaries contain acronyms. Note that `\SetAcronymStyle` must be used after adding labels to this macro.

```

413 \newcommand*{\@glsacronymlists}{}

```

`dtoacronymlists`

```

414 \newcommand*{\@addtoacronymlists}[1]{%

```

```

415 \ifx\@glsacronymlists\@empty
416   \protected@xdef\@glsacronymlists{#1}%
417 \else
418   \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
419 \fi
420 }

```

DeclareAcronymList Identifies the named glossary as a list of acronyms and adds to the list. (Doesn't check if the glossary exists, but checks if label already in list. Use `\SetAcronymStyle` after identifying all the acronym lists.)

```

421 \newcommand*\DeclareAcronymList}[1]{%
422   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
423 }

```

glsIfListOfAcronyms

```
\glsIfListOfAcronyms{<label>}{<true part>}{<false part>}
```

Determines if the glossary with the given label has been identified as being a list of acronyms.

```

424 \newcommand{\glsIfListOfAcronyms}[1]{%
425   \edef\@do@gls@islistofacronyms{%
426     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
427   \@do@gls@islistofacronyms
428 }

```

Internal command requires label and list to be expanded:

```

429 \newcommand{\@gls@islistofacronyms}[4]{%
430   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
431     \def\@gls@before{##1}\def\@gls@after{##2}}%
432   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
433   \ifx\@gls@after\@nnil

```

Not found

```

434     #4%
435   \else

```

Found

```

436     #3%
437   \fi
438 }

```

glsisacronymlist Convenient boolean.

```
439 \newif\if@glsisacronymlist
```

glscheckisacronymlist Sets the above boolean if argument is a label representing a list of acronyms.

```

440 \newcommand*\glscheckisacronymlist}[1]{%
441   \glsIfListOfAcronyms{#1}%
442   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
443 }

```

SetAcronymLists Sets the “list of acronyms” list. Argument must be a comma-separated list of glossary labels. (Doesn’t check at this point if the glossaries exists.)

```
444 \newcommand*{\SetAcronymLists}[1]{%
445   \renewcommand*{\@glsacronymlists}{#1}%
446 }
```

acronymlists

```
447 \define@key{glossaries.sty}{acronymlists}{%
448   \DeclareAcronymList{#1}%
449 }
```

The default counter associated with the numbers in the glossary is stored in `\glscounter`. This is initialised to the page counter. This is used as the default counter when a new glossary is defined, unless a different counter is specified in the optional argument to `\newglossary` (see [section 1.6](#)).

\glscounter

```
450 \newcommand{\glscounter}{page}
```

counter The counter option changes the default counter. (This just redefines `\glscounter`.)

```
451 \define@key{glossaries.sty}{counter}{%
452   \renewcommand*{\glscounter}{#1}%
453 }
```

gls@nohyperlist

```
454 \newcommand*{\@gls@nohyperlist}{}
```

lareNoHyperList

```
455 \newcommand*{\GlsDeclareNoHyperList}[1]{%
456   \ifdefempty\@gls@nohyperlist
457   {%
458     \renewcommand*{\@gls@nohyperlist}{#1}%
459   }%
460   {%
461     \appto\@gls@nohyperlist{,#1}%
462   }%
463 }
```

nohypertypes

```
464 \define@key{glossaries.sty}{nohypertypes}{%
465   \GlsDeclareNoHyperList{#1}%
466 }
```

ossariesWarning Prints a warning message.

```
467 \newcommand*{\GlossariesWarning}[1]{%
468   \PackageWarning{glossaries}{#1}%
469 }
```

esWarningNoLine Prints a warning message without the line number.

```

470 \newcommand*\GlossariesWarningNoLine}[1]{%
471   \PackageWarningNoLine{glossaries}{#1}%
472 }

```

tentrieswarning Warn user that sorting may take a long time. This is actually an informational message rather than a warning so just use \typeout.

```

473 \newcommand*\glosortentrieswarning{%
474   \typeout{Using TeX to sort glossary entries---this may
475   take a while}%
476 }

```

nowarn Define package option to suppress warnings

```

477 \@gls@declareoption{nowarn}{%
478   \if@gls@debug
479     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
480   \else
481     \renewcommand*\GlossariesWarning}[1]{}%
482     \renewcommand*\GlossariesWarningNoLine}[1]{}%
483     \renewcommand*\glosortentrieswarning{}%
484     \renewcommand*\@gls@missinglang@warn}[2]{}%
485   \fi
486 }

```

issinglang@warn Missing language warning.

```

487 \newcommand*\@gls@missinglang@warn}[2]{%
488   \PackageWarningNoLine{glossaries}%
489   {No language module detected for '#1'.\MessageBreak
490   Language modules need to be installed separately.\MessageBreak
491   Please check on CTAN for a bundle called\MessageBreak
492   'glossaries-#2' or similar}%
493 }

```

nolangwarn Suppress warning if language support not found.

```

494 \@gls@declareoption{nolangwarn}{%
495   \renewcommand*\@gls@missinglang@warn}[2]{}%
496 }

```

nonglossdefined Issue a warning if overriding \printglossary

```

497 \newcommand*\@gls@warnonglossdefined{%
498   \GlossariesWarning{Overriding \string\printglossary}%
499 }

```

theglossdefined Issue a warning if overriding theglossary

```

500 \newcommand*\@gls@warnontheglossdefined{%
501   \GlossariesWarning{Overriding 'theglossary' environment}%
502 }

```

noredefwarn Suppress warning on redefinition of \printglossary

```

503 \@gls@declareoption{noredefwarn}{%
504   \renewcommand*{\@gls@warnonglossdefined}{}%
505   \renewcommand*{\@gls@warnontheGLOSSdefined}{}%
506 }
```

As from version 3.08a, the only information written to the external glossary files are the label and sort values. Therefore, now, the only sanitize option that makes sense is the one for the sort key. so the sanitize option is now deprecated and there is only a sanitizesort option.

ls@sanitizedesc

```

507 \newcommand*{\@gls@sanitizedesc}{%
508 }
```

setexpandfield `\glssetexpandfield{<field>}`

Sets field to always expand.

```

509 \newcommand*{\glssetexpandfield}[1]{%
510   \csdef{gls@assign@#1@field}##1##2{%
511     \@gls@expand@field{##1}{#1}{##2}%
512   }%
513 }
```

setnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```

514 \newcommand*{\glssetnoexpandfield}[1]{%
515   \csdef{gls@assign@#1@field}##1##2{%
516     \@gls@noexpand@field{##1}{#1}{##2}%
517   }%
518 }
```

sign@type@field The type must always be expandable.

```

519 \glssetexpandfield{type}
```

sign@desc@field The description is not expanded by default:

```

520 \glssetnoexpandfield{desc}
```

escplural@field

```

521 \glssetnoexpandfield{descplural}
```

ls@sanitizename

```

522 \newcommand*{\@gls@sanitizename}{}
```

sign@name@field Don't expand name by default.

```

523 \glssetnoexpandfield{name}
```

@sanitizesymbol

```
524 \newcommand*{\@gls@sanitizesymbol}{}
```

gn@symbol@field Don't expand symbol by default.

```
525 \glssetnoexpandfield{symbol}
```

bolplural@field

```
526 \glssetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```
527 \newcommand*{\@gls@sanitizesort}{%  
528   \ifglssanitizesort  
529     \@gls@sanitizesort  
530   \else  
531     \@gls@nosanitizesort  
532   \fi  
533 }
```

ls@sanitizesort

```
534 \newcommand*{\@gls@sanitizesort{%  
535   \@onelevel@sanitize\@glo@sort  
536 }
```

@nosanitizesort

```
537 \newcommand*{\@gls@nosanitizesort}{}
```

dx@sanitizesort Remove braces around first character (if present) before sanitizing.

```
538 \newcommand*{\@gls@noidx@sanitizesort{%  
539   \ifdefvoid\@glo@sort  
540   }{%  
541   {%  
542     \expandafter\@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort  
543   }%  
544 }  
545 \def\@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%  
546   \def\@glo@sort{#1#2}%  
547   \@onelevel@sanitize\@glo@sort  
548 }
```

@nosanitizesort

```
549 \newcommand*{\@gls@noidx@nosanitizesort}{%  
550   \ifdefvoid\@glo@sort  
551   }{%  
552   {%  
553     \expandafter\@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort  
554   }%
```

```

555 }
556 \def\@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
557   \bgroup
558     \glsnoidxstripaccents
559     \protected@xdef\@glo@sort{#1#2}%
560   \egroup
561   \let\@glo@sort\@glo@sort
562 }

```

`idxstripaccents` This strips accents by redefining the standard accent commands to just do their argument. (This will be localised since `\glsnoidxstripaccents` is used within a group.) Anything outside this standard set really shouldn't be using `\makenoidxglossaries`. It's much better to use `xindy` or `bib2gls` with the correct language setting.

```

563 \newcommand*\glsnoidxstripaccents{%
564   \let\IeC\@firstofone
565   \let\add@accent\@secondoftwo
566   \let\@text@composite\@secondoftwo
567   \let\@tabacckludge\@secondoftwo
568   \expandafter\def\csname \encodingdefault-cmd\endcsname##1##2##3{##3}%
569   \expandafter\def\csname OT1-cmd\endcsname##1##2##3{##3}%
570   \expandafter\def\csname T1-cmd\endcsname##1##2##3{##3}%
571   \expandafter\def\csname PD1-cmd\endcsname##1##2##3{##3}%
572   \let\'\@firstofone
573   \let\'\@firstofone
574   \let\~\@firstofone
575   \let\"@firstofone
576   \let\u\@firstofone
577   \let\t\@firstofone
578   \let\d\@firstofone
579   \let\r\@firstofone
580   \let=\@firstofone
581   \let\.\@firstofone
582   \let\~\@firstofone
583   \let\v\@firstofone
584   \let\H\@firstofone
585   \let\c\@firstofone
586   \let\b\@firstofone

587   \let\a\@secondoftwo
588   \def\AE{AE}%
589   \def\ae{ae}%
590   \def\OE{OE}%
591   \def\oe{oe}%
592   \def\AA{AA}%
593   \def\aa{aa}%
594   \def\L{L}%
595   \def\l{l}%
596   \def\O{O}%
597   \def\o{o}%

```



```

598 \def\SS{SS}%
599 \def\ss{ss}%
600 \def\th{th}%

601 \def\TH{TH}%
602 \def\dh{dh}%
603 \def\DH{DH}%
604 }

```

Need to check if the LaTeX kernel is at least version 2019/10/01 as that changes the way that UTF-8 characters expand.

```

605 \@ifl@t@r\fmtversion{2019/10/01}
606 {%
607 \appto\glsnoidxstripaccents{\let\UTFviii@two@octets\UTFviii@two@octets@combine}%
608 }
609 {}

```

Before defining the sanitize package option, The key-value list for the sanitize value needs to be defined. These are all boolean keys. If they are not given a value, assume true.

```

610 \define@boolkey[glS]{sanitize}{description}[true]{%
611 \GlossariesWarning{sanitize={description} package option deprecated}%
612 \ifglS@sanitize@description
613 \glSsetnoexpandfield{desc}%
614 \glSsetnoexpandfield{descplural}%
615 \else
616 \glSsetexpandfield{desc}%
617 \glSsetexpandfield{descplural}%
618 \fi
619 }

620 \define@boolkey[glS]{sanitize}{name}[true]{%
621 \GlossariesWarning{sanitize={name} package option deprecated}%
622 \ifglS@sanitize@name
623 \glSsetnoexpandfield{name}%
624 \else
625 \glSsetexpandfield{name}%
626 \fi
627 }

628 \define@boolkey[glS]{sanitize}{symbol}[true]{%
629 \GlossariesWarning{sanitize={symbol} package option deprecated}%
630 \ifglS@sanitize@symbol
631 \glSsetnoexpandfield{symbol}%
632 \glSsetnoexpandfield{symbolplural}%
633 \else
634 \glSsetexpandfield{symbol}%
635 \glSsetexpandfield{symbolplural}%
636 \fi
637 }

```

sanitizesort

```
638 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
639   \ifglssanitizesort
640     \glsssetnoexpandfield{sortvalue}%
641     \renewcommand*{\@gls@noidx@setsanitizesort}{%
642       \glssanitizesorttrue
643       \glsssetnoexpandfield{sortvalue}%
644     }%
645   \else
646     \glsssetexpandfield{sortvalue}%
647     \renewcommand*{\@gls@noidx@setsanitizesort}{%
648       \glssanitizesortfalse
649       \glsssetexpandfield{sortvalue}%
650     }%
651   \fi
652 }
```

Default setting:

```
653 \glssanitizesorttrue
654 \glsssetnoexpandfield{sortvalue}%
```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```
655 \newcommand*{\@gls@noidx@setsanitizesort}{%
656   \glssanitizesortfalse
657   \glsssetexpandfield{sortvalue}%
658 }

659 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
660   \setbool{glssanitizesort}{#1}%
661   \ifglssanitizesort
662     \glsssetnoexpandfield{sortvalue}%
663   \else
664     \glsssetexpandfield{sortvalue}%
665   \fi
666   \GlossariesWarning{sanitize={sort} package option
667     deprecated. Use sanitizesort instead}%
668 }
```

sanitize

```
669 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
670   \ifthenelse{\equal{#1}{none}}{%
671     {%
672       \GlossariesWarning{sanitize package option deprecated}%
673       \glsssetexpandfield{name}%
674       \glsssetexpandfield{symbol}%
675       \glsssetexpandfield{symbolplural}%
676       \glsssetexpandfield{desc}%
677       \glsssetexpandfield{descplural}%
678     }%
679   }
```

```

679  {%
680    \setkeys[glS]{sanitize}{#1}%
681  }%
682 }

\ifglstranslate  As from version 3.13a, the translator package option is a choice rather than boolean option
                  so now need to define conditional:
683 \newif\ifglstranslate

otranslatorhook  \@glS@notranslatorhook has been removed.

s@usetranslator
684 \newcommand*\@glS@usetranslator{%
    polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
    polyglossia as well.
685   \@ifpackageloaded{polyglossia}%
686   {%
687     \let\glSifusetranslator\@secondoftwo
688   }%
689   {%
690     \@ifpackageloaded{babel}%
691     {%
692       \IfFileExists{translator.sty}%
693       {%
694         \RequirePackage{translator}%
695         \let\glSifusetranslator\@firstoftwo
696       }%
697     }%
698   }%
699   {}%
700 }%
701 }

dtranslatordict  Checks if given translator dictionary has been loaded.
702 \newcommand{\glSifusedtranslatorDict}[3]{%
703   \glSifusetranslator
704   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
705   {#3}%
706 }

notranslate  Provide a synonym for translate=false that can be passed via the document class.
707 \@glS@declareoption{notranslate}{%
708   \glstranslatefalse
709   \let\@glS@usetranslator\relax
710   \let\glSifusetranslator\@secondoftwo
711 }

```

translate Define translate option. If false don't set up multi-lingual support.

```

712 \define@choicekey{glossaries.sty}{translate}%
713   [\gls@translate@val\gls@translate@nr]%
714   {true,false,babel}[true]%
715   {%
716     \ifcase\gls@translate@nr\relax
717       \glstranslatetrue
718       \renewcommand*\@gls@usetranslator{%
719         \@ifpackageloaded{polyglossia}%
720         {%
721           \let\glsifusetranslator\@secondoftwo
722         }%
723         {%
724           \@ifpackageloaded{babel}%
725           {%
726             \IfFileExists{translator.sty}%
727             {%
728               \RequirePackage{translator}%
729               \let\glsifusetranslator\@firstoftwo
730             }%
731             {}%
732           }%
733           {}%
734         }%
735       }%
736     \or
737       \glstranslatefalse
738       \let\@gls@usetranslator\relax
739       \let\glsifusetranslator\@secondoftwo
740     \or
741       \glstranslatetrue
742       \let\@gls@usetranslator\relax
743       \let\glsifusetranslator\@secondoftwo
744     \fi
745   }

```

Set the default value:

```

746 \glstranslatefalse
747 \let\glsifusetranslator\@secondoftwo
748 \@ifpackageloaded{translator}%
749 {%
750   \glstranslatetrue
751   \let\glsifusetranslator\@firstoftwo
752 }%
753 {%
754   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
755   {
756     \@ifpackageloaded{\gls@thissty}%
757     {%

```

```

758     \glstranslatetrue
759     \@endfortrue
760 }%
761 {}%
762 }
763 }

```

indexonlyfirst Set whether to only index on first use.

```

764 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
765 \glindexonlyfirstfalse

```

hyperfirst Set whether or not terms should have a hyperlink on first use.

```

766 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
767 \glshyperfirsttrue

```

gls@setacrstyle Keep track of whether an acronym style has been set (for the benefit of `\setupglossaries`):

```

768 \newcommand*{\@gls@setacrstyle}{}

```

footnote Set the long form of the acronym in footnote on first use.

```

769 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}%
770 \ifbool{glsacrdescription}%
771 {}%
772 {}%
773 \renewcommand*{\@gls@sanitizedesc}{}%
774 }%
775 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
776 }

```

description Allow acronyms to have a description (needs to be set using the description key in the optional argument of `\newacronym`).

```

777 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}%
778 \renewcommand*{\@gls@sanitizesymbol}{}%
779 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
780 }

```

smallcaps Define `\newacronym` to set the short form in small capitals.

```

781 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}%
782 \renewcommand*{\@gls@sanitizesymbol}{}%
783 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
784 }

```

smaller Define `\newacronym` to set the short form using `\smaller` which obviously needs to be defined by loading the appropriate package.

```

785 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{}%
786 \renewcommand*{\@gls@sanitizesymbol}{}%
787 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
788 }

```

dua Define `\newacronym` to always use the long forms (i.e. don't use acronyms)

```

789 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
790   \renewcommand*{\@gls@sanitizesymbol}{}%
791   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
792 }

```

shortcuts Define acronym shortcuts.

```

793 \define@boolkey{glossaries.sty}[glsacr]{shortcuts}[true]{%

```

\glsorder Stores the glossary ordering. This may either be “word” or “letter”. This passes the relevant information to `makeglossaries`. The default is word ordering.

```

794 \newcommand*{\glsorder}{word}

```

\@glsorder The ordering information is written to the auxiliary file for `makeglossaries`, so ignore the auxiliary information.

```

795 \newcommand*{\@glsorder}[1]{}

```

order

```

796 \define@choicekey{glossaries.sty}{order}{word,letter}{%
797   \def\glsorder{#1}}

```

\ifglxindy Provide boolean to determine whether `xindy` or `makeindex` will be used to sort the glossaries.

```

798 \newif\ifglxindy

```

The default is `makeindex`:

```

799 \glxindyfalse

```

makeindex Define package option to specify that `makeindex` will be used to sort the glossaries:

```

800 \@gls@declareoption{makeindex}{\glxindyfalse}

```

The `xindy` package option may have a value which in turn can be a key=value list. First define the keys for this sub-list. The boolean `glsnumbers` determines whether to automatically add the `glsnumbers` letter group.

```

801 \define@boolkey[gl]{xindy}{glsnumbers}[true]{}
802 \gls@xindy@glnumberstrue

```

y@main@language Define what language to use for each glossary type (if a language is not defined for a particular glossary type the language specified for the main glossary is used.)

```

803 \def\@xdy@main@language{\language}%

```

Define key to set the language

```

804 \define@key[gl]{xindy}{language}{\def\@xdy@main@language{#1}}

```

`\gls@codepage` Define the code page. If `\inputencodingname` is defined use that, otherwise have initialise with no codepage.

```

805 \ifcsundef{inputencodingname}{%
806   \def\gls@codepage{}}{%
807   \def\gls@codepage{\inputencodingname}
808 }

Define a key to set the code page.
809 \define@key[gls]{xindy}{codepage}{\def\gls@codepage{#1}}

```

`xindy` Define package option to specify that xindy will be used to sort the glossaries:

```

810 \define@key{glossaries.sty}{xindy}[]{%
811   \glsxindytrue
812   \setkeys[gls]{xindy}{#1}%
813 }

```

`xindygloss` Provide a synonym for xindy that can be passed via the document class options.

```

814 \@gls@declareoption{xindygloss}{%
815   \glsxindytrue
816 }

```

`xindynoglsnumbers` Provide a synonym for `xindy=glsnumbers=false` that can be passed via the document class options.

```

817 \@gls@declareoption{xindynoglsnumbers}{%
818   \glsxindytrue
819   \gls@xindy@glsnumbersfalse
820 }

```

`omakeglossaries`

```

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

```

`isablemakegloss` Provide a way of disabling `\makeglossaries`. For example, if a class or package explicitly uses `\makeglossaries`. This is a valueless option to allow it to be passed through the document class option list.

```

822 \@gls@declareoption{isablemakegloss}{%
823   \ifdefequal\makeglossaries\@no@makeglossaries
824   {%
825     \GlossariesWarning{Option ‘isablemakegloss’ has no effect
826       (\string\makenoidxglossaries\space has already been used)}%
827   }%
828   {%
829     \ifdefequal\@makeglossary\@gobble
830     {%
831       \GlossariesWarning{Option ‘isablemakegloss’ has no effect
832         (\string\makeglossaries\space has already been used)}%
833     }%
834     {%
835       \renewcommand{\@domakeglossaries}[1]{%

```

```

836     \PackageInfo{glossaries}{\string\makeglossaries\space and
837     \string\makenoidxglossaries\space have been disabled}%
838   }%
839 }%
840 }%
841 }

```

`restoremakegloss` Cancel the effect of `disablemakegloss`.

```

842 \@gls@declareoption{restoremakegloss}{%
843   \ifdefequal\makeglossaries\@no@makeglossaries
844   {%
845     \GlossariesWarning{Option ‘restoremakegloss’ has no effect
846     (\string\makenoidxglossaries\space has already been used)}%
847   }%
848   {%
849     \ifdefequal\@makeglossary\@gobble
850     {%
851       \GlossariesWarning{Option ‘restoremakegloss’ has no effect
852       (\string\makeglossaries\space has already been used)}%
853     }%
854     {%
855       \PackageInfo{glossaries}{\string\makeglossaries\space and
856       \string\makenoidxglossaries\space have been restored}%
857       \let\@domakeglossaries\@firstofone
858     }%
859   }%
860 }

```

`write@glslabels`

```

861 \newcommand*{\@do@write@glslabels}{%
862   \AtEndDocument{\@do@write@glslabels}%
863   \let\@do@write@glslabels\relax
864 }

```

`write@glslabels`

```

865 \newcommand*{\@do@write@glslabels}{%
866   \newwrite\@gls@labelsfile
867   \immediate\openout\@gls@labelsfile=\jobname.glslabels
868   \forallglsentries[\@gls@types,\@ignored@glossaries]{\@glsentry}%
869   {\ifdefempty{\@glsentry}{\immediate\write\@gls@labelsfile{\@glsentry}}}%
870   \immediate\closeout\@gls@labelsfile
871 }

```

`writeglslabels` This option will write all entry labels (including those in ignored glossaries) to the file `\jobname.glslabels`. This file may be used by text editors for label auto-completion.

```

872 \@gls@declareoption{writeglslabels}{\@do@write@glslabels}

```

`\ifglsautomake`

```

873 \newif\ifglsautomake

```


gls@automake@nr

```
874 \newcommand{\gls@automake@nr}{1}
```

automake If this setting is on, automatically run **makeindex/xindy** at the end of the document. Must be used with `\makeglossaries`. Default is false. As from v4.42, this is now a choice rather than boolean key.

```
875 \define@choicekey{glossaries.sty}{automake}%
876   [\gls@automake@val\gls@automake@nr]{true,false,immediate}[true]{%
877   \ifnum\gls@automake@nr=1\relax
878     \glsautomakefalse
879   \else
880     \glsautomaketrue
881   \fi
882   \ifglsautomake
883     \renewcommand*{\@gls@doautomake}{%
884       \PackageError{glossaries}{You must use
885       \string\makeglossaries\space with automake=true}
886       {%
887         Either remove the automake=true setting or
888         add \string\makeglossaries\space to your document preamble.%
889       }%
890     }%
891   \else
892     \renewcommand*{\@gls@doautomake}{}%
893   \fi
894 }
895 \glsautomakefalse
```

@gls@doautomake

```
896 \newcommand*{\@gls@doautomake}{}
897 \AtEndDocument{\@gls@doautomake}
```

savewrites The savewrites package option is provided to save on the number of write registers.

```
898 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
899   \ifglssavewrites
900     \renewcommand*{\glswritefiles}{\@glswritefiles}%
901   \else
902     \let\glswritefiles\@empty
903   \fi
904 }
```

Set default:

```
905 \glssavewritesfalse
906 \let\glswritefiles\@empty
```

compatible-3.07

```
907 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{}
908 \boolfalse{glscompatible-3.07}
```

compatible-2.07

```
909 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
    Also set 3.07 compatibility if this option is set.
910   \ifbool{glscompatible-2.07}%
911   {%
912     \booltrue{glscompatible-3.07}%
913   }%
914   {}%
915 }
916 \boolfalse{glscompatible-2.07}
```

al@makeglossary Store the original definition.

```
917 \let\gls@original@makeglossary\makeglossary
```

iginal@glossary Store the original definition.

```
918 \let\gls@original@glossary\glossary
```

\makeglossary The \makeglossary command is redefined to be identical to \makeglossaries. (This is done partly to reinforce the message that you must either use \@makeglossary for all the glossaries or for none of them, but is also a legacy from the old glossary package.)

```
919 \def\makeglossary{%
920   \GlossariesWarning{Use of \string\makeglossary\space with
921   glossaries.sty is \MessageBreak deprecated. Use \string\makeglossaries\space
922   instead. If you \MessageBreak need the original definition of
923   \string\makeglossary\space use \MessageBreak the package options
924   kernelglossredefs=false (to \MessageBreak restore the former definition of
925   \string\makeglossary) and \MessageBreak nomain (if the file extensions cause a
926   conflict)}}%
927   \makeglossaries
928 }
```

erride@glossary

```
929 \newcommand*{\@gls@override@glossary}[1][main]{%
930   \GlossariesWarning{Use of \string\glossary\space with
931   glossaries.sty is deprecated. \MessageBreak Indexing should be performed
932   with the user level \MessageBreak commands, such as \string\gls\space or
933   \string\glsadd. If you need the \MessageBreak original definition of
934   \string\glossary\space use the package \MessageBreak options
935   kernelglossredefs=false (to restore the \MessageBreak former definition of
936   \string\glossary) and nomain (if the \MessageBreak file extensions cause a
937   conflict)}}%
938   \gls@glossary{#1}%
939 }
```

In v4.10, the redefinition of \glossary was removed since it was never intended as a user level command (and wasn't documented in the user manual), however it seems there are packages that have hacked the internal macros used by glossaries and no longer work with

this redefinition removed, so it's been restored in v4.11 but is not used at all by glossaries. (This may be removed or moved to a compatibility mode in future.) As from v4.41, the use of `\glossary` now triggers a warning. The package option `kernelglossredefs=nowarn` may be used to remove the warning, but it's better not to use `\glossary`.

`\glossary`

```
940 \if@gls@docloaded
941 \else
942   \def\glossary{\@gls@override@glossary}
943 \fi
```

`kernelglossredefs` The glossaries package redefines the kernel commands `\makeglossary` and `\glossary` as a legacy action from the former glossary package. In hindsight that wasn't a good idea as it's possible that the glossaries package may need to be used with another class or package that needs these commands. Neither of these commands are documented in the main user manual and their use is not encouraged. The preferred commands are `\makeglossaries` (to open all associated glossary files) and `\gls`, `\glstext` etc or `\glsadd` for indexing.

```
944 \define@choicekey{glossaries.sty}{kernelglossredefs}%
945 [\@gls@debug@val\@gls@debug@nr]{true,false,nowarn}[true]%
946 {%
947   \ifcase\@gls@debug@nr\relax
948   \def\glossary{\@gls@override@glossary}%
949   \def\makeglossary{%
950     \GlossariesWarning{Use of \string\makeglossary\space with
951       glossaries.sty is deprecated. Use \string\makeglossaries\space
952       instead. If you need the original definition of
953       \string\makeglossary\space use the package options
954       kernelglossredefs=false (to prevent redefinition of
955       \string\makeglossary) and nomain (if the file extensions cause a
956       conflict)}}%
957   \makeglossaries
958 }%
959 \or
960   \let\glossary\@gls@original@glossary
961   \let\makeglossary\@gls@original@makeglossary
962 \or
963   \def\makeglossary{\makeglossaries}%
964   \renewcommand*{\@gls@override@glossary}[1][main]{%
965     \@gls@glossary{##1}%
966   }%
967 \fi
968 }
```

`symbols` Create a “symbols” glossary type

```
969 \@gls@declareoption{symbols}{%
970   \let\@gls@do@symbolsdef\@gls@symbolsdef
971 }
```

Default is not to define the symbols glossary:

```
972 \newcommand*{\@gls@do@symbolsdef}{}
```

@gls@symbolsdef

```
973 \newcommand*{\@gls@symbolsdef}{%
974   \newglossary[slg]{symbols}{sls}{slo}{\glssymbolsgroupname}%
975   \newcommand*{\printsymbols}[1][\printglossary[type=symbols,##1]]%
```

Define hook to set the toc title when translator is in use.

```
976   \newcommand*{\gls@tr@set@symbols@toctitle}{%
977     \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
978   }%
979 }%
```

numbers Create a “symbols” glossary type

```
980 \@gls@declareoption{numbers}{%
981   \let\@gls@do@numbersdef\@gls@numbersdef
982 }
```

Default is not to define the numbers glossary:

```
983 \newcommand*{\@gls@do@numbersdef}{}
```

@gls@numbersdef

```
984 \newcommand*{\@gls@numbersdef}{%
985   \newglossary[nlg]{numbers}{nls}{nlo}{\glsnumbersgroupname}%
986   \newcommand*{\printnumbers}[1][\printglossary[type=numbers,##1]]%
```

Define hook to set the toc title when translator is in use.

```
987   \newcommand*{\gls@tr@set@numbers@toctitle}{%
988     \translatelet{\glossarytoctitle}{Numbers (glossaries)}%
989   }%
990 }%
```

index Create an “index” glossary type

```
991 \@gls@declareoption{index}{%
992   \ifx\@gls@do@indexdef\@empty
993     \let\@gls@do@indexdef\@gls@indexdef
994   \fi
995 }
```

noglossaryindex Counteract index if it happens to be globally used in the document class.

```
996 \@gls@declareoption{noglossaryindex}{%
997   \let\@gls@do@indexdef\relax
998 }
```

Default is not to define index glossary:

```
999 \newcommand*{\@gls@do@indexdef}{}
```

```

\@gls@indexdef \indexname isn't set by glossaries.
1000 \newcommand*{\@gls@indexdef}{%
1001   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
1002   \newcommand*{\printindex}[1][\printglossary[type=index,##1]]{%
1003     \newcommand*{\newterm}[2][\%
1004       \newglossaryentry{##2}%
1005       {type={index},name={##2},description={\nopostdesc},##1}}%
1006   \let\@gls@do@indexdef\relax
1007 }%

```

Process package options. First process any options that have been passed via the document class.

```

1008 \@for\CurrentOption :=\@declaredoptions\do{%
1009   \ifx\CurrentOption\@empty
1010   \else
1011     \@expandtwoargs
1012     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
1013     \ifin@
1014       \@use@ption
1015       \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
1016     \fi
1017   \fi
1018 }

```

Now process options passed to the package:

```
1019 \ProcessOptionsX
```

Load backward compatibility stuff:

```
1020 \RequirePackage{glossaries-compatible-307}
```

setupglossaries Provide way to set options after package has been loaded. However, some options must be set before \ProcessOptionsX, so they have to be disabled:

```

1021 \disable@keys{glossaries.sty}{compatible-2.07,%
1022 xindy,xindygloss,xindynoglsnumbers,makeindex,%
1023 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,%
1024 nomain,noglossaryindex}

```

Now define \setupglossaries:

```

1025 \newcommand*{\setupglossaries}[1]{%
1026   \renewcommand*{\@gls@setacrstyle}{}%
1027   \ifglsacrshortcuts
1028     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
1029   \else
1030     \def\@gls@setupshortcuts{%
1031       \ifglsacrshortcuts
1032         \DefineAcronymSynonyms
1033       \fi
1034     }%
1035   \fi
1036   \glsacrshortcutsfalse

```

```

1037 \let\@gls@do@numbersdef\relax
1038 \let\@gls@do@symbolssdef\relax
1039 \let\@gls@do@indexdef\relax
1040 \let\@gls@do@acronymsdef\relax
1041 \ifglseentrycounter
1042   \let\@gls@doentrycounterdef\relax
1043 \else
1044   \let\@gls@doentrycounterdef\@gls@define@glossaryentrycounter
1045 \fi
1046 \ifglssubentrycounter
1047   \let\@gls@dosubentrycounterdef\relax
1048 \else
1049   \let\@gls@dosubentrycounterdef\@gls@define@glossarysubentrycounter
1050 \fi
1051 \setkeys{glossaries.sty}{#1}%
1052 \@gls@setacrstyle
1053 \@gls@setupshortcuts
1054 \@gls@do@acronymsdef
1055 \@gls@do@numbersdef
1056 \@gls@do@symbolssdef
1057 \@gls@do@indexdef
1058 \@gls@doentrycounterdef
1059 \@gls@dosubentrycounterdef
1060 }

```

If chapters are defined and the user has requested the section counter as a package option, `\@chapter` will be modified so that it adds a section. $\langle n \rangle . 0$ target, otherwise entries placed before the first section of a chapter will have undefined links.

The same problem will also occur if a lower sectional unit is used, but this is less likely to happen. If it does, or if you change `\glscounter` to section later, you will have to specify a different counter for the entries that give rise to a name $\{\langle section-level \rangle . \langle n \rangle . 0\}$ non-existent warning (e.g. `\gls[counter=chapter]{label}`).

```

1061 \ifthenelse{\equal{\glscounter}{section}}%
1062 {%
1063   \ifcsundef{chapter}{}%
1064   {%
1065     \let\@gls@old@chapter\@chapter
1066     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
1067       \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}}}%
1068   }%
1069 }%
1070 {}

```

`\onlypremakeg` Some commands only have an effect when used before `\makeglossaries`. So define a list of commands that should be disabled after `\makeglossaries`

```

1071 \newcommand*{\@gls@onlypremakeg}{

```

`\@onlypremakeg` Adds the specified control sequence to the list of commands that must be disabled after

```

\makeglossaries.
1072 \newcommand*{\@onlypremakeg}[1]{%
1073   \ifx\@gls@onlypremakeg\@empty
1074     \def\@gls@onlypremakeg{#1}%
1075   \else
1076     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
1077     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
1078   \fi
1079 }

```

`\le@onlypremakeg` Disable all commands listed in `\@gls@onlypremakeg`

```

1080 \newcommand*{\@disable@onlypremakeg}{%
1081 \@for\@thiscs:=\@gls@onlypremakeg\do{%
1082   \expandafter\@disable@premakecs\@thiscs%
1083 }}

```

`\sable@premakecs` Disables the given command.

```

1084 \newcommand*{\@disable@premakecs}[1]{%
1085   \def#1{\PackageError{glossaries}{\string#1\space may only be
1086     used before \string\makeglossaries}{You can't use
1087     \string#1\space after \string\makeglossaries}}%
1088 }

```

1.3 Predefined Text

Set up default textual tags that are used by this package. Some of the names may already be defined (e.g. by) so `\providecommand` is used.

Main glossary title:

`\glossaryname`

```

1089 \providecommand*\glossaryname{Glossary}

```

The title for the acronym glossary type (which is defined if acronym package option is used) is given by `\acronymname`. If the acronym package option is not used, `\acronymname` won't be used.

`\acronymname`

```

1090 \providecommand*\acronymname{Acronyms}

```

`\glssettocitle` Sets the TOC title for the given glossary.

```

1091 \newcommand*\glssettocitle}[1]{%
1092   \def\glossarytocitle{\csname @glotype@#1@title\endcsname}}

```

The following commands provide text for the headers used by some of the tabular-like glossary styles. Whether or not they get used in the glossary depends on the glossary style.

`\entryname`

```

1093 \providecommand*\entryname{Notation}

```

descriptionname
1094 \providecommand*{\descriptionname}{Description}

\symbolname
1095 \providecommand*{\symbolname}{Symbol}

\pagelistname
1096 \providecommand*{\pagelistname}{Page List}

Labels for makeindex's symbol and number groups:

ymbolsgroupname
1097 \providecommand*{\glssymbolsgroupname}{Symbols}

umbersgroupname
1098 \providecommand*{\glsnumbersgroupname}{Numbers}

glspluralsuffix The default plural is formed by appending \glspluralsuffix to the singular form.
1099 \newcommand*{\glspluralsuffix}{s}

acrpluralsuffix Default plural suffix for acronyms
1100 \newcommand*{\glsacrpluralsuffix}{\glspluralsuffix}

acrpluralsuffix
1101 \newcommand*{\glsupacrpluralsuffix}{\glstextup{\glsacrpluralsuffix}}

\seename
1102 \providecommand*{\seename}{see}

\andname
1103 \providecommand*{\andname}{\&}

Add multi-lingual support. Thanks to everyone who contributed to the translations from both comp.text.tex and via email.

eGlossariesLang
1104 \newcommand*{\RequireGlossariesLang}[1]{%
1105 \@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}%
1106 }

sGlossariesLang
1107 \newcommand*{\ProvidesGlossariesLang}[1]{%
1108 \ProvidesFile{glossaries-#1.ldf}%
1109 }

ssarytocaptions Does nothing if translator hasn't been loaded.
1110 \newcommand*{\addglossarytocaptions}[1]{}

As from v4.12, multilingual support has been split off into independently-maintained language modules.

```
1111 \ifglstranslate
```

Load tracklang

```
1112 \RequirePackage{tracklang}
```

Load translator if required.

```
1113 \@gls@usetranslator
```

If using , \glossaryname should be defined in terms of \translate, but if babel is also loaded, it will redefine \glossaryname whenever the language is set, so override it. (Don't use \addto as doesn't define it.)

```
1114 \@ifpackageloaded{translator}
```

```
1115 {%
```

If the language options have been specified through the document class, then translator can pick them up. If not, translator will default to English and any language option passed to babel won't be detected, so if \trans@languages is just English and \bbl@loaded isn't simply english, then don't use the translator dictionaries.

```
1116 \ifboolexpr
```

```
1117 {
```

```
1118 test {\ifdefstring{\trans@languages}{English}}
```

```
1119 and not
```

```
1120 test {\ifdefstring{\bbl@loaded}{english}}
```

```
1121 }
```

```
1122 {%
```

```
1123 \let\glsifusetranslator\@secondoftwo
```

```
1124 }%
```

```
1125 {%
```

```
1126 \usedictionary{glossaries-dictionary}%
```

```
1127 \renewcommand*{\addglossarytocaptions}[1]{%
```

```
1128 \ifcsundef{captions#1}{}%
```

```
1129 {%
```

```
1130 \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
```

```
1131 \expandafter\toks@\expandafter{\@gls@tmp
```

```
1132 \renewcommand*{\glossaryname}{\translate{Glossary}}}%
```

```
1133 }%
```

```
1134 \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
```

```
1135 }%
```

```
1136 }%
```

```
1137 }%
```

```
1138 }%
```

```
1139 {}%
```

Check for tracked languages

```
1140 \AnyTrackedLanguages
```

```
1141 {%
```

```
1142 \ForEachTrackedDialect{\this@dialect}{%
```

```
1143 \IfTrackedLanguageFileExists{\this@dialect}%
```

```

1144 {glossaries-}% prefix
1145 {.ldf}%
1146 {%
1147 \RequireGlossariesLang{\CurrentTrackedTag}%
1148 }%
1149 {%
1150 \@gls@missinglang@warn\this@dialect\CurrentTrackedLanguage
1151 }%
1152 }%
1153 }%
1154 {}%

if using translator use translator interface.
1155 \glsifusetranslator
1156 {%
1157 \renewcommand*{\glssettoctitle}[1]{%
1158 \ifcsdef{gls@tr@set@#1@toctitle}%
1159 {%
1160 \csuse{gls@tr@set@#1@toctitle}%
1161 }%
1162 {%
1163 \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
1164 }%
1165 }%
1166 \renewcommand*{\glossaryname}{\translate{Glossary}}}%
1167 \renewcommand*{\acronymname}{\translate{Acronyms}}}%
1168 \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
1169 \renewcommand*{\descriptionname}{%
1170 \translate{Description (glossaries)}}}%
1171 \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
1172 \renewcommand*{\pagelistname}{%
1173 \translate{Page List (glossaries)}}}%
1174 \renewcommand*{\glssymbolsgroupname}{%
1175 \translate{Symbols (glossaries)}}}%
1176 \renewcommand*{\glsnumbersgroupname}{%
1177 \translate{Numbers (glossaries)}}}%
1178 }{}%
1179 \fi

```

`\nopostdesc` Provide a means to suppress description terminator for a given entry. (Useful for entries with no description.) Has no effect outside the glossaries.

```
1180 \DeclareRobustCommand*{\nopostdesc}{}
```

`\@nopostdesc` Suppress next description terminator.

```

1181 \newcommand*{\@nopostdesc}{%
1182 \let\org@gls@postdescription\gls@postdescription
1183 \def\gls@postdescription{%
1184 \let\gls@postdescription\org@gls@postdescription}%
1185 }

```

`\@no@post@desc` Used for comparison purposes.

```
1186 \newcommand*{\@no@post@desc}{\nopostdesc}
```

`\glspar` Provide means of having a paragraph break in glossary entries

```
1187 \newcommand{\glspar}{\par}
```

`\setStyleFile` Sets the style file. The relevant extension is appended.

```
1188 \newcommand{\setStyleFile}[1]{%
```

```
1189   \renewcommand*{\gls@istfilebase}{#1}%
```

Just in case `\istfilename` has been modified.

```
1190   \ifglxsindy
```

```
1191     \def\istfilename{\gls@istfilebase.xdy}
```

```
1192   \else
```

```
1193     \def\istfilename{\gls@istfilebase.ist}
```

```
1194   \fi
```

```
1195 }
```

This command only has an effect prior to using `\makeglossaries`.

```
1196 \@onlypremakeg\setStyleFile
```

The name of the `makeindex` or `xindy` style file is given by `\istfilename`. This file is created by `\writeist` (which is used by `\makeglossaries`) so redefining this command will only have an effect if it is done *before* `\makeglossaries`. As from v1.17, use `\setStyleFile` instead of directly redefining `\istfilename`.

`\istfilename`

```
1197 \ifglxsindy
```

```
1198   \def\istfilename{\gls@istfilebase.xdy}
```

```
1199 \else
```

```
1200   \def\istfilename{\gls@istfilebase.ist}
```

```
1201 \fi
```

`gls@istfilebase`

```
1202 \newcommand*{\gls@istfilebase}{\jobname}
```

The `makeglossaries` Perl script picks up this name from the auxiliary file. If the name ends with `.xdy` it calls `xindy` otherwise it calls `makeindex`. Since its not required by \TeX , `\@istfilename` ignores its argument.

`\@istfilename`

```
1203 \newcommand*{\@istfilename}[1]{}
```

This command is the value of the `page_compositor` `makeindex` key. Again, any redefinition of this command must take place *before* `\writeist` otherwise it will have no effect. As from 1.17, use `\glsSetCompositor` instead of directly redefining `\glscompositor`.

`\glscompositor`

```
1204 \newcommand*{\glscompositor}{.}
```

`\lsSetCompositor` Sets the compositor.

```
1205 \newcommand*{\glsSetCompositor}[1]{%
1206   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
1207 \@onlypremakeg\glsSetCompositor
```

(The page compositor is usually defined as a dash when using `makeindex`, but most of the standard counters used by \TeX use a full stop as the compositor, which is why I have used it as the default.) If `xindy` is used `\glscompositor` only affects the `arabic-page-numbers` location class.

`\Alphacompositor` This is only used by `xindy`. It specifies the compositor to use when location numbers are in the form *<letter><compositor><number>*. For example, if `\@glsAlphacompositor` is set to “.” then it allows locations such as A.1 whereas if `\@glsAlphacompositor` is set to “-” then it allows locations such as A-1.

```
1208 \newcommand*{\@glsAlphacompositor}{\glscompositor}
```

`\AlphaCompositor` Sets the alpha compositor.

```
1209 \ifglsxindy
1210   \newcommand*{\glsSetAlphaCompositor}[1]{%
1211     \renewcommand*{\@glsAlphacompositor}{#1}}
1212 \else
1213   \newcommand*{\glsSetAlphaCompositor}[1]{%
1214     \glsnnoxindywarning\glsSetAlphaCompositor}
1215 \fi
```

Can only be used before `\makeglossaries`

```
1216 \@onlypremakeg\glsSetAlphaCompositor
```

`\gls@suffixF` Suffix to use for a two page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```
1217 \newcommand*{\gls@suffixF}{}
```

`\glsSetSuffixF` Sets the suffix to use for a two page list.

```
1218 \newcommand*{\glsSetSuffixF}[1]{%
1219   \renewcommand*{\gls@suffixF}{#1}}
```

Only has an effect when used before `\makeglossaries`

```
1220 \@onlypremakeg\glsSetSuffixF
```

`\gls@suffixFF` Suffix to use for a three page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```
1221 \newcommand*{\gls@suffixFF}{}
```

`\glsSetSuffixFF` Sets the suffix to use for a three page list.

```
1222 \newcommand*{\glsSetSuffixFF}[1]{%
1223   \renewcommand*{\gls@suffixFF}{#1}%
1224 }
```

`glsnumberformat` The command `\glsnumberformat` indicates the default format for the page numbers in the glossary. (Note that this is not the same as `\glossaryentrynumbers`, but applies to individual numbers or groups of numbers within an entry’s associated number list.) If hyperlinks are defined, it will use `\glshypernumber`, otherwise it will simply display its argument “as is”.

```
1225 \ifcsundef{hyperlink}%
1226 {%
1227   \newcommand*{\glsnumberformat}[1]{#1}%
1228 }%
1229 {%
1230   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
1231 }
```

Individual numbers in an entry’s associated number list are delimited using `\delimN` (which corresponds to the `delim_n` makeindex keyword). The default value is a comma followed by a space.

```
\delimN
1232 \newcommand{\delimN}{, }
```

A range of numbers within an entry’s associated number list is delimited using `\delimR` (which corresponds to the `delim_r` makeindex keyword). The default is an en-dash.

```
\delimR
1233 \newcommand{\delimR}{--}
```

The glossary preamble is given by `\glossarypreamble`. This will appear after the glossary sectioning command, and before the `theglossary` environment. It is designed to allow the user to add information pertaining to the glossary (e.g. “page numbers in italic indicate the primary definition”) therefore `\glossarypreamble` shouldn’t be affected by the glossary style. (So if you define your own glossary style, don’t have it change `\glossarypreamble`.) The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use `\printglossary` for each glossary type, instead of `\printglossaries`, and redefine `\glossarypreamble` before each `\printglossary`.

```
lossarypreamble
1234 \newcommand*{\glossarypreamble}{%
1235   \csuse{@glossarypreamble@\currentglossary}%
1236 }
```

```
lossarypreamble \setglossarypreamble[<type>]{<text>}
```

Code provided by Michael Pock.

```
1237 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
1238   \ifglossaryexists{#1}{%
1239     \csgdef{@glossarypreamble@#1}{#2}%
1240   }
```

```

1240 }{%
1241   \GlossariesWarning{%
1242     Glossary ‘#1’ is not defined%
1243   }%
1244 }%
1245 }

```

The glossary postamble is given by `\glossarypostamble`. This is provided to allow the user to add something after the end of the `\glossary` environment (again, this shouldn't be affected by the glossary style). It is, of course, possible to simply add the text after `\printglossary`, but if you only want the postamble to appear after the first glossary, but not after subsequent glossaries, you can do something like:

```

\renewcommand{\glossarypostamble}{For a complete list of terms
see \cite{blah}\gdef\glossarypreamble{}}

```

`\glossarypostamble`

```

1246 \newcommand*{\glossarypostamble}{}

```

`\glossarysection` The sectioning command that starts a glossary is given by `\glossarysection`. (This does not form part of the glossary style, and so should not be changed by a glossary style.) If `\phantomsection` is defined, it uses `\p@glossarysection`, otherwise it uses `\@glossarysection`.

```

1247 \newcommand*{\glossarysection}[2][\@gls@title]{%
1248   \def\@gls@title{#2}%
1249   \ifcsundef{phantomsection}%
1250   {%
1251     \@glossarysection{#1}{#2}%
1252   }%
1253   {%
1254     \p@glossarysection{#1}{#2}%
1255   }%

1256   \glsglossarymark{\glossarytoctitle}%
1257 }

```

`\glsglossarymark` Sets the header mark for the glossary. Takes the glossary short (TOC) title as the argument.

```

1258 \ifcsundef{glossarymark}%
1259 {%
1260   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1261 }%
1262 {%
1263   \@ifclassloaded{memoir}
1264   {%
1265     \newcommand{\glsglossarymark}[1]{%
1266       \ifglsucmark
1267         \markboth{\memUHead{#1}}{\memUHead{#1}}%
1268       \else
1269         \markboth{#1}{#1}%

```

```

1270     \fi
1271   }
1272 }%
1273 {%
1274   \newcommand{\glsglossarymark}[1]{%
1275     \ifglsucmark
1276       \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1277     \else
1278       \@mkboth{#1}{#1}%
1279     \fi
1280   }
1281 }
1282 }

```

`\glossarymark` Provided for backward compatibility:

```

1283 \providecommand{\glossarymark}[1]{%
1284   \ifglsucmark
1285     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1286   \else
1287     \@mkboth{#1}{#1}%
1288   \fi
1289 }

```

The required sectional unit is given by `\@glossarysec` which was defined by the section package option. The starred form of the command is chosen. If you don't want any sectional command, you will need to redefine `\glossarysection`. The sectional unit can be changed, if different sectional units are required.

`\glossarysection`

```

1290 \newcommand*{\setglossarysection}[1]{%
1291 \setkeys{glossaries.sty}{section=#1}}

```

The command `\@glossarysection` indicates how to start the glossary section if `\phantomsection` is not defined.

`\glossarysection`

```

1292 \newcommand*{\@glossarysection}[2]{%
1293   \ifdefempty\@glossarysecstar
1294     {%
1295       \csname\@glossarysec\endcsname[#1]{#2}%
1296     }%
1297   {%
1298     \csname\@glossarysec\endcsname*{#2}%
1299     \@gls@toc{#1}{\@glossarysec}%
1300   }%

```

Do automatic labelling if required

```

1301   \@glossaryseclabel
1302 }

```

As `\@glossarysection`, but put in `\phantomsection`, and swap where `\@gls@toc` goes. If using chapters do a `\clearpage`. This ensures that the hyper link from the table of contents leads to the line above the heading, rather than the line below it.

`glossarysection`

```

1303 \newcommand*{\@p@glossarysection}[2]{%
1304   \glsclearpage
1305   \phantomsection
1306   \ifdefempty\@glossarysecstar
1307   {%
1308     \csname\@glossarysec\endcsname{#2}%
1309   }%
1310   {%
1311     \@gls@toc{#1}{\@glossarysec}%
1312     \csname\@glossarysec\endcsname*{#2}%
1313   }%
1314   Do automatic labelling if required
1315   \@glossaryseclabel
1316 }

```

`gls@doclearpage` The `\gls@doclearpage` command is used to issue a `\clearpage` (or `\cleardoublepage`) depending on whether the glossary sectional unit is a chapter. If the sectional unit is something else, do nothing.

```

1316 \newcommand*{\gls@doclearpage}{%
1317   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1318     {%
1319       \ifcsundef{cleardoublepage}%
1320       {%
1321         \clearpage
1322       }%
1323     }%
1324     \ifcsdef{if@openright}%
1325     {%
1326       \if@openright
1327         \cleardoublepage
1328       \else
1329         \clearpage
1330       \fi
1331     }%
1332     {%
1333       \cleardoublepage
1334     }%
1335   }%
1336 }%
1337 {}%
1338 }

```

`\glsclearpage` This just calls `\gls@doclearpage`, but it makes it easier to have a user command so that the user can override it.


```
1339 \newcommand*{\glsclearpage}{\gls@doclearpage}
```

The glossary is added to the table of contents if glstoc flag set. If it is set, \@gls@toc will add a line to the .toc file, otherwise it will do nothing. (The first argument to \@gls@toc is the title for the table of contents, the second argument is the sectioning type.)

\@gls@toc

```
1340 \newcommand*{\@gls@toc}[2]{%
1341   \ifglstoc
1342     \ifglxnumberline
1343       \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1344     \else
1345       \addcontentsline{toc}{#2}{#1}%
1346     \fi
1347   \fi
1348 }
```

1.4 Xindy

This section defines commands that only have an effect if xindy is used to sort the glossaries.

glsnoxindywarning Issues a warning if xindy hasn't been specified. These warnings can be suppressed by re-defining \glsnoxindywarning to ignore its argument

```
1349 \newcommand*{\glsnoxindywarning}[1]{%
1350   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1351 }
```

glsnomakeindexwarning Reverse for commands that may only be used with makeindex.

```
1352 \newcommand*{\glsnomakeindexwarning}[1]{%
1353   \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1354 }
```

\@xdyattributes Define list of attributes (\string is used in case the double quote character has been made active)

```
1355 \ifglxindy
1356   \edef\@xdyattributes{\string"default\string"}%
1357 \fi
```

\@xdyattributelist Comma-separated list of attributes.

```
1358 \ifglxindy
1359   \edef\@xdyattributelist{}%
1360 \fi
```

\@xdylocref Define list of markup location references.

```
1361 \ifglxindy
1362   \def\@xdylocref{}
1363 \fi
```

\@gls@ifinlist

```

1364 \newcommand*{\@gls@ifinlist}[4]{%
1365   \def\@do@ifinlist##1,#1,##2\end@do@ifinlist{%
1366     \def\@gls@listsuffix{##2}%
1367     \ifx\@gls@listsuffix\@empty
1368       #4%
1369     \else
1370       #3%
1371     \fi
1372   }%
1373   \@do@ifinlist,##2,#1,\end@do@ifinlist
1374 }

```

sAddXdyCounters Need to know all the counters that will be used in location numbers for Xindy. Argument may be a single counter name or a comma-separated list of counter names.

```

1375 \ifglxindy
1376   \newcommand*{\@xdycounters}{\glscounter}
1377   \newcommand*{\GlsAddXdyCounters}[1]{%
1378     \@for\@gls@ctr:=#1\do{%
1379       \edef\@do@addcounter{%
1380         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1381         {%
1382           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1383             \noexpand\@gls@ctr}%
1384         }%
1385       }%
1386       \@do@addcounter
1387     }
1388   }

```

Only has an effect before \writeist:

```

1389   \@onlypremakeg\GlsAddXdyCounters
1390 \else
1391   \newcommand*{\GlsAddXdyCounters}[1]{%
1392     \glsnoxindywarning\GlsAddXdyAttribute
1393   }
1394 \fi

```

saddxdycounters Counters must all be identified before adding attributes.

```

1395 \newcommand*{\@disabled@glssaddxdycounters{%
1396   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1397     can't be used after \string\GlsAddXdyAttribute}{Move all
1398     occurrences of \string\GlsAddXdyCounters\space before the first
1399     instance of \string\GlsAddXdyAttribute}%
1400 }

```

AddXdyAttribute Adds an attribute.

```

1401 \ifglxindy

```

First define internal command that adds an attribute for a given counter (2nd argument is the counter):

```

1402 \newcommand*\@glsaddxdyattribute[2]{%
    Add to xindy attribute list
1403 \edef\xdyattributes{\xdyattributes ^^J \string"#1\string" ^^J
1404 \string"#2#1\string"}%
    Add to xindy markup location.
1405 \expandafter\toks@\expandafter{\xdylocref}%
1406 \edef\xdylocref{\the\toks@ ^^J%
1407 (markup-locref
1408 :open \string"glstildechar n%
1409 \expandafter\string\csname glsX#2X#1\endcsname
1410 \string" ^^J
1411 :close \string"\string" ^^J
1412 :attr \string"#2#1\string")}%
    Define associated attribute command \glsX<counter>X<attribute>{\<Hprefix>}{\<n>}
1413 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1414 \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1415 }%
1416 }

```

High-level command:

```

1417 \newcommand*\GlsAddXdyAttribute[1]{%
    Add to comma-separated attribute list
1418 \ifx\xdyattributelist\empty
1419 \edef\xdyattributelist{#1}%
1420 \else
1421 \edef\xdyattributelist{\xdyattributelist,#1}%
1422 \fi
    Iterate through all specified counters and add counter-dependent attributes:
1423 \@for\@this@counter:=\xdycounters\do{%
1424 \protected@edef\gls@do@addxdyattribute{%
1425 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1426 }
1427 \gls@do@addxdyattribute
1428 }%

```

All occurrences of \GlsAddXdyCounters must be used before this command

```

1429 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1430 }

```

Only has an effect before \writeist:

```

1431 \@onlypremakeg\GlsAddXdyAttribute
1432 \else
1433 \newcommand*\GlsAddXdyAttribute[1]{%
1434 \glsnoxindywarning\GlsAddXdyAttribute}
1435 \fi

```

`definedattributes` Add known attributes for all defined counters

```

1436 \ifglxindy
1437 \newcommand*{\@gls@addpredefinedattributes}{%
1438   \GlsAddXdyAttribute{glsnumberformat}
1439   \GlsAddXdyAttribute{textrm}
1440   \GlsAddXdyAttribute{textsf}
1441   \GlsAddXdyAttribute{texttt}
1442   \GlsAddXdyAttribute{textbf}
1443   \GlsAddXdyAttribute{textmd}
1444   \GlsAddXdyAttribute{textit}
1445   \GlsAddXdyAttribute{textup}
1446   \GlsAddXdyAttribute{textsl}
1447   \GlsAddXdyAttribute{textsc}
1448   \GlsAddXdyAttribute{emph}
1449   \GlsAddXdyAttribute{glshypernumber}
1450   \GlsAddXdyAttribute{hyperrm}
1451   \GlsAddXdyAttribute{hypersf}
1452   \GlsAddXdyAttribute{hypertt}
1453   \GlsAddXdyAttribute{hyperbf}
1454   \GlsAddXdyAttribute{hypermd}
1455   \GlsAddXdyAttribute{hyperit}
1456   \GlsAddXdyAttribute{hyperup}
1457   \GlsAddXdyAttribute{hypersl}
1458   \GlsAddXdyAttribute{hypersc}
1459   \GlsAddXdyAttribute{hyperemph}

1460   \GlsAddXdyAttribute{glsignore}
1461 }
1462 \else
1463   \let\@gls@addpredefinedattributes\relax
1464 \fi

```

`xdyuseralphabets` List of additional alphabets

```

1465 \def\@xdyuseralphabets{}

```

`\GlsAddXdyAlphabet` `\GlsAddXdyAlphabet{<name>}{<definition>}` adds a new alphabet called *<name>*. The definition must use xindy syntax.

```

1466 \ifglxindy
1467   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1468     \edef\@xdyuseralphabets{%
1469       \@xdyuseralphabets ^^J
1470       (define-alphabet "#1" (#2))}}
1471 \else
1472   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1473     \glsnoxindywarning\GlsAddXdyAlphabet}
1474 \fi

```

This code is only required for xindy:

```

1475 \ifglxindy

```

dy@locationlist List of predefined location names.

```
1476 \newcommand*{\@gls@xdy@locationlist}{%
1477     roman-page-numbers,%
1478     Roman-page-numbers,%
1479     arabic-page-numbers,%
1480     alpha-page-numbers,%
1481     Alpha-page-numbers,%
1482     Appendix-page-numbers,%
1483     arabic-section-numbers%
1484 }
```

Each location class *<name>* has the format stored in \@gls@xdy@Lclass@*<name>*. Set up predefined formats.

an-page-numbers Lower case Roman numerals (i, ii, ...). In the event that \roman has been redefined to produce a fancy form of roman numerals, attempt to work out how it will be written to the output file.

```
1485 \protected@edef\@gls@roman{\@roman{0}\string"
1486     \string"roman-numbers-lowercase\string" :sep \string"}}%
1487 \@onelevel@sanitize\@gls@roman
1488 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1489     :sep \string"}%
1490 \@onelevel@sanitize\@tmp
1491 \ifx\@tmp\@gls@roman
1492     \expandafter
1493     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1494         \string"roman-numbers-lowercase\string"%
1495     }%
1496 \else
1497     \expandafter
1498     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1499         :sep \string"\@gls@roman\string"%
1500     }%
1501 \fi
```

an-page-numbers Upper case Roman numerals (I, II, ...).

```
1502 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1503     \string"roman-numbers-uppercase\string"%
1504 }
```

ic-page-numbers Arabic numbers (1, 2, ...).

```
1505 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1506     \string"arabic-numbers\string"%
1507 }
```

ha-page-numbers Lower case alphabetical (a, b, ...).

```
1508 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1509     \string"alpha\string"%
1510 }
```

alpha-page-numbers Upper case alphabetical (A, B, ...).

```

1511 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1512     \string"ALPHA\string"%
1513 }%
```

ix-page-numbers Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given by \@glsAlphacompositor.

```

1514 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1515     \string"ALPHA\string"
1516     :sep \string"\@glsAlphacompositor\string"
1517     \string"arabic-numbers\string"%
1518 }
```

section-numbers Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by \glscompositor.

```

1519 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1520     \string"arabic-numbers\string"
1521     :sep \string"\glscompositor\string"
1522     \string"arabic-numbers\string"%
1523 }%
```

serlocationdefs List of additional location definitions (separated by ^^J)

```

1524 \def\@xdyuserlocationdefs{}
```

erlocationnames List of additional user location names

```

1525 \def\@xdyuserlocationnames{}
```

End of xindy-only block:

```

1526 \fi
```

xdycrossrefhook Hook used after writing cross-reference class information.

```

1527 \ifglsxindy
1528 \newcommand\@xdycrossrefhook{}
1529 \fi
```

sAddXdyLocation \GlsAddXdyLocation[*<prefix-loc>*]{*<name>*}{*<definition>*} Define a new location called *<name>*. The definition must use xindy syntax. (Note that this doesn't check to see if the location is already defined. That is left to xindy to complain about.)

```

1530 \ifglsxindy
1531 \newcommand*\GlsAddXdyLocation[3][{}]{%
1532     \def\@gls@tmp{#1}%
1533     \ifx\@gls@tmp\@empty
1534         \edef\@xdyuserlocationdefs{%
1535             \@xdyuserlocationdefs ^^J%
1536             (define-location-class \string"#2\string"^^J\space\space
1537             \space(:sep \string"{}\glsopenbrace\string" #3
1538                 :sep \string"\glsclosebrace\string"))
1539     }%
```

```

1540 \else
1541 \edef\@xdyuserlocationdefs{%
1542 \@xdyuserlocationdefs ^^J%
1543 (define-location-class \string"#2\string"^^J\space\space
1544 \space(:sep "\glsoopenbrace"
1545 #1
1546 :sep "\glsclosebrace\glsoopenbrace" #3
1547 :sep "\glsclosebrace"))
1548 }%
1549 \fi

1550 \edef\@xdyuserlocationnames{%
1551 \@xdyuserlocationnames^^J\space\space\space
1552 \string"#2\string"}%
1553 }

```

Only has an effect before \writeist:

```

1554 \@onlypremakeg\GlsAddXdyLocation
1555 \else
1556 \newcommand*\{GlsAddXdyLocation}[2]{%
1557 \glsnnoxindywarning\GlsAddXdyLocation}
1558 \fi

```

ationclassorder Define location class order

```

1559 \ifglxindy
1560 \def\@xdylocationclassorder{^^J\space\space\space
1561 \string"roman-page-numbers\string"^^J\space\space\space
1562 \string"arabic-page-numbers\string"^^J\space\space\space
1563 \string"arabic-section-numbers\string"^^J\space\space\space
1564 \string"alpha-page-numbers\string"^^J\space\space\space
1565 \string"Roman-page-numbers\string"^^J\space\space\space
1566 \string"Alpha-page-numbers\string"^^J\space\space\space
1567 \string"Appendix-page-numbers\string"
1568 \@xdyuserlocationnames^^J\space\space\space
1569 \string"see\string"
1570 }
1571 \fi

```

Change the location order.

ationClassOrder

```

1572 \ifglxindy
1573 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1574 \def\@xdylocationclassorder{#1}}
1575 \else
1576 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1577 \glsnnoxindywarning\GlsSetXdyLocationClassOrder}
1578 \fi

```

\@xdysortrules Define sort rules

```

1579 \ifglxindy
1580   \def\@xdysortrules{}
1581 \fi

```

`\GlsAddSortRule` Add a sort rule

```

1582 \ifglxindy
1583   \newcommand*\GlsAddSortRule[2]{%
1584     \expandafter\toks@\expandafter{\@xdysortrules}%
1585     \protected@edef\@xdysortrules{\the\toks@ ^^J
1586       (sort-rule \string"#1\string" \string"#2\string")}%
1587   }
1588 \else
1589   \newcommand*\GlsAddSortRule[2]{%
1590     \glsnxindywarning\GlsAddSortRule}
1591 \fi

```

`\xyrequiredstyles` Define list of required styles (this should be a comma-separated list of xindy styles)

```

1592 \ifglxindy
1593   \def\@xdyrequiredstyles{tex}
1594 \fi

```

`\GlsAddXdyStyle` Add a xindy style to the list of required styles

```

1595 \ifglxindy
1596   \newcommand*\GlsAddXdyStyle[1]{%
1597     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1598 \else
1599   \newcommand*\GlsAddXdyStyle[1]{%
1600     \glsnxindywarning\GlsAddXdyStyle}
1601 \fi

```

`\GlsSetXdyStyles` Reset the list of required styles

```

1602 \ifglxindy
1603   \newcommand*\GlsSetXdyStyles[1]{%
1604     \edef\@xdyrequiredstyles{#1}}
1605 \else
1606   \newcommand*\GlsSetXdyStyles[1]{%
1607     \glsnxindywarning\GlsSetXdyStyles}
1608 \fi

```

`\findrootlanguage` This used to determine the root language, using a bit of trickery since babel doesn't supply the information, but now that babel is once again actively maintained, we can't do this any more, so `\findrootlanguage` is no longer available. Now provide a command that does nothing (in case it's been patched), but this may be removed completely in the future.

```

1609 \newcommand*\findrootlanguage{}

```

`\@xdylanguage` The xindy language setting is required by `makeglossaries`, so provide a command for `makeglossaries` to pick up the information from the auxiliary file. This command is not needed by the `glossaries` package, so define it to ignore its arguments.

```

1610 \def\@xdylanguage#1#2{}

```


`\GlsSetXdyLanguage` Define a command that allows the user to set the language for a given glossary type. The first argument indicates the glossary type. If omitted the main glossary is assumed.

```

1611 \ifglxsindy
1612   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1613     \ifglossaryexists{#1}{%
1614       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1615     }{%
1616       \PackageError{glossaries}{Can't set language type for
1617         glossary type '#1' --- no such glossary}{%
1618         You have specified a glossary type that doesn't exist}}
1619 \else
1620   \newcommand*\GlsSetXdyLanguage[2][]{%
1621     \glsnoxywarning\GlsSetXdyLanguage}
1622 \fi

```

`\@gls@codepage` The xindy codepage setting is required by makeglossaries, so provide a command for makeglossaries to pick up the information from the auxiliary file. This command is not needed by the glossaries package, so define it to ignore its arguments.

```

1623 \def\@gls@codepage#1#2{}

```

`\GlsSetXdyCodePage` Define command to set the code page.

```

1624 \ifglxsindy
1625   \newcommand*\GlsSetXdyCodePage[1]{%
1626     \renewcommand*\@gls@codepage{#1}%
1627   }

```

Suggested by egreg:

```

1628 \AtBeginDocument{%
1629   \ifx\@gls@codepage\empty
1630     \ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{}%
1631   \fi
1632 }
1633 \else
1634   \newcommand*\GlsSetXdyCodePage[1]{%
1635     \glsnoxywarning\GlsSetXdyCodePage}
1636 \fi

```

`\@xdylettergroups` Store letter group definitions.

```

1637 \ifglxsindy
1638   \ifglsexindy@glsnumbers
1639     \def\@xdylettergroups{(define-letter-group
1640       \string"glslnumbers\string"^^J\space\space\space
1641       :prefixes (\string"0\string" \string"1\string"
1642       \string"2\string" \string"3\string" \string"4\string"
1643       \string"5\string" \string"6\string" \string"7\string"
1644       \string"8\string" \string"9\string")^^J\space\space\space
1645       \@xdynumbergrouporder)}
1646   \else
1647     \def\@xdylettergroups{}

```

```

1648 \fi
1649 \fi

```

\GlsAddLetterGroup Add a new letter group. The first argument is the name of the letter group. The second argument is the xindy code specifying prefixes and ordering.

```

1650 \newcommand*\GlsAddLetterGroup[2]{%
1651   \expandafter\toks@\expandafter{\@xdylettergroups}%
1652   \protected@edef\@xdylettergroups{\the\toks@^^J%
1653   (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1654 }%

```

1.5 Loops and conditionals

\forallglossaries To iterate through all glossaries (or comma-separated list of glossary names given in optional argument) use:

```
\forallglossaries[<glossary list>]{<cmd>}{<code>}
```

where *<cmd>* is a control sequence which will be set to the name of the glossary in the current iteration.

```

1655 \newcommand*\forallglossaries[3][\@glo@types]{%
1656   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1657 }

```

\forallacronyms

```

1658 \newcommand*\forallacronyms[2]{%
1659   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1660 }

```

\forglentries To iterate through all entries in a given glossary use:

```
\forglentries[<type>]{<cmd>}{<code>}
```

where *<type>* is the glossary label and *<cmd>* is a control sequence which will be set to the entry label in the current iteration.

```

1661 \newcommand*\forglentries[3][\glsdefaultttype]{%
1662   \edef\@glo@list{\csname glolist@#1\endcsname}%
1663   \@for#2:=\@glo@list\do
1664   {%
1665     \ifdefempty{#2}{#3}%
1666   }%
1667 }

```

\forallglentries To iterate through all glossary entries over all glossaries listed in the optional argument (the default is all glossaries) use:

```
\foralllglentries[⟨glossary list⟩]{⟨cmd⟩}{⟨code⟩}
```

Within `\foralllglentries`, the current glossary type is given by `\@@this@glo@`.

```
1668 \newcommand*{\foralllglentries}[3][\@glo@types]{%
1669   \expandafter\foralllglossaries\expandafter[#1]{\@@this@glo@}%
1670   {%
1671     \forallglentries[\@@this@glo@]{#2}{#3}%
1672   }%
1673 }
```

`\ifglossaryexists` To check to see if a glossary exists use:

```
\ifglossaryexists{⟨type⟩}{⟨true-text⟩}{⟨false-text⟩}
```

where `⟨type⟩` is the glossary's label.

```
1674 \newcommand{\ifglossaryexists}[3]{%
1675   \ifcsundef{@glo@type@#1@out}{#3}{#2}%
1676 }
```

Since the label is used to form the name of control sequences, by default UTF8 etc characters can't be used in the label. A possible workaround is to use `\scantokens`, but commands such as `\glentrytext` will no longer be usable in sectioning, caption etc commands. If the user really wants to be able to construct a label with UTF8 characters, allow them the means to do so (but on their own head be it, if they then use entries in `\section` etc). This can be done via:

```
\renewcommand*{\glsetoklabel}[1]{\scantokens{#1\noexpand}}
```

(Note, don't use `\detokenize` or it will cause commands like `\gladdall` to fail.) Since re-defining `\glsetoklabel` can cause things to go badly wrong, I'm not going to mention it in the main user guide. Only advanced users who know what they're doing ought to attempt it.

`\glsetoklabel`

```
1677 \newcommand*{\glsetoklabel}[1]{#1}
```

`\ifglentryexists` To check to see if a glossary entry has been defined use:

```
\ifglentryexists{⟨label⟩}{⟨true text⟩}{⟨false text⟩}
```

where `⟨label⟩` is the entry's label.

```
1678 \newcommand{\ifglentryexists}[3]{%
1679   \ifcsundef{glo@\glsetoklabel{#1}@name}{#3}{#2}%
1680 }
```

`\ifglused` To determine if given glossary entry has been used in the document text yet use:

```
\ifglused{<label>}{<true text>}{<false text>}
```

where *<label>* is the entry's label. If true it will do *<true text>* otherwise it will do *<false text>*.

```
1681 \newcommand*\ifglused}[3]{%
1682   \ifbool{glo@\glsetoklabel{#1}@flag}{#2}{#3}%
1683 }
```

The following two commands will cause an error if the given condition fails:

`\glsoifexists`

```
\glsoifexists{<label>}{<code>}
```

Generate an error if entry specified by *<label>* doesn't exist, otherwise do *<code>*.

```
1684 \newcommand{\glsoifexists}[2]{%
1685   \ifglentryexists{#1}{#2}{%
1686     \PackageError{glossaries}{Glossary entry ‘\glsetoklabel{#1}’
1687       has not been defined}{You need to define a glossary entry before you
1688       can use it.}}%
1689 }
```

`\glsoifnoexists`

```
\glsoifnoexists{<label>}{<code>}
```

The opposite: only do second argument if the entry doesn't exist. Generate an error message if it exists.

```
1690 \newcommand{\glsoifnoexists}[2]{%
1691   \ifglentryexists{#1}{#2}{%
1692     \PackageError{glossaries}{Glossary entry ‘\glsetoklabel{#1}’ has already
1693       been defined}{}}{#2}%
1694 }
```

`\glsoifexistsorwarn`

```
\glsoifexistsorwarn{<label>}{<code>}
```

Generate a warning if entry specified by *<label>* doesn't exist, otherwise do *<code>*.

```
1695 \newcommand{\glsoifexistsorwarn}[2]{%
1696   \ifglentryexists{#1}{#2}{%
1697     \GlossariesWarning{Glossary entry ‘\glsetoklabel{#1}’
1698       has not been defined}%
1699   }%
1700 }
```

`\glsoifexistsordo`

```
\glsoifexistsordo{<label>}{<code>}{<undef code>}
```

Generate an error and do *<undef code>* if entry specified by *<label>* doesn't exist, otherwise do *<code>*.

```
1701 \newcommand{\glsoifexistsordo}[3]{%
1702   \ifglentryexists{#1}{#2}{%

```

```

1703 \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1704 has not been defined}{You need to define a glossary entry before you
1705 can use it.}%
1706 #3%
1707 }%
1708 }

```

arynoexistsordo

```
\doifglossarynoexistsordo{<label>}{<code>}{<else code>}
```

If glossary given by *<label>* doesn't exist do *<code>* otherwise generate an error and do *<else code>*.

```

1709 \newcommand{\doifglossarynoexistsordo}[3]{%
1710 \ifglossaryexists{#1}%
1711 {%
1712 \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1713 #3%
1714 }%
1715 {#2}%
1716 }

```

fglshaschildren

```
\ifglshaschildren{<label>}{<true part>}{<false part>}
```

This is inefficient as it has to search through all entries to find out which ones have the given entry as its parent. It's much easier to use bib2gls and get it to store the list of children that have been indexed (which is likely to be more useful).

```

1717 \newrobustcmd{\ifglshaschildren}[3]{%
1718 \glsdoifexists{#1}%
1719 {%
1720 \def\do@glshaschildren{#3}%
1721 \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1722 \expandafter\for@gl@sentries\expandafter
1723 [\csname glo@\@gls@thislabel @type\endcsname]
1724 {\glo@label}%
1725 {%
1726 \letcs\glo@parent{glo@\glo@label @parent}%
1727 \ifdefequal\@gls@thislabel\glo@parent
1728 {%
1729 \def\do@glshaschildren{#2}%
1730 \@endfortrue
1731 }%
1732 }%
1733 }%
1734 \do@glshaschildren
1735 }%
1736 }

```

`\ifglshasparent` `\ifglshasparent{<label>}{<true part>}{<false part>}`

```

1737 \newcommand{\ifglshasparent}[3]{%
1738   \glsdoifexists{#1}%
1739   {%
1740     \ifcsemtty{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1741   }%
1742 }

```

`\ifglshasdesc` `\ifglshasdesc{<label>}{<true part>}{<false part>}`

```

1743 \newcommand*{\ifglshasdesc}[3]{%
1744   \ifcsemtty{glo@\glsdetoklabel{#1}@desc}%
1745   {#3}%
1746   {#2}%
1747 }

```

`sdescsuppressed` `\ifglsdescsuppressed{<label>}{<true part>}{<false part>}` Does <true part> if the description is just `\nopostdesc` otherwise does <false part>.

```

1748 \newcommand*{\ifglsdescsuppressed}[3]{%
1749   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1750   {#2}%
1751   {#3}%
1752 }

```

`\ifglshassymbol` `\ifglshassymbol{<label>}{<true part>}{<false part>}`

```

1753 \newrobustcmd*{\ifglshassymbol}[3]{%
1754   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1755   \ifdefempty\@glo@symbol
1756   {#3}%
1757   {%
1758     \ifdefequal\@glo@symbol\@gls@default@value
1759     {#3}%
1760     {#2}%
1761   }%
1762 }

```

`\ifglshaslong` `\ifglshaslong{<label>}{<true part>}{<false part>}`

```

1763 \newrobustcmd*{\ifglshaslong}[3]{%
1764   \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
1765   \ifdefempty\@glo@long
1766   {#3}%
1767   {%
1768     \ifdefequal\@glo@long\@gls@default@value
1769     {#3}%
1770     {#2}%
1771   }%
1772 }

```

```

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1773 \newrobustcmd*{\ifglshasshort}[3]{%
1774   \letcs{\@glo@short}{glo@\glsdetoklabel{#1}@short}%
1775   \ifdefempty\@glo@short
1776     {#3}%
1777     {%
1778       \ifdefequal\@glo@short\@gls@default@value
1779         {#3}%
1780         {#2}%
1781       }%
1782 }

```

\ifglshasfield

\ifglshasfield{<field>}{<label>}{<true part>}{<false part>}

```

1783 \newrobustcmd*{\ifglshasfield}[4]{%
1784   \glsdoifexists{#2}%
1785   {%
1786     \letcs{\@glo@thisvalue}{glo@\glsdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1787   \ifdef\@glo@thisvalue
1788     {%

```

Is defined, so now check if empty.

```

1789     \ifdefempty\@glo@thisvalue
1790     {%

```

Is empty, so doesn't have field set.

```

1791       #4%
1792     }%
1793   {%

```

Not empty, so check if set to \@gls@default@value

```

1794     \ifdefequal\@glo@thisvalue\@gls@default@value
1795     {%

```

Value is set to the default value.

```

1796       #4%
1797     }%
1798   {%

```

Non-empty, non-default value. Allow user to access this value through \glscurrentfieldvalue.

```

1799     \let\glscurrentfieldvalue\@glo@thisvalue
1800     #3%
1801   }%
1802 }%
1803 }%
1804 {%

```

Field given isn't defined, so check if mapping exists.

```
1805 \gls@fetchfield{\gls@thisfield}{#1}%
```

If `\gls@thisfield` is defined, we've found a map. If not, the field supplied doesn't exist.

```
1806 \ifdef\gls@thisfield
1807 {%
```

Is defined, so now check if empty.

```
1808 \letcs{\glo@thisvalue}{glo\glsdetoklabel{#2}@\gls@thisfield}%
1809 \ifdefempty\glo@thisvalue
1810 {%
```

Is empty so field hasn't been set.

```
1811 #4%
1812 }%
1813 {%
```

Isn't empty so check if it's been set to `\gls@default@value`.

```
1814 \ifdequal\glo@thisvalue\gls@default@value
1815 {%
```

Value is set to the default value.

```
1816 #4%
1817 }%
1818 {%
```

Non-empty, non-default value. Allow user to access this value through `\glscurrentfieldvalue`.

```
1819 \let\glscurrentfieldvalue\glo@thisvalue
1820 #3%
1821 }%
1822 }%
1823 }%
1824 {%
```

Not defined.

```
1825 \GlossariesWarning{Unknown entry field '#1'}%
1826 #4%
1827 }%
1828 }%
1829 }%
1830 }
```

`\glscurrentfieldvalue`

```
1831 \newcommand*{\glscurrentfieldvalue}{}
```

1.6 Defining new glossaries

A comma-separated list of glossary names is stored in `\glo@types`. When a new glossary type is created, its identifying name is added to this list. This is used by commands that iterate through all glossaries (such as `\makeglossaries` and `\printglossaries`).


```

\@glo@types
1832 \newcommand*{\@glo@types}{,}

\provide@newglossary If the user removes the glossary package from their document, ensure the next run doesn't
throw a load of undefined control sequence errors when the aux file is parsed.
1833 \newcommand*\@gls@provide@newglossary{%
1834   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}}%

Only need to do this once.
1835 \let\@gls@provide@newglossary\relax
1836 }

\defglsentryfmt Allow different glossaries to have different display styles.
1837 \newcommand*\defglsentryfmt}[2][\glsdefaultttype]{%
1838   \csgdef{gls@#1@entryfmt}{#2}%
1839 }

\gls@doentryfmt
1840 \newcommand*\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}

\@gls@forbidtextext As a security precaution, don't allow the user to specify a 'tex' extension for any of the glossary
files. (Just in case a seriously confused novice user doesn't know what they're doing.) The
argument must be a control sequence whose replacement text is the requested extension.
1841 \newcommand*\@gls@forbidtextext}[1]{%
1842   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1843     or test {\ifdefstring{#1}{TEX}}}
1844   {%
1845     \def#1{nottex}%
1846     \PackageError{glossaries}%
1847       {Forbidden '.tex' extension replaced with '.nottex'}%
1848     {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1849       Don't use '.tex' as an extension for a temporary file.}%
1850   }%
1851   {%
1852   }%
1853 }

\gls@gobbleopt Discard optional argument.
1854 \newcommand*\gls@gobbleopt{\new@ifnextchar[\@gls@gobbleopt]}
1855 \def\@gls@gobbleopt[#1]{}

```

A new glossary type is defined using `\newglossary`. Syntax:

```
\newglossary[⟨log-ext⟩]{⟨name⟩}{⟨in-ext⟩}{⟨out-ext⟩} {⟨title⟩}[⟨counter⟩]
```

where `⟨log-ext⟩` is the extension of the makeindex transcript file, `⟨in-ext⟩` is the extension of the glossary input file (read in by `\printglossary` and created by makeindex), `⟨out-ext⟩`

is the extension of the glossary output file which is read in by `makeindex` (lines are written to this file by the `\glossary` command), *<title>* is the title of the glossary that is used in `\glossarysection` and *<counter>* is the default counter to be used by entries belonging to this glossary. The `makeglossaries` Perl script reads in the relevant extensions from the auxiliary file, and passes the appropriate file names and switches to `makeindex`.

`\newglossary`

```
1856 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}
```

`\s@newglossary` The starred version will construct the extension based on the label.

```
1857 \newcommand*{\s@newglossary}[2]{%
```

```
1858 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
```

```
1859 }
```

`\ns@newglossary` Define the unstarred version.

```
1860 \newcommand*{\ns@newglossary}[5][glg]{%
```

```
1861 \doifglossarynoexistsordo{#2}%
```

```
1862 {%
```

Check if default has been set

```
1863 \ifundef\glsdefaultttype
```

```
1864 {%
```

```
1865 \gdef\glsdefaultttype{#2}%
```

```
1866 }{}%
```

Add this to the list of glossary types:

```
1867 \toks@{#2}\edef\@glo@types{\@glo@types\the\toks@,}%
```

Define a comma-separated list of labels for this glossary type, so that all the entries for this glossary can be reset with a single command. When a new entry is created, its label is added to this list.

```
1868 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1869 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
```

```
1870 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
```

```
1871 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
```

```
1872 \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
```

```
1873 \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
```

```
1874 \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1875 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
```

```
1876 \@gls@provide@newglossary
```

```
1877 \protected@write\auxout{}{\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsentry` by default). This can be re-defined by the user later if required (see `\defglsentry`). This may already have been defined if this has been specified as a list of acronyms.

```

1878 \ifcsundef{gls@#2@entryfmt}%
1879 {%
1880   \defglsentryfmt[#2]{\glsentryfmt}%
1881 }%
1882 {}%

```

Define sort counter if required:

```

1883 \@gls@defsortcount{#2}%

```

Find out if the final optional argument has been specified, and use it to set the counter associated with this glossary. (Uses `\glscounter` if no optional argument is present.)

```

1884 \@ifnextchar[{\@gls@setcounter{#2}}%
1885   {\@gls@setcounter{#2}[\glscounter]}%
1886 }%
1887 {%
1888   \gls@gobbleopt
1889 }%
1890 }

```

`\altnewglossary`

```

1891 \newcommand*{\altnewglossary}[3]{%
1892   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1893 }

```

Only define new glossaries in the preamble:

```

1894 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1895 \@onlypremakeg\newglossary

```

`\@newglossary` is used to specify the file extensions for the `makeindex` input, output and transcript files. It is written to the auxiliary file by `\newglossary`. Since it is not used by \LaTeX , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```

1896 \newcommand*{\@newglossary}[4]{}

```

Store counter to be used for given glossary type (the first argument is the glossary label, the second argument is the name of the counter):

`@gls@setcounter`

```

1897 \def\@gls@setcounter#1[#2]{%
1898   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```

1899   \ifglsxindy
1900     \GlsAddXdyCounters{#2}%
1901   \fi
1902 }

```

Get counter associated with given glossary (the argument is the glossary label):

@gls@getcounter

```
1903 \newcommand*{\@gls@getcounter}[1]{%
1904 \csname @gls@#1@counter\endcsname
1905 }
```

Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`.

```
1906 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1907 \@gls@do@acronymsdef
```

Define the “symbols”, “numbers” and “index” glossaries if required.

```
1908 \@gls@do@symbolsdef
```

```
1909 \@gls@do@numbersdef
```

```
1910 \@gls@do@indexdef
```

`ignoredglossary` Creates a new glossary that doesn’t have associated files. This glossary is ignored by and commands that iterate over glossaries, such as `\printglossaries`, and won’t work with commands like `\printglossary`. It’s intended for entries that are so commonly-known they don’t require a glossary.

```
1911 \newcommand*{\newignoredglossary}[1]{%
1912 \ifdefempty\@ignored@glossaries
1913 {%
1914 \edef\@ignored@glossaries{#1}%
1915 }%
1916 {%
1917 \eappto\@ignored@glossaries{, #1}%
1918 }%
1919 \csgdef{glslist@#1}{,}%
1920 \ifcsundef{gls@#1@entryfmt}%
1921 {%
1922 \defglsentryfmt[#1]{\glsentryfmt}%
1923 }%
1924 {}%
1925 \ifdefempty\@gls@nohyperlist
1926 {%
1927 \renewcommand*{\@gls@nohyperlist}{#1}%
1928 }%
1929 {%
1930 \eappto\@gls@nohyperlist{, #1}%
1931 }%
1932 }
```

`ignored@glossaries` List of ignored glossaries.

```
1933 \newcommand*{\@ignored@glossaries}{}
```

`ignoredglossary` Tests if the given glossary is an ignored glossary. Expansion is used in case the first argument is a control sequence.

```

1934 \newcommand*{\ifignoredglossary}[3]{%
1935   \edef\@gls@igtype{#1}%
1936   \expandafter\DTLifinlist\expandafter
1937     {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1938 }

```

1.7 Defining new entries

New glossary entries are defined using `\newglossaryentry`. This command requires a label and a key-value list that defines the relevant information for that entry. The definition for these keys follows. Note that the name, description and symbol keys will be sanitized later, depending on the value of the package option `sanitize` (this means that if some of the keys haven't been defined, they can be constructed from the name and description key before they are sanitized).

name The name key indicates the name of the term being defined. This is how the term will appear in the glossary. The name key is required when defining a new glossary entry.

```

1939 \define@key{glossentry}{name}{%
1940 \def\@glo@name{#1}%
1941 }

```

description The description key is usually only used in the glossary, but can be made to appear in the text by redefining `\glsentryfmt` or using `\defglsentryfmt`. The description key is required when defining a new glossary entry. If a long description is required, use `\longnewglossaryentry` instead of `\newglossaryentry`.

```

1942 \define@key{glossentry}{description}{%
1943 \def\@glo@desc{#1}%
1944 }

```

descriptionplural

```

1945 \define@key{glossentry}{descriptionplural}{%
1946 \def\@glo@descplural{#1}%
1947 }

```

sort The sort key needs to be sanitized here (the sort key is provided for `makeindex`'s benefit, not for use in the document). The sort key is optional when defining a new glossary entry. If omitted, the value is given by *<name>* *<description>*.

```

1948 \define@key{glossentry}{sort}{%
1949 \def\@glo@sort{#1}}

```

text The text key determines how the term should appear when used in the document (i.e. outside of the glossary). If omitted, the value of the name key is used instead.

```

1950 \define@key{glossentry}{text}{%
1951 \def\@glo@text{#1}%
1952 }

```

plural The plural key determines how the plural form of the term should be displayed in the document. If omitted, the plural is constructed by appending `\glspluralsuffix` to the value of the text key.

```
1953 \define@key{glossentry}{plural}{%
1954 \def\@glo@plural{#1}%
1955 }
```

first The first key determines how the entry should be displayed in the document when it is first used. If omitted, it is taken to be the same as the value of the text key.

```
1956 \define@key{glossentry}{first}{%
1957 \def\@glo@first{#1}%
1958 }
```

firstplural The firstplural key is used to set the plural form for first use, in the event that the plural is required the first time the term is used. If omitted, it is constructed by appending `\glspluralsuffix` to the value of the first key.

```
1959 \define@key{glossentry}{firstplural}{%
1960 \def\@glo@firstplural{#1}%
1961 }
```

s@default@value

```
1962 \newcommand*{\@gls@default@value}{\relax}
```

symbol The symbol key is ignored by most of the predefined glossary styles, and defaults to `\relax` if omitted. It is provided for glossary styles that require an associated symbol, as well as a name and description. To make this value appear in the glossary, you need to redefine `\glossentry`. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsentryfmt` (or use for `\defglsentryfmt` individual glossaries).

```
1963 \define@key{glossentry}{symbol}{%
1964 \def\@glo@symbol{#1}%
1965 }
```

symbolplural

```
1966 \define@key{glossentry}{symbolplural}{%
1967 \def\@glo@symbolplural{#1}%
1968 }
```

type The type key specifies to which glossary this entry belongs. If omitted, the default glossary is used.

```
1969 \define@key{glossentry}{type}{%
1970 \def\@glo@type{#1}}
```

counter The counter key specifies the name of the counter associated with this glossary entry:

```
1971 \define@key{glossentry}{counter}{%
1972 \ifcsundef{c@#1}%
```

```

1973 {%
1974   \PackageError{glossaries}%
1975   {There is no counter called ‘#1’}%
1976   {%
1977     The counter key should have the name of a valid counter
1978     as its value%
1979   }%
1980 }%
1981 {%
1982   \def\@glo@counter{#1}%
1983 }%
1984 }

```

see The see key specifies a list of cross-references

```

1985 \define@key{glossentry}{see}{%
1986   \gls@set@xr@key{see}{\@glo@see}{#1}%
1987 }

```

\gls@set@xr@key `\gls@set@xr@key{<key name>}{<cs>}{<value>}`

Assign a cross-reference key.

```

1988 \newcommand*{\gls@set@xr@key}[3]{%
1989   \renewcommand*{\gls@xr@key}{#1}%
1990   \gls@checkseeallowed
1991   \def#2{#3}%
1992   \@glo@seeautonumberlist
1993 }

```

\gls@xr@key

```

1994 \newcommand*{\gls@xr@key}{see}

```

checkseeallowed

```

1995 \newcommand*{\gls@checkseeallowed}{%
1996   \@gls@see@noindex
1997 }

```

ed@preambleonly

```

1998 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1999   \GlossariesWarning{glossaries}%
2000   {'\gls@xr@key' key doesn't have any effect when used in the document
2001     environment. Move the definition to the preamble
2002     after \string\makeglossaries\space
2003     or \string\makenoidxglossaries}%
2004 }

```

parent The parent key specifies the parent entry, if required.

```

2005 \define@key{glossentry}{parent}{%
2006   \def\@glo@parent{#1}}

```

nonumberlist The nonumberlist key suppresses or activates the number list for the given entry.

```
2007 \define@choicekey{glossentry}{nonumberlist}{%
2008   [\gls@nonumberlist@val\gls@nonumberlist@nr]{true,false}[true]%
2009 {%
2010   \ifcase\gls@nonumberlist@nr\relax
2011     \def\@glo@prefix{\glsnonextpages}%
2012     \@gls@savenonumberlist{true}%
2013   \else
2014     \def\@glo@prefix{\glsnextpages}%
2015     \@gls@savenonumberlist{false}%
2016   \fi
2017 }
```

savenonumberlist The nonumberlist option isn't saved by default (as it just sets the prefix) which isn't a problem when the entries are defined in the preamble, but causes a problem when entries are defined in the document. In this case, the value needs to be saved so that it can be written to the .glsdefs file.

```
2018 \newcommand*{\@gls@savenonumberlist}[1]{}
```

initnonumberlist

```
2019 \newcommand*{\@gls@initnonumberlist}{}%
```

storenonumberlist

```
2020 \newcommand*{\@gls@storenonumberlist}[1]{}
```

savenonumberlist Allow the nonumberlist value to be saved.

```
2021 \newcommand*{\@gls@enablesavenonumberlist}{%
2022   \renewcommand*{\@gls@initnonumberlist}{%
2023     \undef\@glo@nonumberlist
2024   }%
2025   \renewcommand*{\@gls@savenonumberlist}[1]{%
2026     \def\@glo@nonumberlist{##1}%
2027   }%
2028   \renewcommand*{\@gls@storenonumberlist}[1]{%
2029     \ifdef\@glo@nonumberlist
2030       {%
2031         \cslet{glo@\glsdetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
2032       }%
2033     }%
2034   }%
2035   \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
2036 }
```

Define some generic user keys. (Additional keys can be added by the user.)

user1

```
2037 \define@key{glossentry}{user1}{%
2038   \def\@glo@useri{#1}%
2039 }
```


user2

```
2040 \define@key{glossentry}{user2}{%  
2041   \def\@glo@userii{#1}%  
2042 }
```

user3

```
2043 \define@key{glossentry}{user3}{%  
2044   \def\@glo@useriii{#1}%  
2045 }
```

user4

```
2046 \define@key{glossentry}{user4}{%  
2047   \def\@glo@useriv{#1}%  
2048 }
```

user5

```
2049 \define@key{glossentry}{user5}{%  
2050   \def\@glo@userv{#1}%  
2051 }
```

user6

```
2052 \define@key{glossentry}{user6}{%  
2053   \def\@glo@uservi{#1}%  
2054 }
```

short This key is provided for use by \newacronym. It's not designed for general purpose use, so isn't described in the user manual.

```
2055 \define@key{glossentry}{short}{%  
2056   \def\@glo@short{#1}%  
2057 }
```

shortplural This key is provided for use by \newacronym.

```
2058 \define@key{glossentry}{shortplural}{%  
2059   \def\@glo@shortpl{#1}%  
2060 }
```

long This key is provided for use by \newacronym.

```
2061 \define@key{glossentry}{long}{%  
2062   \def\@glo@long{#1}%  
2063 }
```

longplural This key is provided for use by \newacronym.

```
2064 \define@key{glossentry}{longplural}{%  
2065   \def\@glo@longpl{#1}%  
2066 }
```

```

\@glsnname Define command to generate error if name key is missing.
2067 \newcommand*{\@glsnname}{%
2068   \PackageError{glossaries}{name key required in
2069   \string\newglossaryentry\space for entry '\@glo@label'}{You
2070   haven't specified the entry name}}

\@glsnodelsc Define command to generate error if description key is missing.
2071 \newcommand*\@glsnodelsc{%
2072   \PackageError{glossaries}
2073   {%
2074     description key required in \string\newglossaryentry\space
2075     for entry '\@glo@label'%
2076   }%
2077   {%
2078     You haven't specified the entry description%
2079   }%
2080 }%

lsdefaultplural Now obsolete. Don't use.
2081 \newcommand*{\@glsdefaultplural}{}

missingnumberlist Define a command to generate warning when numberlist not set.
2082 \newcommand*{\@gls@missingnumberlist}[1]{%
2083   ??%
2084   \ifglssavenumberlist
2085     \GlossariesWarning{Missing number list for entry '#1'.
2086     Maybe makeglossaries + rerun required}%
2087   \else
2088     \PackageError{glossaries}%
2089     {Package option 'savenumberlist=true' required}%
2090     {%
2091       You must use the 'savenumberlist' package option
2092       to reference location lists.%
2093     }%
2094   \fi
2095 }

@glsdefaultsort Define command to set default sort.
2096 \newcommand*{\@glsdefaultsort}{\@glo@name}

\gls@level Register to increment entry levels.
2097 \newcount\gls@level

@noexpand@field
2098 \newcommand{\@@gls@noexpand@field}[3]{%
2099   \expandafter\global\expandafter
2100   \let\csname glo@#1@#2\endcsname#3%
2101 }

```

noexpand@fields

```
2102 \newcommand{\@gls@noexpand@fields}[4]{%
2103   \ifcsdef{gls@assign@#3@field}
2104   {%
2105     \ifdefequal{#4}{\@gls@default@value}%
2106     {%
2107       \edef\@gls@value{\expandonce{#1}}%
2108       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
2109     }%
2110     {%
2111       \csuse{gls@assign@#3@field}{#2}{#4}%
2112     }%
2113   }%
2114   {%
2115     \ifdefequal{#4}{\@gls@default@value}%
2116     {%
2117       \edef\@gls@value{\expandonce{#1}}%
2118       \@gls@noexpand@field{#2}{#3}{\@gls@value}%
2119     }%
2120     {%
2121       \@gls@noexpand@field{#2}{#3}{#4}%
2122     }%
2123   }%
2124 }
```

ls@expand@field

```
2125 \newcommand{\@gls@expand@field}[3]{%
2126   \expandafter
2127   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
2128 }
```

s@expand@fields

```
2129 \newcommand{\@gls@expand@fields}[4]{%
2130   \ifcsdef{gls@assign@#3@field}
2131   {%
2132     \ifdefequal{#4}{\@gls@default@value}%
2133     {%
2134       \edef\@gls@value{\expandonce{#1}}%
2135       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
2136     }%
2137     {%
2138       \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
2139       {%
2140         \@gls@expand@field{#2}{#3}{#4}%
2141       }%
2142       {%
2143         \csuse{gls@assign@#3@field}{#2}{#4}%
2144       }%
2145     }%
2146   }%
2147 }
```

```

2145     }%
2146 }%
2147 {%
2148   \ifdefequal{#4}{\@gls@default@value}%
2149   {%
2150     \@gls@expand@field{#2}{#3}{#1}%
2151   }%
2152   {%
2153     \@gls@expand@field{#2}{#3}{#4}%
2154   }%
2155 }%
2156 }

```

switexpandonce

```

2157 \def\@gls@expandonce{\expandonce}
2158 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
2159   \def\@gls@tmp{#1}%
2160   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
2161 }

```

\gls@assign@field

```
\gls@assign@field{<def value>}{<label>}{<field>}{<tmp cs>}
```

Assigns an entry field. Expansion performed by default (except for name, symbol and description where backward compatibility required). If *<tmp cs>* is *<@gls@default@value>*, *<def value>* is used instead.

```
2162 \let\gls@assign@field\@gls@expand@fields
```

glsexpandfields Fully expand values when assigning fields (except for specific fields that are overridden by *\glssetnoexpandfield*).

```

2163 \newcommand*{\glsexpandfields}{%
2164   \let\gls@assign@field\@gls@expand@fields
2165 }

```

glsnoexpandfields Don't expand values when assigning fields (except for specific fields that are overridden by *\glssetexpandfield*).

```

2166 \newcommand*{\glsnoexpandfields}{%
2167   \let\gls@assign@field\@gls@noexpand@fields
2168 }

```

\newglossaryentry Define *\newglossaryentry {<label>}{<key-val list>}*. There are two required fields in *<key-val list>*: name (or parent) and description. (See above.)

```
2169 \newrobustcmd{\newglossaryentry}[2]{%
```

Check to see if this glossary entry has already been defined:

```

2170   \glsdoifnoexists{#1}%
2171   {%
2172     \gls@defglossaryentry{#1}{#2}%

```

```

2173 }%
2174 }

```

`\newglossaryentry` The definition of `\newglossaryentry` is changed at the start of the document environment. The `see` key doesn't work for entries that have been defined in the document environment.

```

2175 \newcommand*{\gls@defdocnewglossaryentry}{%
2176   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
2177   \let\newglossaryentry\new@glossaryentry
2178 }

```

`\deglossaryentry` Like `\newglossaryentry` but does nothing if the entry has already been defined.

```

2179 \newrobustcmd{\provideglossaryentry}[2]{%
2180   \ifglstryexists{#1}%
2181   {%
2182     %
2183     \gls@defglossaryentry{#1}{#2}%
2184   }%
2185 }
2186 \@onlypreamble{\provideglossaryentry}

```

`\w@glossaryentry` For use in document environment. This opens the `.glsdefs` file, if not already open, so that the entry definition can be saved for the next \TeX run. This means that any glossaries at the start of the document can access the entry information.

```

2187 \newrobustcmd{\new@glossaryentry}[2]{%
2188   \ifundef\@gls@deffile
2189   {%
2190     \global\newwrite\@gls@deffile
2191     \immediate\openout\@gls@deffile=\jobname.glsdefs
2192   }%
2193   {%
2194     \ifglstryexists{#1}{%
2195       %
2196       \gls@defglossaryentry{#1}{#2}%
2197     }%
2198     \@gls@writedef{#1}%
2199 }

```

At the start of the document input the `.glsdefs` file if it exists. This is now done by `\gls@begindocdefs`, which is redefined by `glossaries-extra`, so that this step can be skipped to avoid loading an obsolete `.glsdefs` file if the user switches to `glossaries-extra` with `docdef=restricted`.

```

2200 \AtBeginDocument{\gls@begindocdefs}

```

The end of the document needs to check if the `.glsdefs` file has been opened, in which case it needs to be closed.

```

2201 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}

```

`\ls@begindocdefs` Input the `.glsdefs` file if it exists and enable document definitions if permitted.

```

2202 \newcommand*{\gls@begindocdefs}{%
2203   \gls@enablesavenonumberlist
2204   \edef\gls@restoreat{\noexpand\catcode'\noexpand\@=\number\catcode'\@}\relax}%
2205   \makeatletter
2206   \InputIfFileExists{\jobname.glsdefs}{\gls@defdocnewglossaryentry}{\gls@restoreat}%
2207   \gls@restoreat
2208   \undef\gls@restoreat
2209   \gls@defdocnewglossaryentry
2210 }

```

`\gls@writedef` Writes glossary entry definition to `\gls@deffile`.

```

2211 \newcommand*{\gls@writedef}[1]{%
2212   \immediate\write\gls@deffile
2213   {%
2214     \string\ifglsentryexists{#1}\glspercentchar^^J%
2215     \expandafter\@gobble\string\{\glspercentchar^^J%
2216     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
2217     \expandafter\@gobble\string\{\glspercentchar%
2218   }%
2219   \for\gls@map:=\gls@keymap\do
2220   {%
2221     \letcs\gls@value{gls@detoklabel{#1}}\expandafter\@secondoftwo\gls@map}%
2222     \ifdef\gls@value
2223     {%
2224       \onelevel@sanitize\gls@value
2225       \immediate\write\gls@deffile
2226       {%
2227         \expandafter\@firstoftwo\gls@map
2228         =\expandafter\@gobble\string\{\gls@value\expandafter\@gobble\string\},%
2229         \glspercentchar
2230       }%
2231     }%
2232   }%
2233 }%

```

Provide hook:

```

2234 \gls@writedefhook
2235 \immediate\write\gls@deffile
2236 {%
2237   \glspercentchar^^J%
2238   \expandafter\@gobble\string\}\glspercentchar^^J%
2239   \expandafter\@gobble\string\}\glspercentchar%
2240 }%
2241 }

```

`\gls@keymap` List of entry definition key names and corresponding tag in control sequence used to store the value.

```

2242 \newcommand*{\gls@keymap}{%

```

```

2243 {name}{name},%
2244 {sort}{sortvalue},% unescaped sort value
2245 {type}{type},%
2246 {first}{first},%
2247 {firstplural}{firstpl},%
2248 {text}{text},%
2249 {plural}{plural},%
2250 {description}{desc},%
2251 {descriptionplural}{descplural},%
2252 {symbol}{symbol},%
2253 {symbolplural}{symbolplural},%
2254 {user1}{useri},%
2255 {user2}{userii},%
2256 {user3}{useriii},%
2257 {user4}{useriv},%
2258 {user5}{userv},%
2259 {user6}{uservi},%
2260 {long}{long},%
2261 {longplural}{longpl},%
2262 {short}{short},%
2263 {shortplural}{shortpl},%
2264 {counter}{counter},%
2265 {parent}{parent}%
2266 }

```

@gls@fetchfield

```
\@gls@fetchfield{<cs>}{<field>}
```

Fetches the internal field label from the given user *<field>* and stores in *<cs>*.

```
2267 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
2268 \edef\@gls@thisval{#2}%
```

Iterate through known mappings until we find the one for this field.

```

2269 \@for\@gls@map:=\@gls@keymap\do{%
2270 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2271 \ifdefequal{\@this@key}{\@gls@thisval}%
2272 {%

```

Found it.

```
2273 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```

2274 \@endfortrue
2275 }%
2276 {}%
2277 }%
2278 }

```

lsaddstoragekey

```
\glsaddstoragekey{<key>}{<default value>}{<no link cs>}
```

Similar to `\glsaddkey` but intended for keys whose values aren't explicitly used in the document, but might be required behind the scenes by other commands.

```
2279 \newcommand*{\glsaddstoragekey}{\ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```
2280 \newcommand*{\@sglsaddstoragekey}[1]{%
2281   \key@ifundefined{glossentry}{#1}%
2282   {%
2283     \expandafter\newcommand\expandafter*\expandafter
2284     {\csname gls@assign@#1@field\endcsname}[2]{%
2285       \@gls@expand@field{##1}{#1}{##2}%
2286     }%
2287   }%
2288   {}%
2289   \@glsaddstoragekey{#1}%
2290 }
```

Unstarred version doesn't override default expansion.

```
2291 \newcommand*{\@glsaddstoragekey}[3]{%
```

Check the specified key doesn't already exist.

```
2292   \key@ifundefined{glossentry}{#1}%
2293   {%
```

Set up the key.

```
2294     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2295     \appto\@gls@keymap{,{#1}{#1}}%
```

Set the default value.

```
2296     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2297     \appto\@newglossaryentryposthook{%
2298       \letcs{\@glo@tmp}{@glo@#1}%
2299       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2300     }%
```

Define the no-link commands.

```
2301     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2302   }%
2303   {%
2304     \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2305   }%
2306 }
```

\glsaddkey

```
\glsaddkey{<key>}{<default value>}{<no link cs>}{<no link ucfirst cs>}
{\link cs}{\link ucfirst cs}{\link allcaps cs}
```

Allow user to add their own custom keys.


```
2307 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
2308 \newcommand*{\@sglsaddkey}[1]{%
2309   \key@ifundefined{glossentry}{#1}%
2310   {%
2311     \expandafter\newcommand\expandafter*\expandafter
2312     {\csname gls@assign@#1@field\endcsname}[2]{%
2313       \@gls@expand@field{##1}{#1}{##2}%
2314     }%
2315   }%
2316   {}%
2317   \@glsaddkey{#1}%
2318 }
```

Unstarred version doesn't override default expansion.

```
2319 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2320   \key@ifundefined{glossentry}{#1}%
2321   {%
```

Set up the key.

```
2322     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2323     \appto\@gls@keymap{,{#1}{#1}}%
```

Set the default value.

```
2324     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2325     \appto\@newglossaryentryposthook{%
2326       \letcs{\@glo@tmp}{@glo@#1}%
2327       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2328     }%
```

Define the no-link commands.

```
2329     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2330     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

Now for the commands with links. First the version with no case change:

```
2331     \ifcsdef{@gls@user@#1@}%
2332     {%
2333       \PackageError{glossaries}%
2334       {Can't define '\string#5' as helper command
2335       '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
2336       {}%
2337     }%
2338     {%
2339       \expandafter\newcommand\expandafter*\expandafter
2340       {\csname @gls@user@#1\endcsname}[2][1]{%
2341         \new@ifnextchar[%
2342         {\csuse{@gls@user@#1@}{##1}{##2}}}%

```

```

2343         {\csuse{@gls@user@#1@}{##1}{##2} [] }}%
2344     \csdef{@gls@user@#1@}##1##2[##3]{%
2345         \@gls@field@link{##1}{##2}{#3{##2}##3}%
2346     }%
2347     \newrobustcmd*{#5}{%
2348         \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2349     }%

```

Next the version with the first letter converted to upper case:

```

2350     \ifcsdef{@Gls@user@#1@}%
2351     {%
2352         \PackageError{glossaries}%
2353         {Can't define '\string#6' as helper command
2354         '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
2355     }%
2356 }%
2357 {%
2358     \expandafter\newcommand\expandafter*\expandafter
2359     {\csname @Gls@user@#1\endcsname}[2] [] {%
2360         \new@ifnextchar[%
2361             {\csuse{@Gls@user@#1@}{##1}{##2}}%
2362             {\csuse{@Gls@user@#1@}{##1}{##2} [] }}%
2363     \csdef{@Gls@user@#1@}##1##2[##3]{%
2364         \@gls@field@link{##1}{##2}{#4{##2}##3}%
2365     }%
2366     \newrobustcmd*{#6}{%
2367         \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2368     }%

```

Finally the all caps version:

```

2369     \ifcsdef{@GLS@user@#1@}%
2370     {%
2371         \PackageError{glossaries}%
2372         {Can't define '\string#7' as helper command
2373         '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2374     }%
2375 }%
2376 {%
2377     \expandafter\newcommand\expandafter*\expandafter
2378     {\csname @GLS@user@#1\endcsname}[2] [] {%
2379         \new@ifnextchar[%
2380             {\csuse{@GLS@user@#1@}{##1}{##2}}%
2381             {\csuse{@GLS@user@#1@}{##1}{##2} [] }}%
2382     \csdef{@GLS@user@#1@}##1##2[##3]{%
2383         \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2384     }%
2385     \newrobustcmd*{#7}{%
2386         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%

```

```

2387 }%
2388 }%
2389 {%
2390 \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2391 }%
2392 }

```

`\glsfieldxdef` `\glsfieldxdef{<label>}{<field>}{<definition>}`

```

2393 \newcommand{\glsfieldxdef}[3]{%
2394 \glsdoifexists{#1}%
2395 {%
2396 \edef\@glo@label{\glsdetoklabel{#1}}%
2397 \ifcsdef{glo@\@glo@label @#2}%
2398 {%
2399 \protected@csxdef{glo@\@glo@label @#2}{#3}%
2400 }%
2401 {%
2402 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2403 }%
2404 }%
2405 }

```

`\glsfielddedef` `\glsfielddedef{<label>}{<field>}{<definition>}`

```

2406 \newcommand{\glsfielddedef}[3]{%
2407 \glsdoifexists{#1}%
2408 {%
2409 \edef\@glo@label{\glsdetoklabel{#1}}%
2410 \ifcsdef{glo@\@glo@label @#2}%
2411 {%
2412 \protected@csdedef{glo@\@glo@label @#2}{#3}%
2413 }%
2414 {%
2415 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2416 }%
2417 }%
2418 }

```

`\glsfieldgdef` `\glsfieldgdef{<label>}{<field>}{<definition>}`

```

2419 \newcommand{\glsfieldgdef}[3]{%
2420 \glsdoifexists{#1}%

```

```

2421 {%
2422   \edef\@glo@label{\glstoklabel{#1}}%
2423   \ifcsdef{glo@\@glo@label @#2}%
2424   {%
2425     \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2426   }%
2427   {%
2428     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2429   }%
2430 }%
2431 }

```

`\glsfielddef` `\glsfielddef{<label>}{<field>}{<definition>}`

```

2432 \newcommand{\glsfielddef}[3]{%
2433   \glsdoifexists{#1}%
2434   {%
2435     \edef\@glo@label{\glstoklabel{#1}}%
2436     \ifcsdef{glo@\@glo@label @#2}%
2437     {%
2438       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2439     }%
2440     {%
2441       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2442     }%
2443   }%
2444 }

```

`\glsfieldfetch` `\glsfieldfetch{<label>}{<field>}{<cs>}`

Fetches the value of the given field and stores in the given control sequence.

```

2445 \newcommand{\glsfieldfetch}[3]{%
2446   \glsdoifexists{#1}%
2447   {%
2448     \edef\@glo@label{\glstoklabel{#1}}%
2449     \ifcsdef{glo@\@glo@label @#2}%
2450     {%
2451       \letcs#3{glo@\@glo@label @#2}%
2452     }%
2453     {%
2454       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2455     }%
2456   }%
2457 }

```

`\ifglsfieldeq` `\ifglsfieldeq{<label>}{<field>}{<string>}{<true>}{<false>}`

Tests if the value of the given field is equal to the given string.

```

2458 \newcommand{\ifglsfieldeq}[5]{%
2459   \glsdoifexists{#1}%
2460   {%
2461     \edef\@glo@label{\glsdetoklabel{#1}}%
2462     \ifcsdef{glo@\@glo@label @#2}%
2463     {%
2464       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2465     }%
2466   }%
2467   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2468 }%
2469 }%
2470 }

```

`\ifglsfieldddefeq` `\ifglsfieldddefeq{<label>}{<field>}{<command>}{<true>}{<false>}`

Tests if the value of the given field is equal to the replacement text of the given command.

```

2471 \newcommand{\ifglsfieldddefeq}[5]{%
2472   \glsdoifexists{#1}%
2473   {%
2474     \edef\@glo@label{\glsdetoklabel{#1}}%
2475     \ifcsdef{glo@\@glo@label @#2}%
2476     {%
2477       \expandafter\ifdefstrequal
2478       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2479     }%
2480   }%
2481   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2482 }%
2483 }%
2484 }

```

`\ifglsfieldcseq` `\ifglsfieldcseq{<label>}{<field>}{<cs name>}{<true>}{<false>}`

As above but uses `\ifcsstrequal` instead of `\ifdefstrequal`

```

2485 \newcommand{\ifglsfieldcseq}[5]{%
2486   \glsdoifexists{#1}%
2487   {%
2488     \edef\@glo@label{\glsdetoklabel{#1}}%
2489     \ifcsdef{glo@\@glo@label @#2}%
2490     {%
2491       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2492     }%

```

```

2493   {%
2494       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2495   }%
2496 }%
2497 }

```

gls.writedefhook

```

2498 \newcommand*{\gls.writedefhook}{}

```

gls.assign@desc

```

2499 \newcommand*{\gls.assign@desc}[1]{%
2500   \gls.assign@field{}{#1}{desc}{\@glo@desc}%
2501   \gls.assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2502 }

```

newglossaryentry

```

2503 \newcommand{\longnewglossaryentry}[3]{%
2504   \glsdoifnoexists{#1}%
2505   {%
2506     \bgroup
2507     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2508     \long\def\@newglossaryentryprehook{%
2509       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2510       \@org@newglossaryentryprehook
2511     }%
2512     \renewcommand*{\gls.assign@desc}[1]{%
2513       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2514       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2515     }
2516     \gls@defglossaryentry{#1}{#2}%
2517   \egroup
2518 }
2519 }

```

Only allowed in the preamble. (Otherwise a long description could cause problems when writing the entry definition to the temporary file.)

```

2520 \@onlypreamble{\longnewglossaryentry}

```

deglossaryentry As the above but only defines the entry if it doesn’t already exist.

```

2521 \newcommand{\longprovideglossaryentry}[3]{%
2522   \ifglstryexists{#1}{}%
2523   {\longnewglossaryentry{#1}{#2}{#3}}%
2524 }
2525 \@onlypreamble{\longprovideglossaryentry}

```

defglossaryentry

```

\gls@defglossaryentry{<label>}{<key-val list>}

```

Defines a new entry without checking if it already exists.

```

2526 \newcommand{\gls@defglossaryentry}[2]{%

```

Prevent any further use of \GlsSetQuote:

```
2527 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2528 \edef\@glo@label{\glsdetoklabel{#1}}%
```

Provide a means for user defined keys to reference the label:

```
2529 \let\glslabel\@glo@label
```

Set up defaults. If the name or description keys are omitted, an error will be generated.

```
2530 \let\@glo@name\@gls@noname
```

```
2531 \let\@glo@desc\@gls@nodesc
```

```
2532 \let\@glo@descplural\@gls@default@value
```

```
2533 \let\@glo@type\@gls@default@value
```

```
2534 \let\@glo@symbol\@gls@default@value
```

```
2535 \let\@glo@symbolplural\@gls@default@value
```

```
2536 \let\@glo@text\@gls@default@value
```

```
2537 \let\@glo@plural\@gls@default@value
```

Using \let instead of \def to make later comparison avoid expansion issues. (Thanks to Ulrich Diez for suggesting this.)

```
2538 \let\@glo@first\@gls@default@value
```

```
2539 \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2540 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2541 \let\@glo@counter\@gls@default@value
```

```
2542 \def\@glo@see{}%
```

```
2543 \def\@glo@parent{}%
```

```
2544 \def\@glo@prefix{}%
```

Initialise nonnumberlist setting if we're in the document environment.

```
2545 \@gls@initnonumberlist
```

```
2546 \def\@glo@useri{}%
```

```
2547 \def\@glo@userii{}%
```

```
2548 \def\@glo@useriii{}%
```

```
2549 \def\@glo@useriv{}%
```

```
2550 \def\@glo@userv{}%
```

```
2551 \def\@glo@uservi{}%
```

```

2552 \def\@glo@short{}%
2553 \def\@glo@shortpl{}%
2554 \def\@glo@long{}%
2555 \def\@glo@longpl{}%

```

Add start hook in case another package wants to add extra keys.

```

2556 \@newglossaryentryprehook

```

Extract key-val information from third parameter:

```

2557 \setkeys{glossentry}{#2}%

```

Check there is a default glossary.

```

2558 \ifundef\glsdefaulttype
2559 {%
2560   \PackageError{glossaries}%
2561     {No default glossary type (have you used ‘nomain’ by mistake?)}%
2562     {If you use package option ‘nomain’ you must define
2563      a new glossary before you can define entries}%
2564 }%
2565 {}%

```

Assign type. This must be fully expandable

```

2566 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2567 \edef\@glo@type{\glsentrytype{\@glo@label}}%

```

Check to see if this glossary type has been defined, if it has, add this label to the relevant list, otherwise generate an error.

```

2568 \ifcsundef{glolist@\@glo@type}%
2569 {%
2570   \PackageError{glossaries}%
2571     {Glossary type ‘\@glo@type’ has not been defined}%
2572     {You need to define a new glossary type, before making entries
2573      in it}%
2574 }%
2575 {}%

```

Check if it's an ignored glossary

```

2576 \ifignoredglossary\@glo@type
2577 {%

```

The description may be omitted for an entry in an ignored glossary.

```

2578   \ifx\@glo@desc\@glsnodel
2579     \let\@glo@desc\@empty
2580   \fi
2581 }%
2582 {%
2583 }%
2584 \protected@edef\@glolist@\csname glolist@\@glo@type\endcsname}%
2585 \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
2586   \@glolist{\@glo@label},}%
2587 }%

```


Initialise level to 0.

```

2588 \gls@level=0\relax

```

Has this entry been assigned a parent?

```

2589 \ifx\@glo@parent\@empty

```

Doesn't have a parent. Set `\glo@<label>@parent` to empty.

```

2590 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2591 \else

```

Has a parent. Check to ensure this entry isn't its own parent.

```

2592 \ifdefequal\@glo@label\@glo@parent%
2593 {%
2594 \PackageError{glossaries}{Entry '@glo@label' can't be its own parent}{}%
2595 \def\@glo@parent{}%
2596 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2597 }%
2598 {%

```

Check the parent exists:

```

2599 \ifglentryexists{\@glo@parent}%
2600 {%

```

Parent exists. Set `\glo@<label>@parent`.

```

2601 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2602 \@glo@parent}%

```

Determine level.

```

2603 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2604 \advance\gls@level by 1\relax

```

If name hasn't been specified, use same as the parent name

```

2605 \ifx\@glo@name\@gls@name
2606 \expandafter\let\expandafter\@glo@name
2607 \csname glo@\@glo@parent @name\endcsname

```

If name and plural haven't been specified, use same as the parent

```

2608 \ifx\@glo@plural\@gls@default@value
2609 \expandafter\let\expandafter\@glo@plural
2610 \csname glo@\@glo@parent @plural\endcsname
2611 \fi
2612 \fi
2613 }%
2614 {%

```

Parent doesn't exist, so issue an error message and change this entry to have no parent

```

2615 \PackageError{glossaries}%
2616 {%
2617 Invalid parent '@glo@parent'
2618 for entry '@glo@label' - parent doesn't exist%
2619 }%
2620 {%
2621 Parent entries must be defined before their children%

```

```

2622     }%
2623     \def\@glo@parent{}%
2624     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2625     }%
2626     }%
2627 \fi

Set the level for this entry
2628 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

Define commands associated with this entry:
2629 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2630 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2631 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2632 \expandafter\gls@assign@field\expandafter
2633     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2634     {\@glo@label}{plural}{\@glo@plural}%
2635 \expandafter\gls@assign@field\expandafter
2636     {\csname glo@\@glo@label @text\endcsname}%
2637     {\@glo@label}{first}{\@glo@first}%

If first has been specified, make the default by appending \glspluralsuffix, otherwise
make the default the value of the plural key.
2638 \ifx\@glo@first\@gls@default@value
2639     \expandafter\gls@assign@field\expandafter
2640         {\csname glo@\@glo@label @plural\endcsname}%
2641         {\@glo@label}{firstpl}{\@glo@firstplural}%
2642 \else
2643     \expandafter\gls@assign@field\expandafter
2644         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2645         {\@glo@label}{firstpl}{\@glo@firstplural}%
2646 \fi

2647 \ifcsundef{@glotype@\@glo@type @counter}%
2648 {%
2649     \def\@glo@defaultcounter{\glscounter}%
2650 }%
2651 {%
2652     \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
2653 }%
2654 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2655 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2656 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2657 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2658 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2659 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2660 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2661 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2662 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2663 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2664 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%

```

```

2665 \ifx\@glo@name\@glsnoname
2666   \@glsnoname
2667   \let\@gloname\@gls@default@value
2668 \fi
2669 \gls@assign@field{}\@glo@label{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2670 \ifcsundef{glo@\@glo@label @numberlist}%
2671 {%
2672   \csxdef{glo@\@glo@label @numberlist}{%
2673     \noexpand\@gls@missingnumberlist{\@glo@label}}%
2674 }%
2675 {}%

```

Store nonumberlist setting if we're in the document environment.

```

2676 \@gls@storenonumberlist{\@glo@label}%

```

The smaller and smallcaps options set the description to \@glo@first. Need to check for this, otherwise it won't get expanded if the description gets sanitized.

```

2677 \def\@glo@@desc{\@glo@first}%
2678 \ifx\@glo@desc\@glo@@desc
2679   \let\@glo@desc\@glo@first
2680 \fi
2681 \ifx\@glo@desc\@glsnodelc
2682   \@glsnodelc
2683   \let\@glodelc\@gls@default@value
2684 \fi
2685 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2686 \@gls@defsort{\@glo@type}{\@glo@label}%

2687 \def\@glo@@symbol{\@glo@text}%
2688 \ifx\@glo@symbol\@glo@@symbol
2689   \let\@glo@symbol\@glo@text
2690 \fi
2691 \gls@assign@field{relax}{\@glo@label}{symbol}{\@glo@symbol}%
2692 \expandafter
2693   \gls@assign@field\expandafter
2694   {\csname glo@\@glo@label @symbol\endcsname}
2695   {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

Define an associated boolean variable to determine whether this entry has been used yet (needs to be defined globally):

```

2696 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2697   \noexpand\global
2698   \noexpand\let\expandafter\noexpand
2699   \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2700 }%
2701 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2702   \noexpand\global

```

```

2703      \noexpand\let\expandafter\noexpand
2704      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2705  }%
2706  \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```
2707  \@glo@autosee
```

Determine and store main part of the entry's index format.

```

2708  \ifignoredglossary\@glo@type
2709  {%
2710    \csdef{glo@\@glo@label @index}{}%
2711  }
2712  {%
2713    \do@glo@storeentry{\@glo@label}%
2714  }%

```

Define entry counters if enabled:

```
2715  \@newglossaryentry@defcounters
```

Add end hook in case another package wants to add extra keys.

```

2716  \@newglossaryentryposthook
2717 }

```

\@glo@autosee Automatically implement \glssee.

```

2718 \newcommand*{\@glo@autosee}{%
2719   \ifdefvoid\@glo@see{}%
2720   {%
2721     \protected@edef\@do@glssee{%
2722       \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2723       \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
2724     \@do@glssee
2725   }%
2726   \@glo@autoseehook
2727 }%

```

glo@autoseehook

```
2728 \newcommand*{\@glo@autoseehook}{}
```

aryentryprehook Allow extra information to be added to glossary entries:

```
2729 \newcommand*{\@newglossaryentryprehook}{}
```

ryentryposthook Allow extra information to be added to glossary entries:

```
2730 \newcommand*{\@newglossaryentryposthook}{}
```

try@defcounters

```
2731 \newcommand*{\@newglossaryentry@defcounters}{}
```

`\glsmoveentry` Moves entry whose label is given by first argument to the glossary named in the second argument.

```

2732 \newcommand*{\glsmoveentry}[2]{%
2733   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2734   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2735   \def\glo@list{,}%
2736   \forglsentries[\glo@type]{\glo@label}%
2737   {%
2738     \ifdefequal\@glo@thislabel\glo@label
2739       {\eappto\glo@list{\glo@label,}}%
2740     }%
2741     \cslet\glo@list@\glo@type{\glo@list}%
2742     \csdef\glo@\@glo@thislabel @type{#2}%
2743 }

```

`glossaryentryfield` Indicate what command should be used to display each entry in the glossary. (This enables the glossaries-accsupp package to use `\accsuppglossaryentryfield` instead.)

```

2744 \ifglxindy
2745   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2746 \else
2747   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2748 \fi

```

`glossarysubentryfield` Indicate what command should be used to display each subentry in the glossary. (This enables the glossaries-accsupp package to use `\accsuppglossarysubentryfield` instead.)

```

2749 \ifglxindy
2750   \newcommand*{\@glossarysubentryfield}{%
2751     \string\subglossentry}
2752 \else
2753   \newcommand*{\@glossarysubentryfield}{%
2754     \string\subglossentry}
2755 \fi

```

`@glo@storeentry` `\@glo@storeentry{<label>}`

Determine the format to write the entry in the glossary output (.glo) file. The argument is the entry's label (should already have been de-tok'ed if required). The result is stored in `\glo@<label>@index`, where `<label>` is the entry's label. (This doesn't include any formatting or location information.)

```

2756 \newcommand{\@glo@storeentry}[1]{%

```

Escape makeindex/xindy special characters in the label:

```

2757   \edef\@glo@esclabel{#1}%
2758   \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters

```

2759   \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2760   \@gls@checkmkidxchars\@glo@sort

```

Same again for the name string. Escape any special characters in the prefix

```
2761 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2762 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2763 \ifglxindy
```

Store using xindy syntax.

```
2764 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2765 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2766 (\string"\@glo@sort\string" %
2767 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2768 }%
2769 \else
```

Entry has a parent

```
2770 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2771 \csname glo@\@glo@parent @index\endcsname
2772 (\string"\@glo@sort\string" %
2773 \string"\@glo@prefix\@glossarysubentryfield
2774 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2775 }%
2776 \fi
2777 \else
```

Store using makeindex syntax.

```
2778 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
2779 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
2780 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2781 \@glo@sort\@gls@actualchar\@glo@prefix
2782 \@glossaryentryfield{\@glo@esclabel}%
2783 }%
2784 \else
```

Entry has a parent

```
2785 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2786 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2787 \@glo@sort\@gls@actualchar\@glo@prefix
2788 \@glossarysubentryfield
2789 {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2790 }%
2791 \fi
2792 \fi
2793 }
```

1.8 Resetting and unsetting entry flags

Each glossary entry is assigned a conditional of the form `\ifglo@<label>@flag` which determines whether or not the entry has been used (see also `\ifglsused` defined below). These flags can be set and unset using the following macros, but first we need to know if we're in `amsmath's` align environment's measuring pass.

`@ifnotmeasuring`

```
2794 \AtBeginDocument{%
2795   \ifpackageloaded{amsmath}%
2796   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2797   }{%
2798 }
2799 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2800   \ifmeasuring@
2801   \else
2802     #1%
2803   \fi
2804 }
2805 \newcommand*\gls@ifnotmeasuring[1]{#1}
```

`\glspatchtabularx` Patch `\TX@trial` (as per David Carlisle's answer in <http://tex.stackexchange.com/a/94895>). This does nothing if `\TX@trial` hasn't been defined.

```
2806 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2807   \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2808 }
2809 \newcommand*\glspatchtabularx{%
2810   \ifdef\TX@trial
2811   {%
2812     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2813     \let\glspatchtabularx\relax
2814   }%
2815   }%
2816 }
```

`\glsreset` The command `\glsreset{<label>}` can be used to set the entry flag to indicate that it hasn't been used yet. The required argument is the entry label.

```
2817 \newcommand*{\glsreset}[1]{%
2818   \gls@ifnotmeasuring
2819   {%
2820     \glsdoifexists{#1}%
2821     {%
2822       \@glsreset{#1}%
2823     }%
2824   }%
2825 }
```

`\glslocalreset` As above, but with only a local effect:

```

2826 \newcommand*{\glslocalreset}[1]{%
2827   \gls@ifnotmeasuring
2828   {%
2829     \glsdoifexists{#1}%
2830     {%
2831       \@glslocalreset{#1}%
2832     }%
2833   }%
2834 }

```

`\glsunset` The command `\glsunset{<label>}` can be used to set the entry flag to indicate that it has been used. The required argument is the entry label.

```

2835 \newcommand*{\glsunset}[1]{%
2836   \gls@ifnotmeasuring
2837   {%
2838     \glsdoifexists{#1}%
2839     {%
2840       \@glsunset{#1}%
2841     }%
2842   }%
2843 }

```

`\glslocalunset` As above, but with only a local effect:

```

2844 \newcommand*{\glslocalunset}[1]{%
2845   \gls@ifnotmeasuring
2846   {%
2847     \glsdoifexists{#1}%
2848     {%
2849       \@glslocalunset{#1}%
2850     }%
2851   }%
2852 }

```

`\@glslocalunset` Local unset. This defaults to just `\@glslocalunset` but is changed by `\glsenableentrycount`.

```

2853 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2854 \newcommand*{\@glslocalunset}[1]{%
2855   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2856 }

```

`\@glsunset` Global unset. This defaults to just `\@glsunset` but is changed by `\glsenableentrycount`.

```

2857 \newcommand*{\@glsunset}{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2858 \newcommand*{\@@glsunset}[1]{%
2859   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2860 }

```


`\@glslocalreset` Local reset. This defaults to just `\@@glslocalreset` but is changed by `\glsenableentrycount`.

```
2861 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`\@@glslocalreset` Local reset without checks.

```
2862 \newcommand*{\@@glslocalreset}[1]{%
2863   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2864 }
```

`\@glsreset` Global reset. This defaults to just `\@@glsreset` but is changed by `\glsenableentrycount`.

```
2865 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2866 \newcommand*{\@@glsreset}[1]{%
2867   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2868 }
```

Reset all entries for the named glossaries (supplied in a comma-separated list). Syntax:
`\glsresetall[⟨glossary-list⟩]`

`\glsresetall`

```
2869 \newcommand*{\glsresetall}[1][\@glo@types]{%
2870   \forallglsentries[#1]{\@glsentry}%
2871   {%
2872     \glsreset{\@glsentry}%
2873   }%
2874 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2875 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2876   \forallglsentries[#1]{\@glsentry}%
2877   {%
2878     \glslocalreset{\@glsentry}%
2879   }%
2880 }
```

Unset all entries for the named glossaries (supplied in a comma-separated list). Syntax:
`\glsunsetall[⟨glossary-list⟩]`

`\glsunsetall`

```
2881 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2882   \forallglsentries[#1]{\@glsentry}%
2883   {%
2884     \glsunset{\@glsentry}%
2885   }%
2886 }
```

As above, but with only a local effect:

lslocalunsetall

```
2887 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2888   \forallglsentries[#1]{\@glsentry}%
2889   {%
2890     \glslocalunset{\@glsentry}%
2891   }%
2892 }
```

1.9 Keeping Track of How Many Times an Entry Has Been Unset

Version 4.14 introduced `\glsenableentrycount` that keeps track of how many times an entry is marked as used. The counter is reset back to zero when the first use flag is reset. Note that although the word “counter” is used here, it’s not an actual \TeX counter or even an explicit \TeX count register but is just a macro. Any of the commands that use `\glsunset` or `\glslocalunset`, such as `\gls`, will automatically increment this value. Commands that don’t modify the first use flag (such as `\glstext` or `\glsentrytext`) don’t modify this value.

try@defcounters Define entry fields to keep track of how many times that entry has been marked as used.

```
2893 \newcommand*{\@newglossaryentry@defcounters}{%
2894   \csdef{glo@\@glo@label @currcount}{0}%
2895   \csdef{glo@\@glo@label @prevcount}{0}%
2896 }
```

nableentrycount Enables tracking of how many times an entry has been marked as used.

```
2897 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2898   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2899   \renewcommand*{\gls@defdocnewglossaryentry}{%
2900     \renewcommand*{\newglossaryentry}[2]{%
2901       \PackageError{glossaries}{\string\newglossaryentry\space
2902       may only be used in the preamble when entry counting has
2903       been activated}{If you use \string\glsenableentrycount\space
2904       you must place all entry definitions in the preamble not in
2905       the document environment}%
2906     }%
2907   }%
```

Define commands `\glsentrycurrcount` and `\glsentryprevcount` to access these new fields. Default to zero if undefined.

```
2908   \newcommand*{\glsentrycurrcount}[1]{%
2909     \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2910     {0}{\@gls@entry@field{##1}{currcount}}%
2911   }%
2912   \newcommand*{\glsentryprevcount}[1]{%
```

```

2913 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2914 {0}{\@gls@entry@field{##1}{prevcount}}}%
2915 }%

```

Make the unset and reset functions also increment or reset the entry counter.

```

2916 \renewcommand*{\@glsunset}[1]{%
2917   \@glsunset{##1}%
2918   \@gls@increment@currcount{##1}%
2919 }%
2920 \renewcommand*{\@glslocalunset}[1]{%
2921   \@glslocalunset{##1}%
2922   \@gls@local@increment@currcount{##1}%
2923 }%
2924 \renewcommand*{\@glsreset}[1]{%
2925   \@glsreset{##1}%
2926   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2927 }%
2928 \renewcommand*{\@glslocalreset}[1]{%
2929   \@glslocalreset{##1}%
2930   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2931 }%

```

Alter behaviour of \cgl's. (Only global unset is used if previous count was one as it doesn't make sense to have a local unset here given that the previous count was global.)

```

2932 \def\@cgl's@##1##2[##3]{%
2933   \ifnum\glsentryprevcount{##2}=1\relax
2934     \cgl'sformat{##2}{##3}%
2935     \glsunset{##2}%
2936   \else
2937     \@gls@{##1}{##2}[##3]%
2938   \fi
2939 }%

```

Similarly for the analogous commands. No case change plural:

```

2940 \def\@cgl'spl@##1##2[##3]{%
2941   \ifnum\glsentryprevcount{##2}=1\relax
2942     \cgl'splformat{##2}{##3}%
2943     \glsunset{##2}%
2944   \else
2945     \@cgl'spl@{##1}{##2}[##3]%
2946   \fi
2947 }%

```

First letter uppercase singular:

```

2948 \def\@cGls@##1##2[##3]{%
2949   \ifnum\glsentryprevcount{##2}=1\relax
2950     \cGlsformat{##2}{##3}%
2951     \glsunset{##2}%
2952   \else
2953     \@Gls@{##1}{##2}[##3]%
2954   \fi

```

2955 }%

First letter uppercase plural:

```
2956 \def\@cGlspl@##1##2[##3]{%
2957   \ifnum\glsentryprevcount{##2}=1\relax
2958     \cGlsplformat{##2}{##3}%
2959     \glsunset{##2}%
2960   \else
2961     \@Glspl@{##1}{##2}[##3]%
2962   \fi
2963 }%
```

Write information to aux file at the end of the document

```
2964 \AtEndDocument{\@gls@write@entrycounts}%
```

Fetch previous count information from aux file. (No check here to determine if the entry is still defined.)

```
2965 \renewcommand*{\@gls@entry@count}[2]{%
2966   \csxdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
2967 }%
```

\glsenableentrycount may only be used once and only in the preamble.

```
2968 \let\glsenableentrycount\relax
2969 }
2970 \@onlypreamble\glsenableentrycount
```

ement@currcount

```
2971 \newcommand*{\@gls@increment@currcount}[1]{%
2972   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2973     \number\numexpr\glsentrycurrcount{##1}+1}%
2974 }
```

ement@currcount

```
2975 \newcommand*{\@gls@local@increment@currcount}[1]{%
2976   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2977     \number\numexpr\glsentrycurrcount{##1}+1}%
2978 }
```

ite@entrycounts

Write the entry counts to the aux file. Use \immediate since this occurs right at the end of the document. Only write information for entries that have been used. (Some users have a file containing vast numbers of entries, many of which may not be used. There's no point writing information about the entries that haven't been used and it will only slow things down.)

```
2979 \newcommand*{\@gls@write@entrycounts}{%
2980   \immediate\write\@auxout
2981     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2982   \forallglsentries{\@glsentry}{%
2983     \ifglsused{\@glsentry}%
2984     {\immediate\write\@auxout
2985       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2986     }%
2987 }
```

```

2987 }%
2988 }

```

`\gls@entry@count` Default behaviour is to ignore arguments. Activated by `\glsenableentrycount`.

```

2989 \newcommand*{\@gls@entry@count}[2]{}

```

`\cgl` Define command that works like `\gls` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\gls` but issues a warning.)

```

2990 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}

```

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

```

2991 \newcommand*{\@cgl}[2][]{%
2992   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[]}%
2993 }

```

`\@cgl@` Read in the final optional argument. This defaults to same behaviour as `\gls` but issues a warning.

```

2994 \def\@cgl@#1#2[#3]{%
2995   \GlossariesWarning{\string\cgl\space is defaulting to
2996     \string\gls\space since you haven't enabled entry counting}%
2997   \@gls@{#1}{#2}[#3]%
2998 }

```

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

```

2999 \newcommand*{\cglformat}[2]{%
3000   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
3001 }

```

`\cGl` Define command that works like `\Gls` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\Gls` but issues a warning.)

```

3002 \newrobustcmd*{\cGl}{\@gls@hyp@opt\@cGl}

```

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

```

3003 \newcommand*{\@cGl}[2][]{%
3004   \new@ifnextchar[{\@cGl@{#1}{#2}}{\@cGl@{#1}{#2}[]}%
3005 }

```

`\@cGl@` Read in the final optional argument. This defaults to same behaviour as `\Gls` but issues a warning.

```

3006 \def\@cGl@#1#2[#3]{%
3007   \GlossariesWarning{\string\cGl\space is defaulting to
3008     \string\Gls\space since you haven't enabled entry counting}%
3009   \@Gls@{#1}{#2}[#3]%
3010 }

```

`\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.

```
3011 \newcommand*{\cGlsformat}[2]{%
3012   \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
3013 }
```

`\cglsp1` Define command that works like `\glsp1` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\glsp1` but issues a warning.)

```
3014 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}
```

`\@cglsp1` Defined the un-starred form. Need to determine if there is a final optional argument

```
3015 \newcommand*{\@cglsp1}[2][{}]{%
3016   \new@ifnextchar[\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2}[{}]}%
3017 }
```

`\@cglsp1@` Read in the final optional argument. This defaults to same behaviour as `\glsp1` but issues a warning.

```
3018 \def\@cglsp1@#1#2[#3]{%
3019   \GlossariesWarning{\string\cglsp1\space is defaulting to
3020     \string\glsp1\space since you haven't enabled entry counting}%
3021   \@glsp1@{#1}{#2}[#3]%
3022 }
```

`\cglsp1format` Format used by `\cglsp1` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
3023 \newcommand*{\cglsp1format}[2]{%
3024   \ifglshaslong{#1}{\glsp1long{#1}}{\glsp1firstplural{#1}}#2%
3025 }
```

`\cGlspl` Define command that works like `\Glspl` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\Glspl` but issues a warning.)

```
3026 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}
```

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

```
3027 \newcommand*{\@cGlspl}[2][{}]{%
3028   \new@ifnextchar[\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[{}]}%
3029 }
```

`\@cGlspl@` Read in the final optional argument. This defaults to same behaviour as `\Glspl` but issues a warning.

```
3030 \def\@cGlspl@#1#2[#3]{%
3031   \GlossariesWarning{\string\cGlspl\space is defaulting to
3032     \string\Glspl\space since you haven't enabled entry counting}%
3033   \@Glspl@{#1}{#2}[#3]%
3034 }
```

`\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.

```

3035 \newcommand*{\cGlsplformat}[2]{%
3036   \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
3037 }
```

1.10 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.¹

```
\loadglsentries[<type>]{<filename>}
```

This command will input the file using `\input`. The optional argument specifies to which glossary the entries should be assigned if they haven't used the type key. If the optional argument is not specified, the default glossary is used. Only those entries used in the document (via `\glslink`, `\gls`, `\glspl` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary). The mandatory argument is the filename (with or without `.tex` extension).

`\loadglsentries`

```

3038 \newcommand*{\loadglsentries}[2][\@gls@default]{%
3039   \let\@gls@default\glsdefaulttype
3040   \def\glsdefaulttype{#1}\input{#2}%
3041   \let\glsdefaulttype\@gls@default
3042 }
```

`\loadglsentries` can only be used in the preamble:

```
3043 \@onlypreamble{\loadglsentries}
```

1.11 Using glossary entries in the text

Any term that has been defined using `\newglossaryentry` (or `\newacronym`) can be displayed in the text (i.e. outside of the glossary) using one of the commands defined in this section. Unless you use `\glslink`, the way the term appears in the text is determined by `\glsdisplayfirst` (if it is the first time the term has been used) or `\glsdisplay` (for subsequent use). Any formatting commands (such as `\textbf` is governed by `\glstextformat`. By default this just displays the link text “as is”.

`\glstextformat`

```
3044 \newcommand*{\glstextformat}[1]{#1}
```

`\glsentryfmt` As from version 3.11a, the way in which an entry is displayed is now governed by `\glsentryfmt`. This doesn't take any arguments. The required information is set by commands like `\gls`. To

¹ and any other valid \LaTeX code that can be used in the preamble.

ensure backward compatibility, the default use the old `\glsdisplay` and `\glsdisplayfirst` style of commands

```
3045 \newcommand*{\glsentryfmt}{%
3046   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
3047 }
```

Format that provides backwards compatibility:

```
3048 \newcommand*{\@gls@default@entryfmt}[2]{%
3049   \ifdefempty\glscustomtext
3050   {%
3051     \glsifplural
3052     {%
```

Plural form

```
3053     \glscapscase
3054     {%
```

Don't adjust case

```
3055     \ifglsused\glslabel
3056     {%
```

Subsequent use

```
3057         #2{\glsentryplural{\glslabel}}%
3058         {\glsentrydescplural{\glslabel}}%
3059         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
3060     }%
3061     {%
```

First use

```
3062         #1{\glsentryfirstplural{\glslabel}}%
3063         {\glsentrydescplural{\glslabel}}%
3064         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
3065     }%
3066     }%
3067     {%
```

Make first letter upper case

```
3068     \ifglsused\glslabel
3069     {%
```

Subsequent use. (Expansion was used in version 3.07 and below in case the name wasn't the first thing to be displayed, but now the user can sort out the upper casing in `\defglsentryfmt`, which avoids the issues caused by fragile commands.)

```
3070     \ifbool{glscompatible-3.07}%
3071     {%
3072       \protected@edef\@glo@etext{%
3073         #2{\glsentryplural{\glslabel}}%
3074         {\glsentrydescplural{\glslabel}}%
3075         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3076       \xmakefirstuc\@glo@etext
3077     }%
```



```

3078      {%
3079      #2{\Glsentryplural{\glslabel}}%
3080      {\glsentrydescplural{\glslabel}}%
3081      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
3082      }%
3083      }%
3084      {%

```

First use

```

3085      \ifbool{glscompatible-3.07}%
3086      {%
3087      \protected@edef\@glo@etext{%
3088      #1{\Glsentryfirstplural{\glslabel}}%
3089      {\glsentrydescplural{\glslabel}}%
3090      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3091      \xmakefirstuc\@glo@etext
3092      }%
3093      {%
3094      #1{\Glsentryfirstplural{\glslabel}}%
3095      {\glsentrydescplural{\glslabel}}%
3096      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
3097      }%
3098      }%
3099      }%
3100      {%

```

Make all upper case

```

3101      \ifglsused\glslabel
3102      {%

```

Subsequent use

```

3103      \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
3104      {\glsentrydescplural{\glslabel}}%
3105      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3106      }%
3107      {%

```

First use

```

3108      \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
3109      {\glsentrydescplural{\glslabel}}%
3110      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3111      }%
3112      }%
3113      }%
3114      {%

```

Singular form

```

3115      \glscapscase
3116      {%

```

Don't adjust case

```

3117      \ifglsused\glslabel

```

```

3118      {%
Subsequent use
3119      #2{\glsentrytext{\glslabel}}%
3120      {\glsentrydesc{\glslabel}}%
3121      {\glsentrysymbol{\glslabel}}{\glsinsert}%
3122      }%
3123      {%

First use
3124      #1{\glsentryfirst{\glslabel}}%
3125      {\glsentrydesc{\glslabel}}%
3126      {\glsentrysymbol{\glslabel}}{\glsinsert}%
3127      }%
3128      }%
3129      {%

Make first letter upper case
3130      \ifglused\glslabel
3131      {%

Subsequent use
3132      \ifbool{glscompatible-3.07}%
3133      {%
3134      \protected@edef\@glo@etext{%
3135      #2{\glsentrytext{\glslabel}}%
3136      {\glsentrydesc{\glslabel}}%
3137      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3138      \xmakefirstuc\@glo@etext
3139      }%
3140      {%
3141      #2{\Glsentrytext{\glslabel}}%
3142      {\glsentrydesc{\glslabel}}%
3143      {\glsentrysymbol{\glslabel}}{\glsinsert}%
3144      }%
3145      }%
3146      {%

First use
3147      \ifbool{glscompatible-3.07}%
3148      {%
3149      \protected@edef\@glo@etext{%
3150      #1{\glsentryfirst{\glslabel}}%
3151      {\glsentrydesc{\glslabel}}%
3152      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3153      \xmakefirstuc\@glo@etext
3154      }%
3155      {%
3156      #1{\Glsentryfirst{\glslabel}}%
3157      {\glsentrydesc{\glslabel}}%
3158      {\glsentrysymbol{\glslabel}}{\glsinsert}%

```

```

3159         }%
3160     }%
3161 }%
3162 {%

    Make all upper case
3163     \ifglused\glslabel
3164     {%

        Subsequent use
3165         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
3166         {\glsentrydesc{\glslabel}}%
3167         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3168     }%
3169     {%

        First use
3170         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
3171         {\glsentrydesc{\glslabel}}%
3172         {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3173     }%
3174 }%
3175 }%
3176 }%
3177 {%

    Custom text provided in \glsdisp
3178     \ifglused{\glslabel}%
3179     {%

        Subsequent use
3180         #2{\glscustomtext}%
3181         {\glsentrydesc{\glslabel}}%
3182         {\glsentrysymbol{\glslabel}}{}%
3183     }%
3184     {%

        First use
3185         #1{\glscustomtext}%
3186         {\glsentrydesc{\glslabel}}%
3187         {\glsentrysymbol{\glslabel}}{}%
3188     }%
3189 }%
3190 }

```

`\glsentryfmt` Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

```

3191 \newcommand*{\glsentryfmt}{%
3192     \ifdefempty\glscustomtext
3193     {%
3194         \glsifplural
3195         {%

```

Plural form

```
3196      \glscapscase
3197      {%
```

Don't adjust case

```
3198      \ifglused\glslabel
3199      {%
```

Subsequent use

```
3200      \glentryplural{\glslabel}\glinsert
3201      }%
3202      {%
```

First use

```
3203      \glentryfirstplural{\glslabel}\glinsert
3204      }%
3205      }%
3206      {%
```

Make first letter upper case

```
3207      \ifglused\glslabel
3208      {%
```

Subsequent use.

```
3209      \Glsentryplural{\glslabel}\glinsert
3210      }%
3211      {%
```

First use

```
3212      \Glsentryfirstplural{\glslabel}\glinsert
3213      }%
3214      }%
3215      {%
```

Make all upper case

```
3216      \ifglused\glslabel
3217      {%
```

Subsequent use

```
3218      \mfirstucMakeUppercase
3219      {\glentryplural{\glslabel}\glinsert}%
3220      }%
3221      {%
```

First use

```
3222      \mfirstucMakeUppercase
3223      {\glentryfirstplural{\glslabel}\glinsert}%
3224      }%
3225      }%
3226      }%
3227      {%
```

Singular form

```
3228      \glscapscase
3229      {%
```

Don't adjust case

```
3230      \ifglused\glslabel
3231      {%
```

Subsequent use

```
3232      \glentrytext{\glslabel}\glsinsert
3233      }%
3234      {%
```

First use

```
3235      \glentryfirst{\glslabel}\glsinsert
3236      }%
3237      }%
3238      {%
```

Make first letter upper case

```
3239      \ifglused\glslabel
3240      {%
```

Subsequent use

```
3241      \Glsentrytext{\glslabel}\glsinsert
3242      }%
3243      {%
```

First use

```
3244      \Glsentryfirst{\glslabel}\glsinsert
3245      }%
3246      }%
3247      {%
```

Make all upper case

```
3248      \ifglused\glslabel
3249      {%
```

Subsequent use

```
3250      \mfirstucMakeUppercase{\glentrytext{\glslabel}\glsinsert}%
3251      }%
3252      {%
```

First use

```
3253      \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glsinsert}%
3254      }%
3255      }%
3256      }%
3257      }%
3258      {%
```

Custom text provided in \glstdisp. (The insert is most likely to be empty at this point.)

```
3259      \glscustomtext\glsinsert
```

```

3260 }%
3261 }

```

`\glsgenacfmt` Define a generic acronym format that uses the long and short keys (or their plurals) and `\acrfullformat`, `\firstacronymfont` and `\acronymfont`.

```

3262 \newcommand*{\glsgenacfmt}{%
3263   \ifdefempty\glscustomtext
3264     {%
3265       \ifglused\glslabel
3266         {%

```

Subsequent use:

```

3267     \glsifplural
3268     {%

```

Subsequent plural form:

```

3269     \glscapscase
3270     {%

```

Subsequent plural form, don't adjust case:

```

3271     \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
3272     }%
3273     {%

```

Subsequent plural form, make first letter upper case:

```

3274     \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
3275     }%
3276     {%

```

Subsequent plural form, all caps:

```

3277     \mfirstucMakeUppercase
3278     {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
3279     }%
3280     }%
3281     {%

```

Subsequent singular form

```

3282     \glscapscase
3283     {%

```

Subsequent singular form, don't adjust case:

```

3284     \acronymfont{\glsentryshort{\glslabel}}\glsinsert
3285     }%
3286     {%

```

Subsequent singular form, make first letter upper case:

```

3287     \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
3288     }%
3289     {%

```

Subsequent singular form, all caps:

```

3290     \mfirstucMakeUppercase
3291     {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%

```

3292 }%
 3293 }%
 3294 }%
 3295 {%

First use:

3296 \glsifplural
 3297 {%

First use plural form:

3298 \glscapscase
 3299 {%

First use plural form, don't adjust case:

3300 \genplacrformat{\glslabel}{\glsinsert}%
 3301 }%
 3302 {%

First use plural form, make first letter upper case:

3303 \Genplacrformat{\glslabel}{\glsinsert}%
 3304 }%
 3305 {%

First use plural form, all caps:

3306 \mfirstucMakeUppercase
 3307 {\genplacrformat{\glslabel}{\glsinsert}}}%
 3308 }%
 3309 }%
 3310 {%

First use singular form

3311 \glscapscase
 3312 {%

First use singular form, don't adjust case:

3313 \genacrformat{\glslabel}{\glsinsert}%
 3314 }%
 3315 {%

First use singular form, make first letter upper case:

3316 \Genacrformat{\glslabel}{\glsinsert}%
 3317 }%
 3318 {%

First use singular form, all caps:

3319 \mfirstucMakeUppercase
 3320 {\genacrformat{\glslabel}{\glsinsert}}}%
 3321 }%
 3322 }%
 3323 }%
 3324 }%
 3325 {%

User supplied text.

```
3326 \glscustomtext
3327 }%
3328 }
```

enacrfullformat `\genacrfullformat{<label>}{<insert>}`

The full format used by \glsgenacfmt (singular).

```
3329 \newcommand*{\genacrfullformat}[2]{%
3330 \glentrylong{#1}#2\space
3331 (\protect\firstacronymfont{\glentryshort{#1}})%
3332 }
```

enacrfullformat `\Genacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```
3333 \newcommand*{\Genacrfullformat}[2]{%
3334 \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3335 \xmakefirstuc\gls@text
3336 }
```

placrfullformat `\genplacrfullformat{<label>}{<insert>}`

The full format used by \glsgenacfmt (plural).

```
3337 \newcommand*{\genplacrfullformat}[2]{%
3338 \glentrylongpl{#1}#2\space
3339 (\protect\firstacronymfont{\glentryshortpl{#1}})%
3340 }
```

placrfullformat `\Genplacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```
3341 \newcommand*{\Genplacrfullformat}[2]{%
3342 \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3343 \xmakefirstuc\gls@text
3344 }
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
3345 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

\glsdisplay Deprecated. Kept for backward compatibility.

```
3346 \newcommand*{\glsdisplay}[4]{#1#4}
```


`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3347 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3348   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3349   Use \string\defglsentryfmt\space instead}%
3350   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3351   \edef\@gls@doentrydef{%
3352     \noexpand\defglsentryfmt[#1]{%
3353       \noexpand\ifcsdef{gls@#1@displayfirst}%
3354       {%
3355         \noexpand\@@gls@default@entryfmt
3356         {\noexpand\csuse{gls@#1@displayfirst}}}%
3357         {\noexpand\csuse{gls@#1@display}}}%
3358       }%
3359       {%
3360         \noexpand\@@gls@default@entryfmt
3361         {\noexpand\glsdisplayfirst}%
3362         {\noexpand\csuse{gls@#1@display}}}%
3363       }%
3364     }%
3365   }%
3366   \@gls@doentrydef
3367 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3368 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3369   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3370   Use \string\defglsentryfmt\space instead}%
3371   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3372   \edef\@gls@doentrydef{%
3373     \noexpand\defglsentryfmt[#1]{%
3374       \noexpand\ifcsdef{gls@#1@display}%
3375       {%
3376         \noexpand\@@gls@default@entryfmt
3377         {\noexpand\csuse{gls@#1@displayfirst}}}%
3378         {\noexpand\csuse{gls@#1@display}}}%
3379       }%
3380       {%
3381         \noexpand\@@gls@default@entryfmt
3382         {\noexpand\csuse{gls@#1@displayfirst}}}%
3383         {\noexpand\glsdisplay}%
3384       }%
3385     }%
3386   }%
3387   \@gls@doentrydef
3388 }
```

Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink` and `\glsdisp`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defglsentryfmt`). It goes against the \TeX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}['s]` rather than, say, `\gls[append='s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{<label>}` to ignore following spaces, so `\new@ifnextchar` from the package is required.

The following keys can be used in the first optional argument. The counter key checks that the value is the name of a valid counter.

```
3389 \define@key{glslink}{counter}{%
3390   \ifcsundef{c@#1}%
3391   {%
3392     \PackageError{glossaries}%
3393     {There is no counter called '#1'}%
3394     {%
3395       The counter key should have the name of a valid counter
3396       as its value%
3397     }%
3398   }%
3399   {%
3400     \def\@gls@counter{#1}%
3401   }%
3402 }
```

The value of the format key should be the name of a command (without the initial backslash) that has a single mandatory argument which can be used to format the associated entry number.

```
3403 \define@key{glslink}{format}{%
3404   \def\@glsnumberformat{#1}}
```

The hyper key is a boolean key, it can either have the value true or false, and indicates whether or not to make a hyperlink to the relevant glossary entry. If hyper is false, an entry will still be made in the glossary, but the given text won't be a hyperlink.

```
3405 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3406 \ifdef{\hyperlink}{\KV@glslink@hypertrue}{\KV@glslink@hyperfalse}
```

The local key is a boolean key. If true this indicates that commands such as `\gls` should only do a local reset rather than a global one.

```
3407 \define@boolkey{glslink}{local}[true]{}
```

The original `\glsifhyper` command isn't particularly useful as it makes more sense to check the actual hyperlink setting rather than testing whether the starred or unstarred version has been used. Therefore, as from version 4.08, `\glsifhyper` is deprecated in favour of

`\glsifhyperon`. In case there is a particular need to know whether the starred or unstarred version was used, provide a new command that determines whether the *-version, +-version or unmodified version was used.

```
\glslinkvar{<unmodified case>}{<star case>}{<plus case>}
```

`\glslinkvar` Initialise to unmodified case.

```
3408 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3409 \newcommand*{\glsifhyper}[2]{%
3410 \glslinkvar{#1}{#2}{#1}%
3411 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3412 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3413 }
```

`\@gls@hyp@opt` Used by the commands such as `\glslink` to determine whether to modify the hyper option.

```
3414 \newcommand*{\@gls@hyp@opt}[1]{%
3415 \let\glslinkvar\@firstofthree
3416 \let\@gls@hyp@opt@cs#1\relax
3417 \@ifstar{\s@gls@hyp@opt}%
3418 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3419 }
```

`\s@gls@hyp@opt` Starred version

```
3420 \newcommand*{\s@gls@hyp@opt}[1] []{%
3421 \let\glslinkvar\@secondofthree
3422 \@gls@hyp@opt@cs[hyper=false,#1]}
```

`\p@gls@hyp@opt` Plus version

```
3423 \newcommand*{\p@gls@hyp@opt}[1] []{%
3424 \let\glslinkvar\@thirdofthree
3425 \@gls@hyp@opt@cs[hyper=true,#1]}
```

Syntax:

```
\glslink[<options>]{<label>}{<text>}
```

Display `<text>` in the document, and add the entry information for `<label>` into the relevant glossary. The optional argument should be a key value list using the `\glslink` keys defined above.

There is also a starred version:

```
\glslink*{<options>}{<label>}{<text>}
```

which is equivalent to `\glslink[hyper=false,<options>]{<label>}{<text>}`

First determine which version is being used:

`\glslink`

```
3426 \newrobustcmd*{\glslink}{%  
3427 \@gls@hyp@opt\@gls@@link  
3428 }
```

`\@gls@@link` The main part of the business is in `\@gls@link` which shouldn't check if the term is defined as it's called by `\gls` etc which also perform that check.

```
3429 \newcommand*{\@gls@@link}[3][\@gls@link]{%  
3430 \glsdoifexistsordo{#2}%  
3431 {%  
3432 \let\do@gls@link@checkfirsthyper\relax  
3433 \@gls@link[#1]{#2}{#3}%  
3434 }%
```

Display the specified text. (The entry doesn't exist so there's nothing to link it to.)

```
3435 \glstextformat{#3}%  
3436 }%  
  
3437 \glspostlinkhook  
3438 }
```

`glspostlinkhook`

```
3439 \newcommand*{\glspostlinkhook}{}
```

`checkfirsthyper` Check for first use and switch off hyper key if hyperlink not wanted. (Should be off if first use and `hyper=false` is on or if first use and both the entry is in an acronym list and the `acrfootnote` setting is on.) This assumes the glossary type is stored in `\glstype` and the label is stored in `\glslabel`.

```
3440 \newcommand*{\@gls@link@checkfirsthyper}{%  
3441 \ifglsused{\glslabel}%  
3442 {%  
3443 }%  
3444 {%  
3445 \gls@checkisacronymlist\glstype  
3446 \ifglshyperfirst  
3447 \if@glsisacronymlist  
3448 \ifglsacrfootnote  
3449 \KV@glslink@hyperfalse  
3450 \fi  
3451 \fi  
3452 \else  
3453 \KV@glslink@hyperfalse  
3454 \fi  
3455 }%
```

Allow user to hook into this

```
3456 \glslinkcheckfirsthyperhook  
3457 }
```

`checkfirsthyperhook` Allow used to hook into the `\@gls@link@checkfirsthyper` macro
3458 `\newcommand*{\glslinkcheckfirsthyperhook}{}`

`linkpostsetkeys`
3459 `\newcommand*{\glslinkpostsetkeys}{}`

`\glsifhyperon` Check the value of the hyper key:
3460 `\newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}`

`disablehyperinlist` Disable hyperlink if in the “nohyper” list.
3461 `\newcommand*{\do@glssdisablehyperinlist}{%`
3462 `\expandafter\DTLifinlist\expandafter{\gls@type}{\@gls@nohyperlist}%`
3463 `{\KV@glslink@hyperfalse}{}}%`
3464 `}`

`\let@glslink@opts` Hook to set default options for `\@glslink`.
3465 `\newcommand*{\@gls@setdefault@glslink@opts}{}`

`\@gls@link`
3466 `\def\@gls@link[#1]#2#3{%`
Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with tabularx).
3467 `\leavevmode`
3468 `\edef\glslabel{\glsdetoklabel{#2}}%`
Save options in `\@gls@link@opts` and label in `\@gls@link@label`
3469 `\def\@gls@link@opts{#1}%`
3470 `\let\@gls@link@label\glslabel`
3471 `\def\@glsnumberformat{glsnumberformat}%`
3472 `\edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%`
If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default
3473 `\edef\gls@type{\csname glo@\glslabel @type\endcsname}%`
Save original setting
3474 `\let\org@ifKV@glslink@hyper\ifKV@glslink@hyper`
Set defaults:
3475 `\@gls@setdefault@glslink@opts`
Switch off hyper setting if the glossary type has been identified in nohyperlist.
3476 `\do@glssdisablehyperinlist`
Macros must set this before calling `\@gls@link`. The commands that check the first use flag should set this to `\@gls@link@checkfirsthyper` otherwise it should be set to `\relax`.
3477 `\do@gls@link@checkfirsthyper`
3478 `\setkeys{glslink}{#1}%`
Add a hook for the user to customise things after the keys have been set.
3479 `\glslinkpostsetkeys`

```

    Store the entry's counter in \theglsentrycounter
3480   \@gls@saveentrycounter

    Define sort key if necessary:
3481   \@gls@setsort{\glslabel}%

    (De-tok'ing done by \@do@wrglossary)
3482   \@do@wrglossary{#2}%
3483   \ifKV@glslink@hyper
3484     \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3485   \else

3486     \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3487   \fi

    Restore original setting
3488   \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3489 }

\glolinkprefix
3490 \newcommand*{\glolinkprefix}{glo:}

glsentrycounter  Set default value of entry counter
3491 \def\glsentrycounter{\glscounter}%

saveentrycounter  Need to check if using equation counter in align environment:
3492 \newcommand*{\@gls@saveentrycounter}{%
3493   \def\@gls@Hcounter{}}%

    Are we using equation counter?
3494   \ifthenelse{\equal{\@gls@counter}{equation}}{%
3495     {

        If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as
        may be inside an inner environment.)
3496     \ifcsundef{xatlevel@}%
3497     {%
3498       \edef\theglsentrycounter{\expandafter\noexpand
3499         \csname the\@gls@counter\endcsname}%
3500     }%
3501     {%
3502       \ifx\xatlevel@\@empty
3503         \edef\theglsentrycounter{\expandafter\noexpand
3504           \csname the\@gls@counter\endcsname}%
3505       \else
3506         \savecounters@
3507         \advance\c@equation by 1\relax
3508         \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3509     \ifcsundef{theH\@gls@counter}%
3510     {%
3511         \def\@gls@Hcounter{\theglsentrycounter}%
3512     }%
3513     {%
3514         \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3515     }%
3516     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3517     \restorecounters@
3518 \fi
3519 }%
3520 }%
3521 {%

```

Not using equation counter so no special measures:

```

3522     \edef\theglsentrycounter{\expandafter\noexpand
3523         \csname the\@gls@counter\endcsname}%
3524 }%

```

Check if hyperref version of this counter

```

3525 \ifx\@gls@Hcounter\@empty
3526     \ifcsundef{theH\@gls@counter}%
3527     {%
3528         \def\theHglentrycounter{\theglsentrycounter}%
3529     }%
3530     {%
3531         \protected@edef\theHglentrycounter{\expandafter\noexpand
3532             \csname theH\@gls@counter\endcsname}%
3533     }%
3534 \fi
3535 }

```

t@glo@numformat Set the formatting information in the format required by makeindex. The first argument is the format specified by the user (via the format key), the second argument is the name of the counter used to indicate the location, the third argument is a control sequence which stores the required format and the fourth argument (new to v3.0) is the hyper-prefix.

```

3536 \def\@set@glo@numformat#1#2#3#4{%
3537     \expandafter\@glo@check@mkidxrangechar#3\@nil
3538     \protected@edef#1{%
3539         \@glo@prefix setentrycounter[#4]{#2}%
3540         \expandafter\string\csname\@glo@suffix\endcsname
3541     }%
3542     \@gls@checkmkidxchars#1%
3543 }

```

Check to see if the given string starts with a (or). If it does set \@glo@prefix to the starting character, and \@glo@suffix to the rest (or glsnumberformat if there is nothing else), otherwise set \@glo@prefix to nothing and \@glo@suffix to all of it.

```

3544 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3545 \if#1(\relax
3546   \def\@glo@prefix{()%
3547   \if\relax#2\relax
3548     \def\@glo@suffix{glsnumberformat}%
3549   \else
3550     \def\@glo@suffix{#2}%
3551   \fi
3552 \else
3553   \if#1)\relax
3554     \def\@glo@prefix{}}}%
3555   \if\relax#2\relax
3556     \def\@glo@suffix{glsnumberformat}%
3557   \else
3558     \def\@glo@suffix{#2}%
3559   \fi
3560 \else
3561   \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3562 \fi
3563 \fi}

```

`\@gls@escbsdq` Escape backslashes and double quote marks. The argument must be a control sequence.

```

3564 \newcommand*{\@gls@escbsdq}[1]{%
3565   \def\@gls@checkedmkidx{%
3566     \let\gls@xdystring=#1\relax
3567     \@onelevel@sanitize\gls@xdystring
3568     \edef\do@gls@xdycheckbackslash{%
3569       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3570       \@backslashchar\@backslashchar\noexpand\null}%
3571     \do@gls@xdycheckbackslash
3572     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3573     \def\@gls@checkedmkidx{%
3574       \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3575       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

Unsanitize\gls@numberpage,\gls@alphpage,\gls@Alphpage and\gls@romanpage (thanks to David Carlisle for the suggestion.)

```

3576 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3577 {%
3578   \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3579   \@onelevel@sanitize\@gls@sanitized@tmp
3580   \edef\gls@dosubst{%
3581     \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3582     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3583   }%
3584   \gls@dosubst
3585 }%

```

Assign to required control sequence

```

3586 \let#1=\gls@xdystring

```


3587 }

Catch special characters (argument must be a control sequence):

checkmkidxchars

```
3588 \newcommand{\@gls@checkmkidxchars}[1]{%
3589   \ifglxsindy
3590     \@gls@escbsdq{#1}%
3591   \else
3592     \def\@gls@checkedmkidx{%
3593       \expandafter\@gls@checkquote#1\@nil""\null
3594       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3595       \def\@gls@checkedmkidx{%
3596         \expandafter\@gls@checkescquote#1\@nil\\"\null
3597         \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3598         \def\@gls@checkedmkidx{%
3599           \expandafter\@gls@checkescactual#1\@nil??\null
3600           \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3601           \def\@gls@checkedmkidx{%
3602             \expandafter\@gls@checkactual#1\@nil??\null
3603             \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3604             \def\@gls@checkedmkidx{%
3605               \expandafter\@gls@checkbar#1\@nil||\null
3606               \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3607               \def\@gls@checkedmkidx{%
3608                 \expandafter\@gls@checkescbar#1\@nil\\|\null
3609                 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3610                 \def\@gls@checkedmkidx{%
3611                   \expandafter\@gls@checklevel#1\@nil!!\null
3612                   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3613                 \fi
3614             }

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```
3615 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
3616 \newtoks\@gls@tmpb
```

@gls@checkquote Replace " with "" since " is a makeindex special character.

```
3617 \def\@gls@checkquote#1"#2"#3\@nil{%
3618   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3619   \toks@={#1}%
3620   \ifx\@nil#2\@nil
3621     \ifx\@nil#3\@nil
3622       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3623     \def\@gls@checkquote{\relax}%
3624   \else

```

```

3625 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3626 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3627 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
3628 \fi
3629 \else
3630 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3631 \@gls@quotechar\@gls@quotechar}%
3632 \ifx\null#3\null
3633 \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
3634 \else
3635 \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
3636 \fi
3637 \fi
3638 \@@gls@checkquote
3639 }

```

s@checkescquote Do the same for \":

```

3640 \def\@gls@checkescquote#1\"#2\"#3\null{%
3641 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3642 \toks@={#1}%
3643 \ifx\null#2\null
3644 \ifx\null#3\null
3645 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3646 \def\@@gls@checkescquote{\relax}%
3647 \else
3648 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3649 \@gls@quotechar\string\"@gls@quotechar
3650 \@gls@quotechar\string\"@gls@quotechar}%
3651 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3652 \fi
3653 \else
3654 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3655 \@gls@quotechar\string\"@gls@quotechar}%
3656 \ifx\null#3\null
3657 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3658 \else
3659 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3660 \fi
3661 \fi
3662 \@@gls@checkescquote
3663 }

```

@checkescactual Similarly for \? (which is replaces @ as makeindex's special character):

```

3664 \def\@gls@checkescactual#1\?#2\?#3\null{%
3665 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3666 \toks@={#1}%
3667 \ifx\null#2\null
3668 \ifx\null#3\null
3669 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```

```

3670 \def\@gls@checkescactual{\relax}%
3671 \else
3672 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3673 \@gls@quotechar\string"\@gls@actualchar
3674 \@gls@quotechar\string"\@gls@actualchar}%
3675 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3676 \fi
3677 \else
3678 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3679 \@gls@quotechar\string"\@gls@actualchar}%
3680 \ifx\null#3\null
3681 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3682 \else
3683 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3684 \fi
3685 \fi
3686 \@gls@checkescactual
3687 }

```

`gls@checkescbar` Similarly for `\|`:

```

3688 \def\@gls@checkescbar#1\|#2\|#3\null{%
3689 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3690 \toks@={#1}%
3691 \ifx\null#2\null
3692 \ifx\null#3\null
3693 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3694 \def\@gls@checkescbar{\relax}%
3695 \else
3696 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3697 \@gls@quotechar\string"\@gls@encapchar
3698 \@gls@quotechar\string"\@gls@encapchar}%
3699 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3700 \fi
3701 \else
3702 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3703 \@gls@quotechar\string"\@gls@encapchar}%
3704 \ifx\null#3\null
3705 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3706 \else
3707 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3708 \fi
3709 \fi
3710 \@gls@checkescbar
3711 }

```

`s@checkesclevel` Similarly for `\!`:

```

3712 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3713 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3714 \toks@={#1}%

```

```

3715 \ifx\null#2\null
3716 \ifx\null#3\null
3717 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3718 \def\@gls@checkesclevel{\relax}%
3719 \else
3720 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3721 \@gls@quotechar\string"\@gls@levelchar
3722 \@gls@quotechar\string"\@gls@levelchar}%
3723 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3724 \fi
3725 \else
3726 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3727 \@gls@quotechar\string"\@gls@levelchar}%
3728 \ifx\null#3\null
3729 \def\@gls@checkesclevel{\@gls@checkesclevel#2!!\null}%
3730 \else
3731 \def\@gls@checkesclevel{\@gls@checkesclevel#2!#3\null}%
3732 \fi
3733 \fi
3734 \@gls@checkesclevel
3735 }

```

\@gls@checkbar and for |:

```

3736 \def\@gls@checkbar#1|#2|#3\null{%
3737 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3738 \toks@={#1}%
3739 \ifx\null#2\null
3740 \ifx\null#3\null
3741 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3742 \def\@gls@checkbar{\relax}%
3743 \else
3744 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3745 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3746 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3747 \fi
3748 \else
3749 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3750 \@gls@quotechar\@gls@encapchar}%
3751 \ifx\null#3\null
3752 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3753 \else
3754 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3755 \fi
3756 \fi
3757 \@gls@checkbar
3758 }

```

@gls@checklevel and for !:

```

3759 \def\@gls@checklevel#1!#2!#3\null{%

```

```

3760 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3761 \toks@={#1}%
3762 \ifx\null#2\null
3763   \ifx\null#3\null
3764     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3765     \def\@gls@checklevel{\relax}%
3766   \else
3767     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3768       \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3769     \def\@gls@checklevel{\@gls@checklevel#3\null}%
3770   \fi
3771 \else
3772   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3773     \@gls@quotechar\@gls@levelchar}%
3774   \ifx\null#3\null
3775     \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3776   \else
3777     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3778   \fi
3779 \fi
3780 \@gls@checklevel
3781 }

```

gls@checkactual and for ?:

```

3782 \def\@gls@checkactual#1?#2?#3\null{%
3783   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3784   \toks@={#1}%
3785   \ifx\null#2\null
3786     \ifx\null#3\null
3787       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3788       \def\@gls@checkactual{\relax}%
3789     \else
3790       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3791         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3792       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3793     \fi
3794   \else
3795     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3796       \@gls@quotechar\@gls@actualchar}%
3797     \ifx\null#3\null
3798       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3799     \else
3800       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3801     \fi
3802   \fi
3803   \@gls@checkactual
3804 }

```

s@xdycheckquote As before but for use with xindy

```

3805 \def\@gls@xdycheckquote#1"#2"#3\null{%
3806   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3807   \toks@={#1}%
3808   \ifx\null#2\null
3809     \ifx\null#3\null
3810       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3811       \def\@gls@xdycheckquote{\relax}%
3812     \else
3813       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3814         \string"\string"}%
3815       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3816     \fi
3817   \else
3818     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3819       \string"}%
3820     \ifx\null#3\null
3821       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3822     \else
3823       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3824     \fi
3825   \fi
3826   \@gls@xdycheckquote
3827 }

```

ycheckbackslash Need to escape all backslashes for xindy. Define command that will define \@gls@xdycheckbackslash

```

3828 \edef\def\@gls@xdycheckbackslash{%
3829   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3830     ##2\@backslashchar##3\noexpand\null{%
3831     \noexpand\@gls@tmpb=\noexpand\expandafter
3832       {\noexpand\@gls@checkedmkidx}%
3833     \noexpand\toks@={##1}%
3834     \noexpand\ifx\noexpand\null##2\noexpand\null
3835       \noexpand\ifx\noexpand\null##3\noexpand\null
3836         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3837           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3838         \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3839       \noexpand\else
3840         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3841           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3842           \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3843       \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3844         \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3845       \noexpand\fi
3846     \noexpand\else
3847       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3848         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3849         \@backslashchar\@backslashchar}%
3850     \noexpand\ifx\noexpand\null##3\noexpand\null
3851       \noexpand\def\noexpand\@gls@xdycheckbackslash{%

```

```

3852     \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3853     \@backslashchar\noexpand\null}%
3854 \noexpand\else
3855     \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3856     \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3857     ##3\noexpand\null}%
3858 \noexpand\fi
3859 \noexpand\fi
3860 \noexpand\@gls@xdycheckbackslash
3861 }%
3862 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3863 \def@gls@xdycheckbackslash

```

\glsdohypertarget

```

3864 \newlength@gls@tmplen
3865 \newcommand*\glsdohypertarget}[2]{%
3866   \@glsshowtarget{#1}%
3867   \settoheight@gls@tmplen{#2}%
3868   \raisebox@gls@tmplen{\hypertarget{#1}{}}#2%
3869 }

```

\glsdohyperlink

```

3870 \newcommand*\glsdohyperlink}[2]{%
3871   \@glsshowtarget{#1}%
3872   \hyperlink{#1}{#2}%
3873 }

```

\glsdonohyperlink

```

3874 \newcommand*\glsdonohyperlink}[2]{#2}

```

\@glslink If \hyperlink is not defined \@glslink ignores its first argument and just does the second argument, otherwise it is equivalent to \hyperlink.

```

3875 \ifcsundef{hyperlink}%
3876 {%
3877   \let@glslink@glsdonohyperlink
3878 }%
3879 {%
3880   \let@glslink@glsdohyperlink
3881 }

```

\@glstarget If \hypertarget is not defined, \@glstarget ignores its first argument and just does the second argument, otherwise it is equivalent to \hypertarget.

```

3882 \ifcsundef{hypertarget}%
3883 {%
3884   \let@glstarget@secondoftwo
3885 }%

```

```

3886 {%
3887   \let\@glstarget\glstdohypertarget
3888 }

```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```

3889 \newcommand{\glsdisablehyper}{%
3890   \KV@glslink@hyperfalse
3891   \let\@glslink\glstdonohyperlink
3892   \let\@glstarget\@secondoftwo
3893 }

```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```

3894 \newcommand{\glsenablehyper}{%
3895   \KV@glslink@hypertrue
3896   \let\@glslink\glstdohyperlink
3897   \let\@glstarget\glstdohypertarget
3898 }

```

Provide some convenience commands if not already defined:

```

3899 \providecommand{\@firstofthree}[3]{#1}
3900 \providecommand{\@secondofthree}[3]{#2}

```

Syntax:

```
\gls[<options>]{<label>}[<insert text>]
```

Link to glossary entry using singular form. The link text is taken from the value of the text or first keys used when the entry was defined.

The first optional argument is a key-value list, the same as `\glslink`, the mandatory argument is the entry label. After the mandatory argument, there is another optional argument to insert extra text in the link text (the location of the inserted text is governed by `\glsdisplay` and `\glsdisplayfirst`). As with `\glslink` there is a starred version which is the same as the unstarred version but with the hyper key set to false. (Additional options can also be specified in the first optional argument.)

First determine which version is being used:

`\gls`

```
3901 \newrobustcmd*\gls{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```

3902 \newcommand*\@gls[2][ ]{%
3903   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}%
3904 }

```


`\@gls@` Read in the final optional argument:

```
3905 \def\@gls@#1#2[#3]{%
3906   \glsdoifexists{#2}%
3907   {%
3908     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3909     \let\glsifplural\@secondoftwo
3910     \let\glscapscase\@firstofthree
3911     \let\glscustomtext\@empty
3912     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3913   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3914   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3915   \ifKV@glslink@local
3916     \glslocalunset{#2}%
3917   \else
3918     \glsunset{#2}%
3919   \fi
3920 }%
```

```
3921 \glspostlinkhook
3922 }
```

`\Gls` behaves like `\gls`, but the first letter of the link text is converted to uppercase (note that if the first letter has an accent, the accented letter will need to be grouped when you define the entry). It is mainly intended for terms that start a sentence:

`\Gls`

```
3923 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3924 \newcommand*{\@Gls}[2][]{%
3925   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[]}%
3926 }
```

`\@Gls@` Read in the final optional argument:

```
3927 \def\@Gls@#1#2[#3]{%
3928   \glsdoifexists{#2}%
3929   {%
3930     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3931 \let\glsifplural\@secondoftwo
3932 \let\glsifcaps\@secondofthree
3933 \let\glsifcustomtext\@empty
3934 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \glo@text) Note that \gls@link sets \glstyp.

```

3935 \def\glo@text{\csname gls@\glstyp @entryfmt\endcsname}%

```

Call \gls@link If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3936 \gls@link{#1}{#2}{\glo@text}%

```

Indicate that this entry has now been used

```

3937 \ifKV@glslink@local
3938 \glslocalunset{#2}%
3939 \else
3940 \glsunset{#2}%
3941 \fi
3942 }%

```

```

3943 \glspostlinkhook
3944 }

```

\GLS behaves like \gls, but the link text is converted to uppercase:

\GLS

```

3945 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

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

```

3946 \newcommand*{\@GLS}[2] [] {%
3947 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} []}%
3948 }

```

\@GLS@ Read in the final optional argument:

```

3949 \def\@GLS@#1#2[#3] {%
3950 \glsdoifexists{#2}%
3951 {%
3952 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3953 \let\glsifplural\@secondoftwo
3954 \let\glsifcaps\@thirdofthree
3955 \let\glsifcustomtext\@empty
3956 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \glo@text). Note that \gls@link sets \glstyp.

```

3957 \def\glo@text{\csname gls@\glstyp @entryfmt\endcsname}%

```

Call \gls@link If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3958 \gls@link{#1}{#2}{\glo@text}%

```

Indicate that this entry has now been used

```
3959 \ifKV@glslink@local
3960 \glsllocalunset{#2}%
3961 \else
3962 \glsunset{#2}%
3963 \fi
3964 }%

3965 \glspostlinkhook
3966 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3967 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3968 \newcommand*{\@glspl}[2] []{%
3969 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2} []}%
3970 }
```

`\@glspl@` Read in the final optional argument:

```
3971 \def\@glspl@#1#2[#3]{%
3972 \glsdofexists{#2}%
3973 {%
3974 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3975 \let\glsifplural\@firstoftwo
3976 \let\glscapscase\@firstofthree
3977 \let\glscustomtext\@empty
3978 \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3979 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3980 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3981 \ifKV@glslink@local
3982 \glsllocalunset{#2}%
3983 \else
3984 \glsunset{#2}%
3985 \fi
3986 }%

3987 \glspostlinkhook
3988 }
```

`\Glspl` behaves in the same way as `\glsp1`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped).

`\Glspl`

```
3989 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3990 \newcommand*{\@Glspl}[2] [] {%
3991   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2} []}%
3992 }
```

`\@Glspl@` Read in the final optional argument:

```
3993 \def\@Glspl@#1#2[#3] {%
3994   \glsdoifexists{#2}%
3995   {%
3996     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3997     \let\glsifplural\@firstoftwo
3998     \let\glsapscase\@secondofthree
3999     \let\glscustomtext\@empty
4000   \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`). This needs to be expanded so that the `\@glo@text` can be passed to `\xmakefirstuc`. Note that `\@gls@link` sets `\glstype`.

```
4001   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
4002   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
4003   \ifKV@glslink@local
4004     \glslocalunset{#2}%
4005   \else
4006     \glsunset{#2}%
4007   \fi
4008 }%
```

```
4009 \glspostlinkhook
4010 }
```

`\GLSp1` behaves like `\glsp1` except that all the link text is converted to uppercase.

`\GLSp1`

```
4011 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
4012 \newcommand*{\@GLSp1}[2] [] {%
4013   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
4014 }
```

`\@GLSp1` Read in the final optional argument:

```

4015 \def\@GLSp1@#1#2[#3]{%
4016   \glsdoifexists{#2}%
4017   {%
4018     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

4019     \let\glsifplural\@firstoftwo
4020     \let\glscapscase\@thirdofthree
4021     \let\glscustomtext\@empty
4022     \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```

4023   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

4024   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

4025   \ifKV@glslink@local
4026     \glslocalunset{#2}%
4027   \else
4028     \glsunset{#2}%
4029   \fi
4030 }%

```

```

4031 \glspostlinkhook
4032 }

```

`\glsdisp` `\glsdisp[<options>]{<label>}{<text>}` This is like `\gls` except that the link text is provided. This differs from `\glslink` in that it uses `\glsdisplay` or `\glsdisplayfirst` and unsets the first use flag.

First determine if we are using the starred form:

```

4033 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

4034 \newcommand*{\@glsdisp}[3][ ]{%
4035   \glsdoifexists{#2}{%

4036     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

4037     \let\glsifplural\@secondoftwo
4038     \let\glscapscase\@firstofthree
4039     \def\glscustomtext{#3}%
4040     \def\glsinsert{}%

```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstyp`.

```
4041 \def\@glo@text{\csname gls@\glstyp @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
4042 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
4043 \ifKV@glslink@local
4044 \glslocalunset{#2}%
4045 \else
4046 \glsunset{#2}%
4047 \fi
4048 }%
```

```
4049 \glspostlinkhook
4050 }
```

`checkfirsthyper` Instead of just setting `\do@gls@link@checkfirsthyper` to `\relax` in `\@gls@field@link`, set it to `\@gls@link@nocheckfirsthyper` in case some other action needs to take place.

```
4051 \newcommand*{\@gls@link@nocheckfirsthyper}{}
```

`@gls@field@link`

```
4052 \newcommand{\@gls@field@link}[3]{%
4053 \glsdoifexists{#2}%
4054 {%
4055 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4056 \@gls@link[#1]{#2}{#3}%
4057 }%
4058 \glspostlinkhook
4059 }
```

`\glstext` behaves like `\gls` except it always uses the value given by the `text` key and it doesn't mark the entry as used.

`\glstext`

```
4060 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}
```

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

```
4061 \newcommand*{\@glstext}[2][{}]{%
4062 \new@ifnextchar[\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[{}]}}
```

Read in the final optional argument:

```
4063 \def\@glstext@#1#2[#3]{%
4064 \@gls@field@link{#1}{#2}{\glstext{#2}#3}%
4065 }
```

`\GLStext` behaves like `\glstext` except the text is converted to uppercase.

`\GLStext`

```
4066 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
4067 \newcommand*{\@GLStext}[2] [] {%
```

```
4068   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4069 \def\@GLStext@#1#2[#3] {%
```

```
4070   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glentrytext{#2}#3}}%
```

```
4071 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
4072 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
4073 \newcommand*{\@Glstext}[2] [] {%
```

```
4074   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4075 \def\@Glstext@#1#2[#3] {%
```

```
4076   \@gls@field@link{#1}{#2}{\glentrytext{#2}#3}}%
```

```
4077 }
```

`\glsfirst` behaves like `\gls` except it always uses the value given by the first key and it doesn't mark the entry as used.

`\glsfirst`

```
4078 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
4079 \newcommand*{\@glsfirst}[2] [] {%
```

```
4080   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4081 \def\@glsfirst@#1#2[#3] {%
```

```
4082   \@gls@field@link{#1}{#2}{\glentryfirst{#2}#3}}%
```

```
4083 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
4084 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
4085 \newcommand*{\@Glsfirst}[2] [] {%
```

```
4086   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4087 \def\@Glsfirst@#1#2[#3]{%
4088   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
4089 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
4090 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
4091 \newcommand*{\@GLSfirst}[2] [] {%
4092   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4093 \def\@GLSfirst@#1#2[#3]{%
4094   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryfirst{#2}#3}}%
4095 }
```

\glsplural behaves like \gls except it always uses the value given by the plural key and it doesn't mark the entry as used.

\glsplural

```
4096 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
4097 \newcommand*{\@glsplural}[2] [] {%
4098   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4099 \def\@glsplural@#1#2[#3]{%
4100   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
4101 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
4102 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
4103 \newcommand*{\@Glsplural}[2] [] {%
4104   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4105 \def\@Glsplural@#1#2[#3]{%
4106   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
4107 }
```

\GLSplural behaves like \glsplural except that the text is converted to uppercase.

\GLSplural

```
4108 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```


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

```
4109 \newcommand*{\@GLSplural}[2] [] {%  
4110   \new@ifnextchar [{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4111 \def\@GLSplural@#1#2[#3] {%  
4112   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}{#3}}}%  
4113 }
```

\glsfirstplural behaves like \gls except it always uses the value given by the firstplural key and it doesn't mark the entry as used.

\glsfirstplural

```
4114 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
4115 \newcommand*{\@glsfirstplural}[2] [] {%  
4116   \new@ifnextchar [{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4117 \def\@glsfirstplural@#1#2[#3] {%  
4118   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}{#3}}%  
4119 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
4120 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
4121 \newcommand*{\@Glsfirstplural}[2] [] {%  
4122   \new@ifnextchar [{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4123 \def\@Glsfirstplural@#1#2[#3] {%  
4124   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}{#3}}%  
4125 }
```

\GLSfirstplural behaves like \glsfirstplural except that the link text is converted to uppercase.

\GLSfirstplural

```
4126 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
4127 \newcommand*{\@GLSfirstplural}[2] [] {%  
4128   \new@ifnextchar [{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4129 \def\@GLSfirstplural@#1#2[#3] {%  
4130   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}{#3}}}%  
4131 }
```

`\glsname` behaves like `\gls` except it always uses the value given by the name key and it doesn't mark the entry as used.

`\glsname`

```
4132 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
4133 \newcommand*{\@glsname}[2] [] {%
```

```
4134   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4135 \def\@glsname@#1#2[#3] {%
```

```
4136   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
```

```
4137 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
4138 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
4139 \newcommand*{\@Glsname}[2] [] {%
```

```
4140   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4141 \def\@Glsname@#1#2[#3] {%
```

```
4142   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
```

```
4143 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
4144 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4145 \newcommand*{\@GLSname}[2] [] {%
```

```
4146   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4147 \def\@GLSname@#1#2[#3] {%
```

```
4148   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
```

```
4149 }
```

`\glsdesc` behaves like `\gls` except it always uses the value given by the description key and it doesn't mark the entry as used.

`\glsdesc`

```
4150 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
4151 \newcommand*{\@glsdesc}[2] [] {%
```

```
4152   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4153 \def\@glsdesc@#1#2[#3]{%
4154   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
4155 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
4156 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4157 \newcommand*{\@Glsdesc}[2][\%]
4158   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4159 \def\@Glsdesc@#1#2[#3]{%
4160   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
4161 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
4162 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4163 \newcommand*{\@GLSdesc}[2][\%]
4164   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4165 \def\@GLSdesc@#1#2[#3]{%
4166   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
4167 }
```

\glsdescplural behaves like \gls except it always uses the value given by the description-plural key and it doesn't mark the entry as used.

\glsdescplural

```
4168 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4169 \newcommand*{\@glsdescplural}[2][\%]
4170   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4171 \def\@glsdescplural@#1#2[#3]{%
4172   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
4173 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
4174 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4175 \newcommand*{\@Glsdescplural}[2] [] {%  
4176   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4177 \def\@Glsdescplural@#1#2[#3] {%  
4178   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%  
4179 }
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
4180 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4181 \newcommand*{\@GLSdescplural}[2] [] {%  
4182   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4183 \def\@GLSdescplural@#1#2[#3] {%  
4184   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%  
4185 }
```

`\glssymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glssymbol`

```
4186 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
4187 \newcommand*{\@glssymbol}[2] [] {%  
4188   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4189 \def\@glssymbol@#1#2[#3] {%  
4190   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}%  
4191 }
```

`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
4192 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4193 \newcommand*{\@Glssymbol}[2] [] {%  
4194   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4195 \def\@Glssymbol@#1#2[#3] {%  
4196   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}%  
4197 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
4198 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4199 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
4200   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4201 \def\@GLSsymbol@#1#2[#3] {%
```

```
4202   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}}%
```

```
4203 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the symbol-plural key and it doesn't mark the entry as used.

`glssymbolplural`

```
4204 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4205 \newcommand*{\@glssymbolplural}[2] [] {%
```

```
4206   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4207 \def\@glssymbolplural@#1#2[#3] {%
```

```
4208   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}}%
```

```
4209 }
```

`\Glsymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`Glsymbolplural`

```
4210 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4211 \newcommand*{\@Glsymbolplural}[2] [] {%
```

```
4212   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4213 \def\@Glsymbolplural@#1#2[#3] {%
```

```
4214   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}}%
```

```
4215 }
```

`\GLSsymbolplural` behaves like `\glssymbolplural` except that the link text is converted to uppercase.

`GLSsymbolplural`

```
4216 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4217 \newcommand*{\@GLSsymbolplural}[2] [] {%
```

```
4218   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4219 \def\@GLSsymbolplural@#1#2[#3]{%
4220   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
4221 }
```

`\glsuseri` behaves like `\gls` except it always uses the value given by the `user1` key and it doesn't mark the entry as used.

`\glsuseri`

```
4222 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4223 \newcommand*{\@glsuseri}[2] [] {%
4224   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4225 \def\@glsuseri@#1#2[#3]{%
4226   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
4227 }
```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```
4228 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4229 \newcommand*{\@Glsuseri}[2] [] {%
4230   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4231 \def\@Glsuseri@#1#2[#3]{%
4232   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
4233 }
```

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
4234 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4235 \newcommand*{\@GLSuseri}[2] [] {%
4236   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4237 \def\@GLSuseri@#1#2[#3]{%
4238   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
4239 }
```

`\glsuserii` behaves like `\gls` except it always uses the value given by the `user2` key and it doesn't mark the entry as used.

`\glsuserii`

```
4240 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
4241 \newcommand*{\@glsuserii}[2] [] {%  
4242   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4243 \def\@glsuserii@#1#2[#3] {%  
4244   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%  
4245 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
4246 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4247 \newcommand*{\@GLSuserii}[2] [] {%  
4248   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4249 \def\@GLSuserii@#1#2[#3] {%  
4250   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%  
4251 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
4252 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
4253 \newcommand*{\@GLSuserii}[2] [] {%  
4254   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4255 \def\@GLSuserii@#1#2[#3] {%  
4256   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%  
4257 }
```

\glsuseriii behaves like \gls except it always uses the value given by the user3 key and it doesn't mark the entry as used.

\glsuseriii

```
4258 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4259 \newcommand*{\@glsuseriii}[2] [] {%  
4260   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4261 \def\@glsuseriii@#1#2[#3] {%  
4262   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%  
4263 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
4264 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4265 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
4266   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
4267 \def\@Glsuseriii@#1#2[#3] {%
```

```
4268   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
4269 }
```

\Glsuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
4270 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4271 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
4272   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
4273 \def\@GLSuseriii@#1#2[#3] {%
```

```
4274   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
```

```
4275 }
```

\glsuseriv behaves like \gls except it always uses the value given by the user4 key and it doesn't mark the entry as used.

\glsuseriv

```
4276 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4277 \newcommand*{\@glsuseriv}[2] [] {%
```

```
4278   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]
```

Read in the final optional argument:

```
4279 \def\@glsuseriv@#1#2[#3] {%
```

```
4280   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
4281 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\GLSuseriv

```
4282 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4283 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
4284   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}]
```

Read in the final optional argument:

```
4285 \def\@GLSuseriv@#1#2[#3] {%
```

```
4286   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
4287 }
```


`\GLSuseriv` behaves like `\glsuseriv` except that the link text is converted to uppercase.

`\GLSuseriv`

```
4288 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4289 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
4290   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4291 \def\@GLSuseriv@#1#2[#3] {%
```

```
4292   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
```

```
4293 }
```

`\glsuserv` behaves like `\gls` except it always uses the value given by the `user5` key and it doesn't mark the entry as used.

`\glsuserv`

```
4294 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4295 \newcommand*{\@glsuserv}[2] [] {%
```

```
4296   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4297 \def\@glsuserv@#1#2[#3] {%
```

```
4298   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}}%
```

```
4299 }
```

`\Glsuserv` behaves like `\glsuserv` except that the first letter is converted to uppercase.

`\Glsuserv`

```
4300 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4301 \newcommand*{\@Glsuserv}[2] [] {%
```

```
4302   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4303 \def\@Glsuserv@#1#2[#3] {%
```

```
4304   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}}%
```

```
4305 }
```

`\GLSuserv` behaves like `\glsuserv` except that the link text is converted to uppercase.

`\GLSuserv`

```
4306 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4307 \newcommand*{\@GLSuserv}[2] [] {%
```

```
4308   \new@ifnextchar[{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4309 \def\@GLSuserv@#1#2[#3]{%
4310   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4311 }
```

\glsuservi behaves like \gls except it always uses the value given by the user6 key and it doesn't mark the entry as used.

\glsuservi

```
4312 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4313 \newcommand*{\@glsuservi}[2][{}]{%
4314   \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4315 \def\@glsuservi@#1#2[#3]{%
4316   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
4317 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4318 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4319 \newcommand*{\@Glsuservi}[2][{}]{%
4320   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4321 \def\@Glsuservi@#1#2[#3]{%
4322   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
4323 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4324 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4325 \newcommand*{\@GLSuservi}[2][{}]{%
4326   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4327 \def\@GLSuservi@#1#2[#3]{%
4328   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
4329 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4330 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\ns@acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4331 \newcommand*{\ns@acrshort}[2] [] {%
4332   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} [] }%
4333 }
```

Read in the final optional argument:

```
4334 \def\@acrshort#1#2[#3] {%
4335   \glsdoifexists{#2}%
4336   {%
4337     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4338     \let\glsifplural\@secondoftwo
4339     \let\glscapscase\@firstofthree
4340     \let\glsinsert\@empty
4341     \def\glscustomtext{%
4342       \acronymfont{\glentryshort{#2}}#3%
4343     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4344   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4345   }%
4346   \glspostlinkhook
4347 }
```

\Acrshort

```
4348 \newrobustcmd*{\Acrshort}{\@gl@hyp@opt\ns@Acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4349 \newcommand*{\ns@Acrshort}[2] [] {%
4350   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} [] }%
4351 }
```

Read in the final optional argument:

```
4352 \def\@Acrshort#1#2[#3] {%
4353   \glsdoifexists{#2}%
4354   {%
4355     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4356     \def\glslabel{#2}%
4357     \let\glsifplural\@secondoftwo
4358     \let\glscapscase\@secondofthree
4359     \let\glsinsert\@empty
4360     \def\glscustomtext{%
4361       \acronymfont{\Glsentryshort{#2}}#3%
4362     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4363   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4364   }%
```

```

4365 \glspostlinkhook
4366 }

```

\ACRshort

```

4367 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\@ns@ACRshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4368 \newcommand*{\ns@ACRshort}[2][\%
4369 \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
4370 }

```

Read in the final optional argument:

```

4371 \def\@ACRshort#1#2[#3]{%
4372 \glsdoifexists{#2}%
4373 {%
4374 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4375 \def\glslabel{#2}%
4376 \let\glsifplural\@secondoftwo
4377 \let\glsapscase\@thirdofthree
4378 \let\glsinsert\@empty
4379 \def\glscustomtext{%
4380 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4381 }%

```

Call \@gls@link Note that \@gls@link sets \gls@type.

```

4382 \@gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%
4383 }%
4384 \glspostlinkhook
4385 }

```

Short plural:

\acrshortpl

```

4386 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\@ns@acrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4387 \newcommand*{\ns@acrshortpl}[2][\%
4388 \new@ifnextchar[\@acrshortpl{#1}{#2}]{\@acrshortpl{#1}{#2}[]}%
4389 }

```

Read in the final optional argument:

```

4390 \def\@acrshortpl#1#2[#3]{%
4391 \glsdoifexists{#2}%
4392 {%
4393 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4394 \def\glslabel{#2}%
4395 \let\glsifplural\@firstoftwo
4396 \let\glsupcase\@firstofthree
4397 \let\glsinsert\@empty
4398 \def\glscustomtext{%
4399 \acronymfont{\glsentryshortpl{#2}}#3%
4400 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4401 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4402 }%

4403 \glspostlinkhook
4404 }

```

\Acrshortpl

```

4405 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4406 \newcommand*{\ns@Acrshortpl}[2] [] {%
4407 \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
4408 }

Read in the final optional argument:
4409 \def\@Acrshortpl#1#2[#3] {%
4410 \glsdoifexists{#2}%
4411 {%

4412 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4413 \def\glslabel{#2}%
4414 \let\glsifplural\@firstoftwo
4415 \let\glsupcase\@secondofthree
4416 \let\glsinsert\@empty
4417 \def\glscustomtext{%
4418 \acronymfont{\Glsentryshortpl{#2}}#3%
4419 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4420 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4421 }%

4422 \glspostlinkhook
4423 }

```

\ACRshortpl

```

4424 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4425 \newcommand*{\ns@ACRshortpl}[2] [] {%
4426 \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
4427 }

```

Read in the final optional argument:

```
4428 \def\@ACRshortpl#1#2[#3]{%
4429   \glsdoifexists{#2}%
4430   {%
4431     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4432     \def\glslabel{#2}%
4433     \let\glsifplural\@firstoftwo
4434     \let\glscapscase\@thirdofthree
4435     \let\glsinsert\@empty
4436     \def\glscustomtext{%
4437       \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
4438     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4439   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4440   }%
4441   \glspostlinkhook
4442 }
```

\acrlong

```
4443 \newrobustcmd*{\acrlong}{\@gl@hyp@opt\@ns@acrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4444 \newcommand*{\@ns@acrlong}[2][ ]{%
4445   \new@ifnextchar{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[ ]}%
4446 }
```

Read in the final optional argument:

```
4447 \def\@acrlong#1#2[#3]{%
4448   \glsdoifexists{#2}%
4449   {%
4450     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4451     \def\glslabel{#2}%
4452     \let\glsifplural\@secondoftwo
4453     \let\glscapscase\@firstofthree
4454     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4455   \def\glscustomtext{%
4456     \glsentrylong{#2}#3%
4457   }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4458   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4459   }%
```

```

4460 \glspostlinkhook
4461 }

```

\Acrlong

```

4462 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4463 \newcommand*{\ns@Acrlong}[2][\@gls@hyp@opt\@ns@Acrlong]{%
4464   \new@ifnextchar[\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[\@gls@hyp@opt\@ns@Acrlong]}%
4465 }

```

Read in the final optional argument:

```

4466 \def\@Acrlong#1#2[#3]{%
4467   \glsdoifexists{#2}%
4468   {%
4469     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4470     \def\glslabel{#2}%
4471     \let\glsifplural\@secondoftwo
4472     \let\glsapscase\@secondofthree
4473     \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4474   \def\glscustomtext{%
4475     \Glsentrylong{#2}#3%
4476   }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4477   \@gls@link{#1}{#2}{\csname gls@glstype @entryfmt\endcsname}%
4478   }%
4479   \glspostlinkhook
4480 }

```

\ACRlong

```

4481 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\@ns@ACRlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4482 \newcommand*{\ns@ACRlong}[2][\@gls@hyp@opt\@ns@ACRlong]{%
4483   \new@ifnextchar[\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[\@gls@hyp@opt\@ns@ACRlong]}%
4484 }

```

Read in the final optional argument:

```

4485 \def\@ACRlong#1#2[#3]{%
4486   \glsdoifexists{#2}%
4487   {%
4488     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4489 \def\glslabel{#2}%
4490 \let\glsifplural\@secondoftwo
4491 \let\glsupcase\@thirdofthree
4492 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4493 \def\glscustomtext{%
4494 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4495 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4496 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4497 }%

```

```

4498 \glspostlinkhook
4499 }

```

Short plural:

\acrlongpl

```

4500 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4501 \newcommand*{\ns@acrlongpl}[2][\%
4502 \new@ifnextchar[\@acrlongpl{#1}{#2}]{\@acrlongpl{#1}{#2}[]}%
4503 }

```

Read in the final optional argument:

```

4504 \def\@acrlongpl#1#2[#3]{%
4505 \glsdoifexists{#2}%
4506 {%
4507 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4508 \def\glslabel{#2}%
4509 \let\glsifplural\@firstoftwo
4510 \let\glsupcase\@firstofthree
4511 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4512 \def\glscustomtext{%
4513 \glsentrylongpl{#2}#3%
4514 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4515 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4516 }%

```

```

4517 \glspostlinkhook
4518 }

```


\Acrlongpl

```
4519 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\@ns@Acrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4520 \newcommand*{\ns@Acrlongpl}[2][\%  
4521 \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%  
4522 }
```

Read in the final optional argument:

```
4523 \def\@Acrlongpl#1#2[#3]{%  
4524 \glsdoifexists{#2}%  
4525 {%  
  
4526 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4527 \def\glslabel{#2}%  
4528 \let\glsifplural\@firstoftwo  
4529 \let\glsapscase\@secondofthree  
4530 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4531 \def\glscustomtext{%  
4532 \Glsentrylongpl{#2}#3%  
4533 }%
```

Call \@gls@link. Note that \@gls@link sets \glsstyle.

```
4534 \@gls@link[#1]{#2}{\csname gls@glstyle @entryfmt\endcsname}%  
4535 }%  
  
4536 \glspostlinkhook  
4537 }
```

\ACRlongpl

```
4538 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4539 \newcommand*{\ns@ACRlongpl}[2][\%  
4540 \new@ifnextchar[\@ACRlongpl{#1}{#2}]{\@ACRlongpl{#1}{#2}[]}%  
4541 }
```

Read in the final optional argument:

```
4542 \def\@ACRlongpl#1#2[#3]{%  
4543 \glsdoifexists{#2}%  
4544 {%  
  
4545 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4546 \def\glslabel{#2}%  
4547 \let\glsifplural\@firstoftwo  
4548 \let\glsapscase\@thirdofthree  
4549 \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```

4550 \def\glscustomtext{%
4551 \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
4552 }%

Call \@gls@link. Note that \@gls@link sets \glstype.

4553 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4554 }%

4555 \glspostlinkhook
4556 }

```

Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

`\gls@entry@field` Generic version.

```
\@gls@entry@field{<label>}{<field>}
```

```

4557 \newcommand*{\@gls@entry@field}[2]{%
4558 \csname glo@\glsdetoklabel{#1}@#2\endcsname
4559 }

```

`\glsletentryfield` `\glsletentryfield{<cs>}{<label>}{<field>}`

```

4560 \newcommand*{\glsletentryfield}[3]{%
4561 \letcs\@glo@text{glo@\glsdetoklabel{#2}@#3}%
4562 }

```

`\Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{<label>}{<field>}
```

```

4563 \newcommand*{\@Gls@entry@field}[2]{%
4564 \glsdoifexistsordo{#1}%
4565 {%
4566 \letcs\@glo@text{glo@\glsdetoklabel{#1}@#2}%
4567 \ifdef\@glo@text
4568 {%
4569 \xmakefirstuc{\@glo@text}%
4570 }%
4571 }%
4572 ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4573 entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry

```

```

4574     label and the field name}%
4575 }%
4576 }%
4577 {%
4578   ??%
4579 }%
4580 }

```

Get the entry name (as specified by the name key when the entry was defined). The argument is the label associated with the entry. Note that unless you used `name=false` in the `sanitize` package option you may get unexpected results if the name key contains any commands.

`\glsentryname`

```

4581 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4582 \newrobustcmd*{\Glsentryname}[1]{%
4583   \@Gls@entryname{#1}%
4584 }

```

`\@Gls@entryname` This is a workaround in the event that the user defies the warning in the manual about not using `\Glsname` or `\Glsentryname` with acronyms. First the default behaviour:

```

4585 \newcommand*{\@Gls@entryname}[1]{%
4586   \@Gls@entry@field{#1}{name}%
4587 }

```

`\ls@acentryname` Now the behaviour when `\setacronymstyle` is used:

```

4588 \newcommand*{\@Gls@acentryname}[1]{%
4589   \ifglshaslong{#1}%
4590   {%
4591     \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
4592     \@gls@getbody is defined by mfirstuc (which used to be part of glossaries).
4593     \expandafter\@gls@getbody\@glo@text{}\@nil
4594     \expandafter\ifx\@gls@body\glsentrylong\relax
4595       \expandafter\Glsentrylong\@gls@rest
4596     \else
4597       \expandafter\ifx\@gls@body\glsentryshort\relax
4598         \expandafter\Glsentryshort\@gls@rest
4599       \else
4600         \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make `\glsentryshort` behave like `\Glsentryshort`. (This is on the assumption that the argument of `\acronymfont` is `\glsentryshort{⟨label⟩}`, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

4600     {%
4601       \let\glsentryshort\Glsentryshort
4602       \@glo@text

```

```

4603     }%
4604   \else

4605     \expandafter\ifx\@gls@body\glsshortaccessdisplay\relax
4606     {%
4607       \let\glsentryshort\Glsentryshort
4608       \@glo@text
4609     }%
4610   \else
4611     \xmakefirstuc{\@glo@text}%
4612   \fi
4613 \fi
4614 \fi
4615 \fi
4616 }%
4617 {%

```

Not an acronym

```

4618   \@Gls@entry@field{#1}{name}%
4619 }%
4620 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

`\glsentrydesc`

```

4621 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4622 \newrobustcmd*{\Glsentrydesc}[1]{%
4623   \@Gls@entry@field{#1}{desc}%
4624 }

```

Plural form:

`entrydescplural`

```

4625 \newcommand*{\glsentrydescplural}[1]{%
4626   \@gls@entry@field{#1}{descplural}%
4627 }

```

`entrydescplural`

```

4628 \newrobustcmd*{\Glsentrydescplural}[1]{%
4629   \@Gls@entry@field{#1}{descplural}%
4630 }

```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

`\glsentrytext`

```
4631 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4632 \newrobustcmd*{\Glsentrytext}[1]{%
4633   \@Gls@entry@field{#1}{text}%
4634 }
```

Get the plural form:

`\glsentryplural`

```
4635 \newcommand*{\glsentryplural}[1]{%
4636   \@gls@entry@field{#1}{plural}%
4637 }
```

`\Glsentryplural`

```
4638 \newrobustcmd*{\Glsentryplural}[1]{%
4639   \@Gls@entry@field{#1}{plural}%
4640 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4641 \newcommand*{\glsentrysymbol}[1]{%
4642   \@gls@entry@field{#1}{symbol}%
4643 }
```

`\Glsentrysymbol`

```
4644 \newrobustcmd*{\Glsentrysymbol}[1]{%
4645   \@Gls@entry@field{#1}{symbol}%
4646 }
```

Plural form:

`\glsentrysymbolplural`

```
4647 \newcommand*{\glsentrysymbolplural}[1]{%
4648   \@gls@entry@field{#1}{symbolplural}%
4649 }
```

`\Glsentrysymbolplural`

```
4650 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4651   \@Gls@entry@field{#1}{symbolplural}%
4652 }
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

`\glsentryfirst`

```
4653 \newcommand*{\glsentryfirst}[1]{%
4654   \@gls@entry@field{#1}{first}%
4655 }
```

`\Glsentryfirst`

```
4656 \newrobustcmd*{\Glsentryfirst}[1]{%
4657   \@Gls@entry@field{#1}{first}%
4658 }
```

Get the plural form (as specified by the `firstplural` key when the entry was defined).

`\glsentryfirstplural`

```
4659 \newcommand*{\glsentryfirstplural}[1]{%
4660   \@gls@entry@field{#1}{firstpl}%
4661 }
```

`\Glsentryfirstplural`

```
4662 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4663   \@Gls@entry@field{#1}{firstpl}%
4664 }
```

`\glsentrytitlecase`

```
4665 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4666   \glsdoifexists{#1}%
4667   {%
4668     \glsfieldfetch{#1}{#2}{\@gls@value}%
4669     \xcapitalisewords{\@gls@value}%
4670   }%
4671 }
4672 \ifdef\texorpdfstring
4673 {
4674   \newcommand*{\glsentrytitlecase}[2]{%
4675     \texorpdfstring
4676       {\@glsentrytitlecase{#1}{#2}}%
4677       {\@gls@entry@field{#1}{#2}}%
4678   }
4679 }
4680 {
4681   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4682 }
```

Display the glossary type with which this entry is associated (as specified by the `type` key used when the entry was defined)

`\glsentrytype`

```
4683 \newcommand*{\glsentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is `sanitize`, so unexpected results may occur if the sort key contained commands.

```

\glentrysort
4684 \newcommand*{\glentrysort}[1]{%
4685   \@gls@entry@field{#1}{sort}%
4686 }

\glentryparent  Expands to the label of the entry's parent.
4687 \newcommand*{\glentryparent}[1]{%
4688   \@gls@entry@field{#1}{parent}%
4689 }

\glentryuseri  Get the first user key (as specified by the user1 when the entry was defined). The argument is
                the label associated with the entry.
4690 \newcommand*{\glentryuseri}[1]{%
4691   \@gls@entry@field{#1}{useri}%
4692 }

\Glsentryuseri
4693 \newrobustcmd*{\Glsentryuseri}[1]{%
4694   \@Gls@entry@field{#1}{useri}%
4695 }

\glentryuserii  Get the second user key (as specified by the user2 when the entry was defined). The argument
                is the label associated with the entry.
4696 \newcommand*{\glentryuserii}[1]{%
4697   \@gls@entry@field{#1}{userii}%
4698 }

\Glsentryuserii
4699 \newrobustcmd*{\Glsentryuserii}[1]{%
4700   \@Gls@entry@field{#1}{userii}%
4701 }

\glentryuseriii  Get the third user key (as specified by the user3 when the entry was defined). The argument
                is the label associated with the entry.
4702 \newcommand*{\glentryuseriii}[1]{%
4703   \@gls@entry@field{#1}{useriii}%
4704 }

\Glsentryuseriii
4705 \newrobustcmd*{\Glsentryuseriii}[1]{%
4706   \@Gls@entry@field{#1}{useriii}%
4707 }

\glentryuseriv  Get the fourth user key (as specified by the user4 when the entry was defined). The argument
                is the label associated with the entry.
4708 \newcommand*{\glentryuseriv}[1]{%
4709   \@gls@entry@field{#1}{useriv}%
4710 }

```

```

\Glsentryuseriv
4711 \newrobustcmd*{\Glsentryuseriv}[1]{%
4712   \@Gls@entry@field{#1}{useriv}%
4713 }

\glsentryuserv  Get the fifth user key (as specified by the user5 when the entry was defined). The argument is
                  the label associated with the entry.
4714 \newcommand*{\glsentryuserv}[1]{%
4715   \@Gls@entry@field{#1}{userv}%
4716 }

\Glsentryuserv
4717 \newrobustcmd*{\Glsentryuserv}[1]{%
4718   \@Gls@entry@field{#1}{userv}%
4719 }

\glsentryuservi  Get the sixth user key (as specified by the user6 when the entry was defined). The argument
                  is the label associated with the entry.
4720 \newcommand*{\glsentryuservi}[1]{%
4721   \@Gls@entry@field{#1}{uservi}%
4722 }

\Glsentryuservi
4723 \newrobustcmd*{\Glsentryuservi}[1]{%
4724   \@Gls@entry@field{#1}{uservi}%
4725 }

\glsentryshort  Get the short key (as specified by the short the entry was defined). The argument is the label
                  associated with the entry.
4726 \newcommand*{\glsentryshort}[1]{\@Gls@entry@field{#1}{short}}

\Glsentryshort
4727 \newrobustcmd*{\Glsentryshort}[1]{%
4728   \@Gls@entry@field{#1}{short}%
4729 }

glsentryshortpl  Get the short plural key (as specified by the shortplural the entry was defined). The argument
                  is the label associated with the entry.
4730 \newcommand*{\glsentryshortpl}[1]{\@Gls@entry@field{#1}{shortpl}}

Glsentryshortpl
4731 \newrobustcmd*{\Glsentryshortpl}[1]{%
4732   \@Gls@entry@field{#1}{shortpl}%
4733 }

\glsentrylong  Get the long key (as specified by the long the entry was defined). The argument is the label
                  associated with the entry.
4734 \newcommand*{\glsentrylong}[1]{\@Gls@entry@field{#1}{long}}

```



```

\Glsentrylong
4735 \newrobustcmd*{\Glsentrylong}[1]{%
4736   \@Gls@entry@field{#1}{long}%
4737 }

\glsentrylongpl  Get the long plural key (as specified by the longplural the entry was defined). The argument is
                  the label associated with the entry.
4738 \newcommand*{\glsentrylongpl}[1]{\@Gls@entry@field{#1}{longpl}}

\Glsentrylongpl
4739 \newrobustcmd*{\Glsentrylongpl}[1]{%
4740   \@Gls@entry@field{#1}{longpl}%
4741 }

Short cut macros to access full form:

\glsentryfull
4742 \newcommand*{\glsentryfull}[1]{%
4743   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4744 }

\Glsentryfull
4745 \newrobustcmd*{\Glsentryfull}[1]{%
4746   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4747 }

\glsentryfullpl
4748 \newcommand*{\glsentryfullpl}[1]{%
4749   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4750 }

\Glsentryfullpl
4751 \newrobustcmd*{\Glsentryfullpl}[1]{%
4752   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4753 }

entrynumberlist  Displays the number list as is.
4754 \newcommand*{\glsentrynumberlist}[1]{%
4755   \glsdoifexists{#1}%
4756   {%
4757     \@Gls@entry@field{#1}{numberlist}%
4758   }%
4759 }

splaynumberlist  Formats the number list for the given entry label. Doesn't work with hyperref.
4760 \@ifpackageloaded{hyperref} {%
4761   \newcommand*{\glsdisplaynumberlist}[1]{%
4762     \GlossariesWarning

```

```

4763   {%
4764       \string\glsdisplaynumberlist\space
4765       doesn't work with hyperref.^^JUsing
4766       \string\glsentrynumberlist\space instead%
4767   }%
4768   \glsentrynumberlist{#1}%
4769 }%
4770}%
4771{%
4772 \newcommand*{\glsdisplaynumberlist}[1]{%
4773   \glsdoifexists{#1}%
4774   {%
4775       \bgroup

4776       \edef\@glo@label{\glsdetoklabel{#1}}%
4777       \let\@org@glssnumberformat\glssnumberformat
4778       \def\glssnumberformat##1{##1}%
4779       \protected@edef\the@numberlist{%
4780         \csname glo@\@glo@label @numberlist\endcsname}%
4781       \def\@gls@numlist@sep{}%
4782       \def\@gls@numlist@nextsep{}%
4783       \def\@gls@numlist@lastsep{}%
4784       \def\@gls@thislist{}%
4785       \def\@gls@donext@def{}%
4786       \renewcommand\do[1]{%
4787         \protected@edef\@gls@thislist{%
4788           \@gls@thislist
4789           \noexpand\@gls@numlist@sep
4790           ##1%
4791         }%
4792         \let\@gls@numlist@sep\@gls@numlist@nextsep
4793         \def\@gls@numlist@nextsep{\glssnumlistsep}%
4794         \@gls@donext@def
4795         \def\@gls@donext@def{%
4796           \def\@gls@numlist@lastsep{\glssnumlistlastsep}%
4797         }%
4798       }%
4799       \expandafter \glssnumlistparser \expandafter{\the@numberlist}%
4800       \let\@gls@numlist@sep\@gls@numlist@lastsep
4801       \@gls@thislist
4802   \egroup
4803 }%
4804 }
4805 }

```

\glssnumlistsep

```
4806 \newcommand*{\glssnumlistsep}{, }
```

\glssnumlistlastsep

```
4807 \newcommand*{\glssnumlistlastsep}{ \& }
```

`\glshyperlink` Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\glslink` or `\glsadd` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```
4808 \newcommand*{\glshyperlink}[2][\glstrytext{\@glo@label}]{%
4809 \def\@glo@label{#2}%
4810 \@glslink{\glo@linkprefix\glsdetoklabel{#2}}{#1}}
```

1.12 Adding an entry to the glossary without generating text

The following keys are provided for `\glsadd` and `\glsaddall`:

```
4811 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4812 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}
```

This key is only used by `\glsaddall`:

```
4813 \define@key{glossadd}{types}{\def\@glo@type{#1}}
```

`\glsadd[<options>]{<label>}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *<options>* only has two keys: counter and format (the types key will be ignored).

`\glsadd`

```
4814 \newrobustcmd*{\glsadd}[2][]{%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```
4815 \@gls@adjustmode
4816 \glsdoifexists{#2}%
4817 {%
4818 \def\@glsnumberformat{glsnumberformat}%
4819 \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
4820 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
4821 \@gls@saveentrycounter
```

Define sort key if necessary:

```
4822 \@gls@setsort{#2}%
```

This should use `\@@do@wrglossary` rather than `\do@wrglossary` since the whole point of `\glsadd` is to add a line to the glossary.

```
4823 \@@do@wrglossary{#2}%
4824 }%
4825 }
```

@gls@adjustmode

```
4826 \newcommand*{\@gls@adjustmode}{}
4827 \AtBeginDocument{\renewcommand*{\@gls@adjustmode}{\ifvmode\mbox{}\fi}}
```

\glsaddall[*<option list>*]

Add all terms defined for the listed glossaries (without displaying any text). If types key is omitted, apply to all glossary types.

\glsaddall

```
4828 \newrobustcmd*{\glsaddall}[1] [] {%
4829   \edef\@glo@type{\@glo@types}%
4830   \setkeys{glossadd}{#1}%
4831   \forallglsentries[\@glo@type]{\@glo@entry}{%
4832     \glsadd[#1]{\@glo@entry}%
4833   }%
4834 }
```

glsaddallunused

\glsaddallunused[*<glossary type>*]

Add all used terms defined for the listed glossaries (without displaying any text). If optional argument is omitted, apply to all glossary types. This should typically go at the end of the document.

```
4835 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
4836   \forallglsentries[#1]{\@glo@entry}%
4837   {%
4838     \ifglsused{\@glo@entry}{\@glsadd[format=glsignore]{\@glo@entry}}%
4839   }%
4840 }
```

\glsignore

```
4841 \newcommand*{\glsignore}[1] {}
```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The makeindex actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about makeindex special characters).

The symbols and numbers label for group headings are hardwired into the .ist file as glssymbols and glsnumbers, the group titles can be translated (so that \glssymbolsgroupname replaces glssymbols and \glsnumbersgroupname replaces glsnumbers) using the command \glsetgrouptitle which is defined in . This is done to prevent any problem characters in \glssymbolsgroupname and \glsnumbersgroupname from breaking hyperlinks.

```
\glsopenbrace  Define \glsopenbrace to make it easier to write an opening brace to a file.
4842 \edef\glsopenbrace{\expandafter\@gobble\string\{}}

\glsclosebrace  Define \glsclosebrace to make it easier to write an opening brace to a file.
4843 \edef\glsclosebrace{\expandafter\@gobble\string\}}

\glsbackslash  Define \glsbackslash to make it easier to write a backslash to a file.
4844 \edef\glsbackslash{\expandafter\@gobble\string\\}

\glsquote  Define command that makes it easier to write quote marks to a file in the event that the double quote character has been made active.
4845 \edef\glsquote#1{\string"#1\string"}

\glpercentchar  Define \glpercentchar to make it easier to write a percent character to a file.
4846 \edef\glpercentchar{\expandafter\@gobble\string\%}

\glstildechar  Define \glstildechar to make it easier to write a tilde character to a file.
4847 \edef\glstildechar{\string~}

@glsfirstletter  Define the first letter to come after the digits 0,...,9. Only required for xindy.
4848 \ifglsxindy
4849   \newcommand*{\@glsfirstletter}{A}
4850 \fi

letterAfterDigits  Sets the first letter to come after the digits 0,...,9. The starred version sanitizes.
4851 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}{%
4852   \@ifstar\s@GlsSetXdyFirstLetterAfterDigits\@GlsSetXdyFirstLetterAfterDigits}
4853 \ifglsxindy
4854   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4855     \renewcommand*{\@glsfirstletter}{#1}}
4856   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}[1]{%
4857     \renewcommand*{\@glsfirstletter}{#1}%
4858     \@onelevel@sanitize\@glsfirstletter
4859   }
4860 \else
4861   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4862     \glsnoindywarning\GlsSetXdyFirstLetterAfterDigits}
4863   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}{%
4864     \@GlsSetXdyFirstLetterAfterDigits
4865   }
4866 \fi
```

numbergrouporder Specifies the order of the number group.

```
4867 \ifglxindy
4868   \newcommand*{\@xdynumbergrouporder}{:before \string"\@glxfirstletter\string"}
4869 \fi
```

numberGroupOrder Sets the relative location of the number group. The starred version sanitizes.

```
4870 \newcommand*{\GlsSetXdyNumberGroupOrder}[1]{%
4871   \@ifstar\s@GlsSetXdyNumberGroupOrder\@GlsSetXdyNumberGroupOrder
4872 }
4873 \ifglxindy
4874   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4875     \renewcommand*{\@xdynumbergrouporder}{#1}%
4876   }
4877   \newcommand*{\s@GlsSetXdyNumberGroupOrder}[1]{%
4878     \renewcommand*{\@xdynumbergrouporder}{#1}%
4879     \@onelevel@sanitize\@xdynumbergrouporder
4880   }
4881 \else
4882   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4883     \glxnindywarning\GlsSetXdyNumberGroupOrder}
4884   \newcommand*{\s@GlsSetXdyNumberGroupOrder}{%
4885     \@GlsSetXdyNumberGroupOrder}
4886 \fi
```

\@glxminrange Define the minimum number of successive location references to merge into a range.

```
4887 \newcommand*{\@glxminrange}{2}
```

yMinRangeLength Set the minimum range length. The value must either be none or a positive integer. The glossaries package doesn't check if the argument is valid, that is left to xindy.

```
4888 \ifglxindy
4889   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4890     \renewcommand*{\@glxminrange}{#1}}
4891 \else
4892   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4893     \glxnindywarning\GlsSetXdyMinRangeLength}
4894 \fi
```

\writeist

```
4895 \ifglxindy
    Code to use if xindy is required.
4896   \def\writeist{%
    Define write register if not already defined
4897     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4898     \@glx@addpredefinedattributes
```

Open the file.

```
4899 \openout\glswrite=\istfilename
```

Write header comment at the start of the file

```
4900 \write\glswrite{;; xindy style file created by the glossaries
4901 package}%
4902 \write\glswrite{;; for document '\jobname' on
4903 \the\year-\the\month-\the\day}%
```

Specify the required styles

```
4904 \write\glswrite{^^J; required styles^^J}
4905 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4906 \ifx\@xdystyle\@empty
4907 \else
4908 \protected@write\glswrite{}{(require
4909 \string"\@xdystyle.xdy\string")}%
4910 \fi
4911 }%
```

List the allowed attributes (possible values used by the format key)

```
4912 \write\glswrite{^^J%
4913 ; list of allowed attributes (number formats)^^J}%
4914 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4915 \write\glswrite{^^J; user defined alphabets^^J}%
4916 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4917 \write\glswrite{^^J; location class definitions^^J}%
```

As from version 3.0, locations are now specified as $\{\langle Hprefix \rangle\}\{\langle number \rangle\}$, so need to add all possible combinations of location types.

```
4918 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case where $\langle Hprefix \rangle$ is empty:

```
4919 \protected@write\glswrite{}{(define-location-class
4920 \string"\@gls@classI\string"^^J\space\space\space
4921 (
4922 :sep "{}{"
4923 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4924 :sep "}"
4925 )
4926 ^^J\space\space\space
4927 :min-range-length \@glsminrange^^J%
4928 )
4929 }%
```

Nested iteration over all classes:

```
4930 {%
4931 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4932 \protected@write\glswrite{}{(define-location-class
```

```

4933         \string"\@gls@classII-\@gls@classI\string"
4934         ^^J\space\space\space
4935     (
4936         :sep "{"
4937         \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4938         :sep "{{"
4939         \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4940         :sep "}"
4941     )
4942     ^^J\space\space\space
4943     :min-range-length \@glsminrange^^J%
4944 )
4945 }%
4946 }%
4947 }%
4948 }%

```

User defined location classes (needs checking for new location format).

```

4949 \write\glswrite{^^J; user defined location classes}%
4950 \write\glswrite{\@xdyuserlocationdefs}%

```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for `\glsseeformat` which xindy won't recognise.)

```

4951 \write\glswrite{^^J; define cross-reference class^^J}%
4952 \write\glswrite{(define-crossref-class \string"see\string"
4953     :unverified )}%

```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of `\glsseeformat` which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```

4954 \write\glswrite{(markup-crossref-list
4955     :class \string"see\string"^^J\space\space\space
4956     :open \string"\string\glsseeformat\string"
4957     :close \string"{}\string")}%

```

Provide hook to write extra material here (used by `glossaries-extra` to define a `seealso` class).

```

4958 \@xdycrossrefhook

```

List the order to sort the classes.

```

4959 \write\glswrite{^^J; define the order of the location classes}%
4960 \write\glswrite{(define-location-class-order
4961     (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4962 \write\glswrite{^^J; define the glossary markup^^J}%

4963 \write\glswrite{(markup-index^^J\space\space\space
4964     :open \string"\string
4965     \glossarysection[\string\glossarytoctitle]{\string
4966     \glossarytitle}\string\glossarypreamble}%

```


Add all the xindy-only macro definitions (needed to prevent errors in the event that the user changes from xindy to makeindex)

```

4967 \@for\@this@ctr:=\@xdycounters\do{%
4968   {%
4969     \@for\@this@attr:=\@xdyattributelist\do{%
4970       \protected@write\glswrite{}\string\providecommand*%
4971       \expandafter\string
4972       \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4973       {%
4974         \string\setentrycounter
4975         [\expandafter@gobble\string\#1]{\@this@ctr}%
4976         \expandafter\string
4977         \csname\@this@attr\endcsname
4978         {\expandafter@gobble\string\#2}%
4979       }%
4980     }%
4981   }%
4982 }%
4983 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4984 \write\glswrite{%
4985   \string\begin
4986   {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4987   \space\space:close \string"\glspercentchar\glstildechar n\string
4988   \end{theglossary}\string\glossarypostamble
4989   \glstildechar n\string" ^^J\space\space\space
4990   :tree)}}%

```

Specify what to put between letter groups

```

4991 \write\glswrite{(markup-letter-group-list
4992   :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4993 \write\glswrite{(markup-indexentry
4994   :open \string"\string\relax \string\glsresetentrylist
4995   \glstildechar n\string")}%

```

Specify how to format entries

```

4996 \write\glswrite{(markup-locclass-list :open
4997   \string"\glsopenbrace\string\glossaryentrynumbers
4998   \glsopenbrace\string\relax\space \string"^^J\space\space\space
4999   :sep \string", \string"
5000   :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

5001 \write\glswrite{(markup-locref-list
5002   :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

5003 \write\glswrite{(markup-range
5004   :sep \string"\string\delimR\space\string")}%

```

Specify 2-page and 3-page suffixes, if defined. First, the values must be sanitized to write them explicitly.

```

5005     \@onelevel@sanitize\gls@suffixF
5006     \@onelevel@sanitize\gls@suffixFF

5007     \ifx\gls@suffixF\@empty
5008     \else
5009         \write\glswrite{(markup-range
5010             :close "\gls@suffixF" :length 1 :ignore-end)}}%
5011     \fi
5012     \ifx\gls@suffixFF\@empty
5013     \else
5014         \write\glswrite{(markup-range
5015             :close "\gls@suffixFF" :length 2 :ignore-end)}}%
5016     \fi

```

Specify how to format locations.

```

5017     \write\glswrite{^^J; define format to use for locations^^J}%
5018     \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

5019     \write\glswrite{^^J; define letter group list format^^J}%
5020     \write\glswrite{(markup-letter-group-list
5021         :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Define letter group headings.

```

5022     \write\glswrite{^^J; letter group headings^^J}%
5023     \write\glswrite{(markup-letter-group
5024         :open-head \string"\string\glsgroupheading
5025         \glsopenbrace\string"^^J\space\space\space
5026         :close-head \string"\glsclosebrace\string")}%

```

Define additional letter groups.

```

5027     \write\glswrite{^^J; additional letter groups^^J}%
5028     \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

5029     \write\glswrite{^^J; additional sort rules^^J}
5030     \write\glswrite{\@xdysortrules}%

```

Hook for any additional information:

```

5031     \@gls@writeisthook

```

Close the style file

```

5032     \closeout\glswrite

```

Suppress any further calls.

```

5033     \let\writeist\relax
5034 }
5035 \else

```

Code to use if makeindex is required.

```

5036 \edef\@gls@actualchar{\string?}
5037 \edef\@gls@encapchar{\string|}
5038 \edef\@gls@levelchar{\string!}
5039 \edef\@gls@quotechar{\string"}%
5040 \let\GlsSetQuote\gls@nosetquote
5041 \def\writeist{\relax
5042 \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
5043 \openout\glswrite=\istfilename
5044 \write\glswrite{\glspersentchar\space makeindex style file
5045 created by the glossaries package}
5046 \write\glswrite{\glspersentchar\space for document
5047 '\jobname' on \the\year-\the\month-\the\day}
5048 \write\glswrite{actual '\@gls@actualchar'}
5049 \write\glswrite{encap '\@gls@encapchar'}
5050 \write\glswrite{level '\@gls@levelchar'}
5051 \write\glswrite{quote '\@gls@quotechar'}
5052 \write\glswrite{keyword \string\string\glossaryentry\string}
5053 \write\glswrite{preamble \string\string\glossarysection[\string
5054 \glossarytoctitle]{\string\glossarytitle}\string
5055 \glossarypreamble\string\n\string\begin{theglossary}\string
5056 \glossaryheader\string\n\string}
5057 \write\glswrite{postamble \string\string%\string\n\string
5058 \end{theglossary}\string\glossarypostamble\string\n
5059 \string}
5060 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
5061 \string}
5062 \write\glswrite{item_0 \string\string%\string\n\string}
5063 \write\glswrite{item_1 \string\string%\string\n\string}
5064 \write\glswrite{item_2 \string\string%\string\n\string}
5065 \write\glswrite{item_01 \string\string%\string\n\string}
5066 \write\glswrite{item_x1
5067 \string\string\relax \string\glsresetentrylist\string\n
5068 \string}
5069 \write\glswrite{item_12 \string\string%\string\n\string}
5070 \write\glswrite{item_x2
5071 \string\string\relax \string\glsresetentrylist\string\n
5072 \string}

5073 \write\glswrite{delim_0 \string\string\{\string
5074 \glossaryentrynumbers\string\{\string\relax \string}
5075 \write\glswrite{delim_1 \string\string\{\string
5076 \glossaryentrynumbers\string\{\string\relax \string}
5077 \write\glswrite{delim_2 \string\string\{\string
5078 \glossaryentrynumbers\string\{\string\relax \string}
5079 \write\glswrite{delim_t \string\string\}\string\}\string}
5080 \write\glswrite{delim_n \string\string\delimN \string}
5081 \write\glswrite{delim_r \string\string\delimR \string}
5082 \write\glswrite{headings_flag 1}
5083 \write\glswrite{heading_prefix

```

```

5084     \string\string\glsgroupheading\string\{\string"}
5085 \write\glswrite{heading_suffix
5086     \string\string\}\string\relax
5087     \string\glsgroupheading\string"}
5088 \write\glswrite{symhead_positive \string"glssymbols\string"}
5089 \write\glswrite{numhead_positive \string"glslnumbers\string"}
5090 \write\glswrite{page_compositor \string"glscpositor\string"}
5091 \@gls@escbsdq\gls@suffixF
5092 \@gls@escbsdq\gls@suffixFF
5093 \ifx\gls@suffixF\@empty
5094 \else
5095     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
5096 \fi
5097 \ifx\gls@suffixFF\@empty
5098 \else
5099     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
5100 \fi

```

Hook for any additional information:

```

5101 \@gls@writeisthook

```

Close the file and disable \writeist.

```

5102 \closeout\glswrite
5103 \let\writeist\relax
5104 }
5105 \fi

```

SetWriteIstHook Allow user to append information to the style file.

```

5106 \newcommand*\GlsSetWriteIstHook[1]{\renewcommand*\@gls@writeisthook{#1}}
5107 \@onlypremake\GlsSetWriteIstHook

```

ls@writeisthook

```

5108 \newcommand*\@gls@writeisthook{}

```

\GlsSetQuote Allow user to set the makeindex quote character. This is primarily for ngerman users who want to use makeindex's -g option.

```

5109 \ifglxindy
5110 \newcommand*\GlsSetQuote[1]{\glsnomakeindexwarning\GlsSetQuote}
5111 \newcommand*\gls@nosetquote[1]{\glsnomakeindexwarning\GlsSetQuote}
5112 \else
5113 \newcommand*\GlsSetQuote[1]{\edef\@gls@quotechar{\string#1}%

```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```

5114 \@ifpackageloaded{tracklang}%
5115 {%
5116     \IfTrackedLanguage{german}%
5117     {%
5118         \def\@gls@extramakeindexopts{-g}%
5119     }%
5120 }%

```

```

5121 }%
5122 {}%

Need to redefine \@gls@checkquote
5123 \edef\@gls@docheckquotedef{%
5124   \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
5125     \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5126     \noexpand\toks@={####1}%
5127     \noexpand\ifx\noexpand\null####2\noexpand\null
5128     \noexpand\ifx\noexpand\null####3\noexpand\null
5129     \noexpand\edef\noexpand\@gls@checkedmkidx{%
5130       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5131     \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
5132     \noexpand\else
5133     \noexpand\edef\noexpand\@gls@checkedmkidx{%
5134       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5135       \noexpand\@gls@quotechar\noexpand\@gls@quotechar
5136       \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
5137     \noexpand\def\noexpand\@gls@checkquote{%
5138       \noexpand\@gls@checkquote####3\noexpand\null}%
5139     \noexpand\fi
5140     \noexpand\else
5141     \noexpand\edef\noexpand\@gls@checkedmkidx{%
5142       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5143       \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
5144     \noexpand\ifx\noexpand\null####3\noexpand\null
5145     \noexpand\def\noexpand\@gls@checkquote{%
5146       \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
5147     \noexpand\else
5148     \noexpand\def\noexpand\@gls@checkquote{%
5149       \noexpand\@gls@checkquote####2#1#1####3\noexpand\null}%
5150     \noexpand\fi
5151     \noexpand\fi
5152     \noexpand\@gls@checkquote
5153   }%
5154 }%
5155 \@gls@docheckquotedef
5156 \edef\@gls@docheckquotedef{%
5157   \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
5158     \noexpand\def\noexpand\@gls@checkedmkidx{%
5159       \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
5160       #1#1\noexpand\null
5161       \noexpand\expandafter\noexpand\@gls@updatechecked
5162       \noexpand\@gls@checkedmkidx{####1}%
5163       \noexpand\def\noexpand\@gls@checkedmkidx{%
5164         \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
5165         \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
5166         \noexpand\null
5167         \noexpand\expandafter\noexpand\@gls@updatechecked
5168         \noexpand\@gls@checkedmkidx{####1}%

```

```

5169 \noexpand\def\noexpand\@gls@checkedmkidx{%
5170 \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
5171 \noexpand\?\noexpand\?\noexpand\null
5172 \noexpand\expandafter\noexpand\@gls@updatechecked
5173 \noexpand\@gls@checkedmkidx{####1}%
5174 \noexpand\def\noexpand\@gls@checkedmkidx{%
5175 \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
5176 \noexpand?\noexpand?\noexpand\null
5177 \noexpand\expandafter\noexpand\@gls@updatechecked
5178 \noexpand\@gls@checkedmkidx{####1}%
5179 \noexpand\def\noexpand\@gls@checkedmkidx{%
5180 \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
5181 \noexpand|\noexpand|\noexpand\null
5182 \noexpand\expandafter\noexpand\@gls@updatechecked
5183 \noexpand\@gls@checkedmkidx{####1}%
5184 \noexpand\def\noexpand\@gls@checkedmkidx{%
5185 \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
5186 \noexpand||\noexpand||\noexpand\null
5187 \noexpand\expandafter\noexpand\@gls@updatechecked
5188 \noexpand\@gls@checkedmkidx{####1}%
5189 \noexpand\def\noexpand\@gls@checkedmkidx{%
5190 \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
5191 \noexpand!\noexpand!\noexpand\null
5192 \noexpand\expandafter\noexpand\@gls@updatechecked
5193 \noexpand\@gls@checkedmkidx{####1}%
5194 }%
5195 }%
5196 \@gls@docheckquotedef
5197 \edef\@gls@docheckquotedef{%
5198 \noexpand\def\noexpand\@gls@checkescquote####1%
5199 \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
5200 ####3\noexpand\null{%
5201 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5202 \noexpand\toks@={####1}%
5203 \noexpand\ifx\noexpand\null####2\noexpand\null
5204 \noexpand\ifx\noexpand\null####3\noexpand\null
5205 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5206 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5207 \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
5208 \noexpand\else
5209 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5210 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5211 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5212 \csname#1\endcsname}\noexpand\@gls@quotechar
5213 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5214 \csname#1\endcsname}\noexpand\@gls@quotechar}%
5215 \noexpand\def\noexpand\@gls@checkescquote{%
5216 \noexpand\@gls@checkescquote####3\noexpand\null}%
5217 \noexpand\fi

```

```

5218 \noexpand\else
5219 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5220 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5221 \noexpand\@gls@quotechar\noexpand\string
5222 \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
5223 \noexpand\ifx\noexpand\null####3\noexpand\null
5224 \noexpand\def\noexpand\@gls@checkescquote{%
5225 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5226 \expandonce{\csname#1\endcsname}\noexpand\null}%
5227 \noexpand\else
5228 \noexpand\def\noexpand\@gls@checkescquote{%
5229 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5230 ####3\noexpand\null}%
5231 \noexpand\fi
5232 \noexpand\fi
5233 \noexpand\@gls@checkescquote
5234 }%
5235 }%
5236 \@gls@docheckquotedef
5237 }
5238 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
5239 {\string\GlsSetQuote\space not permitted here}%
5240 {Move \string\GlsSetQuote\space earlier in the preamble, as
5241 soon as possible after glossaries.sty has been loaded}}
5242 \fi

```

ramakeindexopts

```

5243 \newcommand*{\@gls@extramakeindexopts}[1]{}

```

The command `\noist` will suppress the creation of the `.ist` file. Obviously you need to use this command before `\writeist` to have any effect.

`\noist`

```

5244 \newcommand{\noist}{%
  Update attributes list
5245 \@gls@addpredefinedattributes
5246 \let\writeist\relax
5247 }

```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the `<out-ext>` parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `.ist` `makeindex` style file).

Note that you can't use `\@makeglossary` for only some of the defined glossaries (with just the base `glossaries` package). You either need to have a `\makeglossaries` for all glossaries or none (otherwise you will end up with a situation where \TeX is trying to write to a non-existent

file). The relevant glossary must be defined prior to using \@makeglossary. glossaries-extra allows for a hybrid approach.

\@makeglossary

```
5248 \newcommand*{\@makeglossary}[1]{%
5249   \ifglossaryexists{#1}%
5250   {%
```

Only create a new write if savewrites=false otherwise create a token to collect the information.

```
5251   \ifglssavewrites
5252     \expandafter\newtoks\csname glo@#1@filetok\endcsname
5253   \else
5254     \expandafter\newwrite\csname glo@#1@file\endcsname
5255     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
5256   \fi
5257   \@gls@renewglossary
5258   \writeist
5259 }%
5260 {%
5261   \PackageError{glossaries}%
5262   {Glossary type ‘#1’ not defined}%
5263   {New glossaries must be defined before using \string\makeglossaries}%
5264 }%
5265 }
```

\@glsopenfile Open write file associated with the given glossary.

```
5266 \newcommand*{\@glsopenfile}[2]{%
5267   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
5268   \PackageInfo{glossaries}{Writing glossary file
5269     \jobname.\csname @glotype@#2@out\endcsname}%
5270 }
```

\@closegls

```
5271 \newcommand*{\@closegls}[1]{%
5272   \closeout\csname glo@#1@file\endcsname
5273 }
```

\@gls@automake

```
5274 \ifglsxindy
5275 \newcommand*{\@gls@automake}[1]{%
5276   \ifglossaryexists{#1}
5277   {%
5278     \@closegls{#1}%
5279     \ifdefstring{\glsorder}{letter}%
5280     {\def\@gls@order{-M ord/letorder }}%
5281     {\let\@gls@order\@empty}%
5282     \ifcsundef{@xdy@#1@language}%
5283     {\let\@gls@langmod\@xdy@main@language}%
5284   }
```



```

5284     {\letcs\@gls@langmod{\xdy@#1@language}}%
5285 \edef\@gls@dothiswrite{\noexpand\write18{xindy
5286   -I xindy
5287   \@gls@order
5288   -L \@gls@langmod\space
5289   -M \@gls@istfilebase\space
5290   -C \@gls@codepage\space
5291   -t \jobname.\csuse{@glotype@#1@log}
5292   -o \jobname.\csuse{@glotype@#1@in}
5293   \jobname.\csuse{@glotype@#1@out}}}%
5294 }%
5295 \@gls@dothiswrite
5296 }%
5297 {%
5298   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5299 }%
5300 }
5301 \else
5302 \newcommand*{\@gls@automake}[1]{%
5303   \ifglossaryexists{#1}
5304   {%
5305     \@closegls{#1}%
5306     \ifdefstring{\glsorder}{letter}%
5307     {\def\@gls@order{-l }}%
5308     {\let\@gls@order\@empty}%
5309     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5310       -s \istfilename\space
5311       -t \jobname.\csuse{@glotype@#1@log}
5312       -o \jobname.\csuse{@glotype@#1@in}
5313       \jobname.\csuse{@glotype@#1@out}}}%
5314     }%
5315     \@gls@dothiswrite
5316   }%
5317   {%
5318     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5319   }%
5320 }
5321 \fi

```

omake@immediate

```

5322 \ifglsxindy
5323 \newcommand*{\@gls@automake@immediate}[1]{%
5324   \ifglossaryexists{#1}
5325   {%
5326     \IfFileExists{\jobname.\csuse{@glotype@#1@out}}%
5327     {%
5328       \ifdefstring{\glsorder}{letter}%
5329       {\def\@gls@order{-M ord/letorder }}%
5330       {\let\@gls@order\@empty}%

```

```

5331 \ifcsundef{@xdy@#1@language}%
5332 {\let\@gls@langmod\@xdy@main@language}%
5333 {\letcs\@gls@langmod{@xdy@#1@language}}%
5334 \edef\@gls@dothiswrite{\noexpand\immediate\noexpand\write18{xindy
5335 -I xindy
5336 \@gls@order
5337 -L \@gls@langmod\space
5338 -M \@gls@istfilebase\space
5339 -C \@gls@codepage\space
5340 -t \jobname.\csuse{@glotype@#1@log}
5341 -o \jobname.\csuse{@glotype@#1@in}
5342 \jobname.\csuse{@glotype@#1@out}}}%
5343 }%
5344 \@gls@dothiswrite
5345 }%
5346 {\GlossariesWarning{can't automake '#1': \jobname.\csuse{@glotype@#1@out}
5347 doesn't exist. Rerun may be required}}%
5348 }%
5349 {%
5350 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5351 }%
5352 }
5353 \else
5354 \newcommand*{\@gls@automake@immediate}[1]{%
5355 \ifglossaryexists{#1}
5356 {%
5357 \IfFileExists{\jobname.\csuse{@glotype@#1@out}}%
5358 {%
5359 \ifdefstring{\glsorder}{letter}%
5360 {\def\@gls@order{-l }}%
5361 {\let\@gls@order\@empty}%
5362 \edef\@gls@dothiswrite{\noexpand\immediate\noexpand\write18{makeindex \@gls@order
5363 -s \istfilename\space
5364 -t \jobname.\csuse{@glotype@#1@log}
5365 -o \jobname.\csuse{@glotype@#1@in}
5366 \jobname.\csuse{@glotype@#1@out}}}%
5367 }%
5368 \@gls@dothiswrite
5369 }%
5370 {\GlossariesWarning{can't automake '#1': \jobname.\csuse{@glotype@#1@out}
5371 doesn't exist. Rerun may be required}}%
5372 }%
5373 {%
5374 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5375 }%
5376 }
5377 \fi

```

omakeglossaries Issue warning that \makeglossaries hasn't been used.

```
5378 \newcommand*{\@warn@nomakeglossaries}{}
```

Only use this if warning if \printglossary has been used without \makeglossaries

```
5379 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}
```

omake@immediate

```
5380 \newcommand{\@gls@@automake@immediate}{%
5381 \ifnum\gls@automake@nr=2\relax
5382 \@for\@gls@type:=\@glo@types\do{%
5383 \ifdefempty{\@gls@type}{}%
5384 {\@gls@automake@immediate{\@gls@type}}%
5385 }%
5386 \glsautomakefalse
5387 \renewcommand*{\@gls@doautomake}{}%
5388 \fi
5389 }
```

\makeglossaries will use \@makeglossary for each glossary type that has been defined. New glossaries need to be defined before using \makeglossary, so have \makeglossaries redefine \newglossary to prevent it being used afterwards.

\makeglossaries

```
5390 \newcommand*{\makeglossaries}{%
5391 \@domakeglossaries
5392 {%
```

If automake=immediate setting is on, use the shell escape now.

```
5393 \@gls@@automake@immediate
```

Define the write used for style file also used for all other output files if savewrites=true.

```
5394 \ifundef{\glswrite}{\newwrite\glswrite}{}%
```

If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```
5395 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}%
5396 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}%}
```

If \@gls@extramakeindexopts has been defined, write it:

```
5397 \ifundef\@gls@extramakeindexopts
5398 {}%
5399 {%
5400 \protected@write\@auxout{}{\string\providecommand
5401 \string\@gls@extramakeindexopts[1]{}%
5402 \protected@write\@auxout{}{\string\@gls@extramakeindexopts
5403 {\@gls@extramakeindexopts}}%
5404 }%
```

Write the name of the style file to the aux file (needed by makeglossaries)

```
5405 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
5406 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}%
```

Iterate through each glossary type and activate it.

```
5407 \for\@glo@type:=\@glo@types\do{%
5408   \ifthenelse{\equal{\@glo@type}{}}{}{}{%
5409     \@makeglossary{\@glo@type}}%
5410 }%
```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```
5411 \renewcommand*\newglossary[4][]{%
5412   \PackageError{glossaries}{New glossaries
5413     must be created before \string\makeglossaries}{You need
5414     to move \string\makeglossaries\space after all your
5415     \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect. The deprecated `\makeglossary` is not redefined here as it either implements `\makeglossaries` or has been restored to its original definition (in which case it shouldn't be changed).

```
5416 \let\@makeglossary\@gobble
5417 \let\makeglossaries\relax
```

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

```
5418 \@disable@onlypremakeg
```

Allow see key:

```
5419 \let\gls@checkseeallowed\relax
```

Suppress warning about no `\makeglossaries`

```
5420 \let\warn@nomakeglossaries\relax
```

Activate warning about missing `\printglossary`

```
5421 \def\warn@noprntglossary{%
5422   \ifdefstring{\@glo@types}{,}%
5423   {%
5424     \GlossariesWarningNoLine{No glossaries have been defined}%
5425   }%
5426   {%
5427     \GlossariesWarningNoLine{No \string\printglossary\space
5428       or \string\printglossaries\space
5429       found. ^^J(Remove \string\makeglossaries\space if you
5430       don't want any glossaries.) ^^JThis document will not
5431       have a glossary}%
5432   }%
5433 }%
```

Declare list parser for `\glsdisplaynumberlist`

```
5434 \ifglssavenumberlist
5435   \edef\@gls@dodeflistparser{\noexpand\DeclareListParser
5436     {\noexpand\glsnumlistparser}{\delimN}}%
5437   \@gls@dodeflistparser
5438 \fi
```

Prevent user from also using `\makenoidxglossaries`

```
5439 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
5440 \renewcommand*{\@printgloss@setsort}{%
5441 \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5442 }%
```

Check the automake setting:

```
5443 \ifglsautomake
5444 \renewcommand*{\@gls@doautomake}{%
5445 \@for\@gls@type:=\@glo@types\do{%
5446 \ifdefempty{\@gls@type}{}%
5447 {\@gls@automake{\@gls@type}}%
5448 }%
5449 }%
5450 \fi
```

Check the sort setting:

```
5451 \@glo@check@sortallowed\makeglossaries
5452 }%
5453 }
```

Must occur in the preamble:

```
5454 \@onlypreamble{\makeglossaries}
```

`\glswrite` The definition of `\glswrite` has now been moved to `\makeglossaries` so that it's only defined if needed.

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
5455 \AtEndDocument{%
5456 \warn@nomakeglossaries
5457 \warn@noprintglossary
5458 }
```

`noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
5459 \newcommand*{\makenoidxglossaries}{%
5460 \@domakeglossaries
5461 }
```

Redefine empty glossary warning:

```
5462 \renewcommand{\@gls@noref@warn}[1]{%
5463 \GlossariesWarning{Empty glossary for
5464 \string\printnoidxglossary[type={##1}].
5465 Rerun may be required (or you may have forgotten to use
5466 commands like \string\gls)}}%
5467 }
```

Don't escape makeindex/xindy characters:

```
5468 \let\@gls@checkmkidxchars\@gobble
```

Don't escape locations:

```
5469 \glscloclocationsfalse
```

Write glossary information to aux instead of glossary files

```
5470 \let\do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5471 \let\gls@getgrouptitle\gls@noidx@getgrouptitle
```

Allow see key:

```
5472 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5473 \renewcommand{\do@seeglossary}[2]{%
5474   \edef\gls@label{\glsdetoklabel{##1}}%
5475   \protected@write\@auxout{}{%
5476     \string\gls@reference
5477     {\csname glo@\gls@label @type\endcsname}%
5478     {\gls@label}%
5479     {%
5480       \string\glsseeformat##2}%
5481     }%
5482   }%
5483 }%
```

If user removes the glossaries package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```
5484 \AtBeginDocument
5485 {%
5486   \write\@auxout{\string\providecommand\string\gls@reference[3]{}}%
5487 }%
```

Change warning about no glossaries

```
5488 \def\warn@noprintglossary{%
5489   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5490     or \string\printnoidxglossaries ^^J
5491     found. (Remove \string\makenoidxglossaries\space if you
5492     don't want any glossaries.)^^JThis document will not have a glossary}%
5493 }%
```

Suppress warning about no \makeglossaries

```
5494 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5495 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5496 \renewcommand*{\@printgloss@setsort}{%
5497   \let\glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
5498   \def\glo@sorttype{\glo@default@sorttype}%
5499 }%
```

All entries must be defined in the preamble:

```

5500 \renewcommand*{\new@glossaryentry[2]}{%
5501   \PackageError{glossaries}{Glossary entries must be
5502     defined in the preamble^^Jwhen you use
5503     \string\makenoidxglossaries}%
5504   {Either move your definitions to the preamble or use
5505     \string\makeglossaries}%
5506 }%

Redefine \glsentrynumberlist
5507 \renewcommand*{\glsentrynumberlist}[1]{%
5508   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5509   \ifdef\@gls@loclist
5510   {%
5511     \glsnoidxloclist{\@gls@loclist}%
5512   }%
5513   {%
5514     ??\glsdoifexists{##1}%
5515     {%
5516       \GlossariesWarning{Missing location list for ‘##1’. Either
5517         a rerun is required or you haven’t referenced the entry}%
5518     }%
5519   }%
5520 }%

Redefine \glsdisplaynumberlist
5521 \renewcommand*{\glsdisplaynumberlist}[1]{%
5522   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5523   \ifdef\@gls@loclist
5524   {%
5525     \def\@gls@noidxloclist@sep{%
5526       \def\@gls@noidxloclist@sep{%
5527         \def\@gls@noidxloclist@sep{%
5528           \glsnumlistsep
5529         }%
5530       \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5531     }%
5532   }%
5533   \def\@gls@noidxloclist@finalsep{}%
5534   \def\@gls@noidxloclist@prev{}%
5535   \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5536   \@gls@noidxloclist@finalsep
5537   \@gls@noidxloclist@prev
5538 }%
5539 {%
5540   ??\glsdoifexists{##1}%
5541   {%
5542     \GlossariesWarning{Missing location list for ‘##1’. Either
5543       a rerun is required or you haven’t referenced the entry}%
5544   }%

```

```

5545 }%
5546 }%

```

Provide a generic way of iterating through the number list:

```

5547 \renewcommand*{\glsnumberlistloop}[3]{%
5548   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5549   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5550   \let\@gls@org@glsseeformat\glsseeformat
5551   \let\glsnoidxdisplayloc##2\relax
5552   \let\glsseeformat##3\relax
5553   \ifdef\@gls@loclist
5554   {%
5555     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5556   }%
5557   {%
5558     ??\glsdoifexists{##1}%
5559     {%
5560       \GlossariesWarning{Missing location list for ‘##1’. Either
5561         a rerun is required or you haven’t referenced the entry}%
5562     }%
5563   }%
5564   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5565   \let\glsseeformat\@gls@org@glsseeformat
5566 }%

```

Modify sanitize sort function

```

5567 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5568 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5569 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5570 \@glo@check@sortallowed\makenoidxglossaries
5571 }%
5572 }

```

Preamble-only command:

```

5573 \@onlypreamble{\makenoidxglossaries}

```

numberlistloop `\glsnumberlistloop{<label>}{<handler>}`

```

5574 \newcommand*{\glsnumberlistloop}[2]{%
5575   \PackageError{glossaries}{\string\glsnumberlistloop\space
5576     only works with \string\makenoidxglossaries}{}%
5577 }

```

listloophandler Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc {<prefix>}{<counter>}{<format>}{<n>}`)

```

5578 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5579   #1%
5580 }

```



```

@makeglossaries Can't use both \makeglossaries and \makenoidxglossaries
5581 \newcommand*{\@no@makeglossaries}{%
5582   \PackageError{glossaries}{You can't use both
5583   \string\makeglossaries\space and \string\makenoidxglossaries}%
5584   {Either use one or other (or none) of those commands but not both
5585   together.}%
5586 }

@gl@s@noref@warn Warning when no instances of \@gl@s@reference found.
5587 \newcommand{\@gl@s@noref@warn}[1]{%
5588   \GlossariesWarning{\string\makenoidxglossaries\space
5589   is required to make \string\printnoidxglossary[type={#1}] work}%
5590 }

```

1.14 Writing information to associated files

`s@noidxglossary` Write the glossary information to the aux file (for the ‘noidx’ method):

```

5591 \newcommand*{\@gl@s@noidxglossary}{%
5592   \protected@write\@auxout{}{%
5593     \string\@gl@s@reference
5594     {\csname glo@\@gl@s@label @type\endcsname}%
5595     {\@gl@s@label}%
5596     {\string\gl@s@noidxdisplayloc
5597     {\@glo@counterprefix}%
5598     {\@gl@s@counter}%
5599     {\@gl@s@numberformat}%
5600     {\@gl@s@locref}%
5601     }%
5602   }%
5603 }

```

`\istfile` Deprecated.

```

5604 \providecommand\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

5605 \AtEndDocument{%
5606   \glswritefiles
5607 }

```

`\@glswritefiles` Only write the files if `savewrites=true`.

```

5608 \newcommand*{\@glswritefiles}{%
  Iterate through all the glossaries.
5609   \forallofglossaries{\@glo@type}{%
    Check for empty glossaries (patch provided by Patrick Häcker)
5610     \ifcsundef{glo@\@glo@type @filetok}%
5611     {%

```

```

5612     \def\gls@tmp{}%
5613 }%
5614 {%
5615     \edef\gls@tmp{\expandafter\the
5616         \csname glo@\@glo@type @filetok\endcsname}%
5617 }%
5618 \ifx\gls@tmp\@empty
5619     \ifx\@glo@type\glsdefaulttype
5620         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5621             entries.^^JRemember to use package option ‘nomain’ if
5622 you
5623             don’t want to^^Juse the main glossary}%
5624     \else
5625         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5626             entries}%
5627     \fi
5628 \else
5629     \@glsopenfile{\glswrite}{\@glo@type}%
5630     \immediate\write\glswrite{%
5631         \expandafter\the
5632         \csname glo@\@glo@type @filetok\endcsname}%
5633     \immediate\closeout\glswrite
5634 \fi
5635 }%
5636 }

```

As from v4.10, the `\glossary` command isn’t used by the `glossaries` package. Since the user isn’t expected to use this command (as `glossaries` takes care of the particular format required for `makeindex/xindy`) there’s no need for a user level command. Using a custom internal command prevents any conflict with other packages (and with the `\mark` mechanism).

The associated number should be stored in `\theglentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```

5637 \newcommand*{\gls@glossary}[1]{%
5638     \@gls@glossary{#1}%
5639 }

```

`\@gls@glossary`

```
\@gls@glossary{<type>}{<indexing info>}
```

(In v4.10, `\@glossary` was redefined to `\@gls@glossary` to avoid conflict with other packages.) Initially define internal `\@gls@glossary` to ignore its argument. Indexing will be enabled when `\@gls@glossary` is redefined by `\@makeglossary`.

This command was originally defined to do `\@index{<indexing info>}` so that it behaved much like `\index`. The definition was then changed to use `\index` as `memoir` changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.)

However, if normal indexing is enabled (for example with `\makeindex`) but no glossary lists are required (so `\@makeglossary` isn’t used), then `\index` will cause a problem here.

The `\@index` trick allows for special characters within *<indexing info>* (so you can do, for example, `\index{%@\%}`), and the original design of `\@glossary` here was actually a legacy from the old glossary package. With the glossaries package, the indexing information supplied in the second argument is more constrained and just consists of the sort value (given by the sort key), the actual value (given by `\glossentry{<label>}` or `\subglossentry{<level>}{<label>}`), and the format. This means that there's no need to worry about special characters appearing in the second argument as they can't be in the label or sort value. (If they are in the sort value then the category code would've needed to be changed when the entry was defined or `\glpercentchar` would be needed with the sort sanitization switched off.) This means that it's safe to simply ignore the second argument.

```
5640 \newcommand*{\@gls@glossary}[2]{%
5641   \if@gls@debug
5642     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5643   \fi
5644 }
```

This is a convenience command to set `\@gls@glossary`. It's used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`\@renewglossary`

```
5645 \newcommand{\@gls@renewglossary}{%
5646   \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
5647   \let\@gls@renewglossary\empty
5648 }
```

The `\gls@wrglossary` command is defined to have two arguments. The first argument is the glossary type, the second argument is the glossary entry (the format of which is set in `\glslink`).

`\gls@wrglossary`

```
5649 \newcommand*{\gls@wrglossary}[2]{%
5650   \ifglssavewrites
5651     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5652     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5653       \expandafter{\@gls@tmp^^J}%
5654   \else
5655     \ifcsdef{glo@#1@file}%
5656     {%
5657       \expandafter\protected@write\csname glo@#1@file\endcsname{%
5658         \gls@disablepagerefexpansion}{#2}%
5659     }%
5660     {%
5661       \ifignoredglossary{#1}{}%
5662       {%
5663         \GlossariesWarning{No file defined for glossary '#1'}%
5664       }%
5665     }%
```

```

5666 \fi
5667 \endgroup\@esphack
5668 }

```

\do@wrglossary

```

5669 \newcommand*\do@wrglossary}[1]{%
5670   \glswriteentry{#1}{\do@wrglossary{#1}}%
5671 }

```

\glswriteentry Provide a user level command so the user can customize whether or not a line should be added to the glossary. The arguments are the label and the code that writes to the glossary file.

```

5672 \newcommand*\glswriteentry}[2]{%
5673   \ifglsexindexonlyfirst
5674     \ifglsexused{#1}{#2}%
5675   \else
5676     #2%
5677   \fi
5678 }

```

protected@pagefmts List of page formats to be protected against expansion.

```

5679 \newcommand{\glsexprotected@pagefmts}{\glsexnumberpage,\glsexalphpage,%
5680   \glsexAlphpage,\glsexromanpage,\glsexRomanpage,\glsexarabicpage}

```

pagerefexpansion

```

5681 \newcommand*\glsexdisablepagerefexpansion{%
5682   \@for\glsexthis:=\glsexprotected@pagefmts\do
5683   {%
5684     \expandafter\let\glsexthis\relax
5685   }%
5686 }

```

\glsexalphpage

```

5687 \newcommand*\glsexalphpage{\@alph\c@page}

```

\glsexAlphpage

```

5688 \newcommand*\glsexAlphpage{\@Alph\c@page}

```

\glsexnumberpage

```

5689 \newcommand*\glsexnumberpage{\number\c@page}

```

\glsexarabicpage

```

5690 \newcommand*\glsexarabicpage{\@arabic\c@page}

```

\glsexromanpage

```

5691 \newcommand*\glsexromanpage{\romannumeral\c@page}

```

`\gls@Romanpage`

```
5692 \newcommand*{\gls@Romanpage}{\@Roman\c@page}
```

`protectedpagefmt`

```
\glsaddprotectedpagefmt{<cs name>}
```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument (`\<csname>\c@page` must be valid).

```
5693 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5694   \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}%
5695   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5696   \eappto\@wrglossarynumberhook{%
5697     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5698     \expandonce{\csname#1\endcsname}%
5699     \noexpand\def\expandonce{\csname#1\endcsname}{%
5700       \noexpand\@wrglossary@pageformat
5701       \expandonce{\csname gls#1page\endcsname}%
5702       \expandonce{\csname org@gls#1\endcsname}%
5703     }%
5704   }%
5705 }
```

`ssarynumberhook` Hook used by `\@@do@wrglossary`

```
5706 \newcommand*\@wrglossarynumberhook{}
```

`sary@pageformat`

```
5707 \newcommand{\@wrglossary@pageformat}[3]{%
5708   \ifx#3\c@page #1\else #2#3\fi
5709 }
```

`@@do@wrglossary` Write the glossary entry in the appropriate format.

```
5710 \newcommand*{\@@do@wrglossary}[1]{%
5711   \ifglsecllocations
5712     \@@do@esc@wrglossary{#1}%
5713   \else
5714     \@@do@noesc@wrglossary{#1}%
5715   \fi
5716 }
```

`oesc@wrglossary` Write the glossary entry in the appropriate format. The locations don't need to be pre-processed before writing the information to the glossary file, but the prefix still needs to be found.

```
5717 \newcommand*{\@@do@noesc@wrglossary}[1]{%
```

Don't fully expand yet.

```
5718 \expandafter\def\expandafter\@glslocref\expandafter{\theglentrycounter}%
5719 \expandafter\def\expandafter\@glsHlocref\expandafter{\theHglentrycounter}%
```

Find the prefix if `\@glsHlocref` and `\@glslocref` aren't the same.

```
5720 \ifx\@glsHlocref\@glslocref
5721 \def\@glo@counterprefix{}%
5722 \else
```

The value of the counter isn't important here as it's the prefix that's of interest. (`\c@page` will have the same value in both `\theHglentrycounter` and `\theHglentrycounter` at this point, even if it hasn't been updated yet. The page number is not expected to occur in the prefix.)

```
5723 \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5724 {\@glslocref}{\@glsHlocref}}%
5725 }%
5726 \@do@gls@getcounterprefix
5727 \fi
```

De-tok label if required.

```
5728 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5729 \@do@wrglossary
5730 }
```

`\owprimitivemods` Conditional to determine whether or not `\@do@esc@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
5731 \newif\ifglswrallowprimitivemods
5732 \glswrallowprimitivemodstrue
```

`\@esc@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\@glsnumberformat` and `\@gls@counter` prior to use.) The argument is the entry's label. This is far more complicated with `xindy` than with other indexing methods. There are two necessary but conflicting requirements with `xindy`:

1. all backslashes in the location must be escaped;
2. `\c@page` can't be prematurely expanded.

(With `makeindex` there's the remote possibility that the page compositor is a `makeindex` special character, so that would also need to be escaped.)

For example, suppose `\thepage` is defined as

```
\renewcommand{\thepage}{\tally{page}}
\newcommand{\tally}[1]{\tallynum{\expandafter\the\csname c@#1\endcsname}}
```

where `\tallynum` is a robust command that takes a number as its argument. With all indexing methods other than `xindy`, a deferred write with `\thepage` as the location will expand to `\tallynum{<n>}` where `<n>` is the page number. Since the write is deferred, the page number is correct. (`makeindex` won't accept this location format, but `\makenoidxglossaries` and `bib2gls` are quite happy with it.) Unfortunately, this fails with `xindy` because `xindy` interprets this location as `\tallynum{<n>}` because `\t` represents a the character "t". The location must be written as `\\tallynum{<n>}`.

This means that the location `\tally{page}` must be expanded and then the backslashes must be doubled. Unfortunately `\c@page` mustn't be expanded until the deferred write is performed, so the location actually needs to be expanded to `\tallynum{\the\c@page}` but the backslashes in `\the\c@page` mustn't be escaped. All other backslashes must be escaped. (In this case, only the backslash in `\tallynum` but the location format may include other control sequences.) The code below works on the assumption that commands like `\tally` are defined in the form

```
\newcommand{\tally}[1]{\tallynum{\expandafter\the\csname c@#1\endcsname}}
```

(note the use of `\expandafter` and `\name`) or in the form

```
\newcommand{\tally}[1]{\tallynum{\arabic{#1}}}
```

In the second case, `\arabic` is one of the known commands that's temporarily adjusted to prevent `\c@page` from being prematurely expanded. In the first case, `\the` is temporarily modified (unless `\glswrallowprimitivemodsfalse`) to check if it's followed by `\c@page`. The `\expandafter` ensures that it is. If `\tally` is defined in another way that hides `\c@page` for example using `\the\value{#1}` then the process fails.

With `makeindex`, `\tallynum` needs to expand to just the decimal number while writing the location to the glossary file, otherwise `makeindex` will reject it. This can be done by defining `\glstallypage` so that `\tally` can locally be set to `\arabic` while expansion is occurring. Again, `\c@page` must be protected from expansion until the deferred write occurs.

The expansion before the write occurs also allows the hyper prefix to be determined where `\theH<counter>` is defined in the form `<prefix>.\the<counter>`. It's possible (although again unlikely) that a `makeindex` character might occur in the prefix, which therefore needs escaping. The prefix is passed as the optional argument of `\setentrycounter` which is needed by commands like `\glshypernumber` to create a hyperlink for a given counter (like `\hyperpage` but for an arbitrary counter).

```
5733 \newcommand*{\@@do@esc@wrglossary}[1]{% please read documented code!
5734 \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions (scoped):

```
5735 \let\gls@orgthe\the
5736 \let\gls@orgnumber\number
5737 \let\gls@orgarabic\@arabic
5738 \let\gls@orgromannumeral\romannumeral
5739 \let\gls@orgalph\@alph
5740 \let\gls@orgAlph\@Alph
5741 \let\gls@orgRoman\@Roman
```

Redefine:

```
5742 \ifglswrallowprimitivemods
```

The redefinition of `\the` to use `\expandafter` solves the problem of `\the\csname c@<counter>\endcsname` but is only a partial solution to the problem of `\the\value`. With `\value`, `\c@page` is too deeply hidden and will be expanded too soon, but at least there won't be an error.

```

5743 \def\gls@the##1{%
5744 \ifx##1\c@page \gls@numberpage\else\gls@orgthe##1\fi}%
5745 \def\the{\expandafter\gls@the}%
5746 \def\gls@number##1{%
5747 \ifx##1\c@page \gls@numberpage\else\gls@orgnumber##1\fi}%
5748 \def\number{\expandafter\gls@number}%
5749 \fi
5750 \def\@arabic##1{%
5751 \ifx##1\c@page \gls@arabicpage\else\gls@orgarabic##1\fi}%
5752 \def\romannumeral##1{%
5753 \ifx##1\c@page \gls@romanpage\else\gls@orgromannumeral##1\fi}%
5754 \def\@Roman##1{%
5755 \ifx##1\c@page \gls@Romanpage\else\gls@orgRoman##1\fi}%
5756 \def\@alph##1{%
5757 \ifx##1\c@page \gls@alphpage\else\gls@orgalph##1\fi}%
5758 \def\@Alph##1{%
5759 \ifx##1\c@page \gls@Alphpage\else\gls@orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```

5760 \@wrglossarynumberhook

```

Prevent expansion:

```

5761 \gls@disablepagerefexpansion

```

Now store location in \@glslocref:

```

5762 \protected@xdef\@glslocref{\theglsentrycounter}%
5763 \endgroup

```

Escape any special characters. It's possible that with makeindex the separator might be a makeindex special character. Although not likely, it still needs to be taken into account.

```

5764 \@gls@checkmkidxchars\@glslocref

```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

5765 \expandafter\ifx\theHglentrycounter\theglsentrycounter\relax
5766 \def\@glo@counterprefix{}%
5767 \else
5768 \protected@edef\@glsHlocref{\theHglentrycounter}%
5769 \@gls@checkmkidxchars\@glsHlocref
5770 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5771 {\@glslocref}{\@glsHlocref}%
5772 }%
5773 \@do@gls@getcounterprefix
5774 \fi

```

De-tok label if required

```

5775 \edef\@gls@label{\glsdetoklabel{#1}}%

```

Write the information to file:

```

5776 \@do@wrglossary
5777 }

```

@do@wrglossary

```

5778 \newcommand*{\@do@wrglossary}{%

```


Determine whether to use xindy or makeindex syntax

```
5779 \ifglsxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```
5780 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5781 \def\@glo@range{}%
5782 \expandafter\if\@glo@prefix(\relax
5783 \def\@glo@range{:open-range}%
5784 \else
5785 \expandafter\if\@glo@prefix)\relax
5786 \def\@glo@range{:close-range}%
5787 \fi
5788 \fi
```

Write to the glossary file using xindy syntax.

```
5789 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5790 (indexentry :key (\csname glo@\@gls@label @index\endcsname)

5791 :locref \string"\@glo@counterprefix}{\@glslocref}\string" %
5792 :attr \string"\@gls@counter\@glo@suffix\string"
5793 \@glo@range
5794 )
5795 }%
5796 \else
```

Convert the format information into the format required for makeindex

```
5797 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5798 {\@glo@counterprefix}%
```

Write to the glossary file using makeindex syntax.

```
5799 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5800 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5801 \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
5802 \fi
5803 }
```

etcounterprefix Get the prefix that needs to be prepended to counter in order to get the hyper counter. (For example, with the standard article class and hyperref, \theequation needs to be prefixed with <section num>. to get the equivalent \theHequation.) NB this assumes that the prefix ends with a dot, which is the standard. (Otherwise it makes the xindy location classes more complicated.)

```
5804 \newcommand*\@gls@getcounterprefix[2]{%
5805 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5806 \ifx\@gls@thisloc\@gls@thisHloc
5807 \def\@glo@counterprefix{}%
5808 \else
5809 \def\@gls@get@counterprefix##1.##2\end@getprefix{%
5810 \def\@glo@tmp{##2}%
5811 \ifx\@glo@tmp\@empty
5812 \def\@glo@counterprefix{}%

```

```

5813     \else
5814         \def\@glo@counterprefix{##1}%
5815     \fi
5816 }%
5817 \@gls@get@counterprefix#2.#1\end@getprefix

Warn if no prefix can be formed.
5818     \ifx\@glo@counterprefix\@empty
5819         \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5820             prefixing^^Jlocation ‘#1’. You need to modify the
5821             definition of \string\theH\@gls@counter^^Jotherwise you
5822             will get the warning: “‘name{\@gls@counter.#1}’ has been^^J
5823             referenced but does not exist”}%
5824     \fi
5825 \fi
5826 }

```

1.15 Glossary Entry Cross-References

`\do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry’s label, the second must be in the form `[\<tag>]{\<list>}`, where `\<tag>` is a tag such as “see” and `\<list>` is a list of labels.

```

5827 \newcommand{\@do@seeglossary}[2]{%
5828 \def\@gls@xref{#2}%
5829 \@onelevel@sanitize\@gls@xref
5830 \@gls@checkmkidxchars\@gls@xref
5831 \ifglxsindy
5832   \gls@glossary{\csname glo@#1@type\endcsname}{%
5833     (indexentry
5834       :key (\csname glo@#1@index\endcsname)
5835       :xref (\string"\@gls@xref\string")
5836       :attr \string"see\string"
5837     )
5838   }%
5839 \else
5840   \gls@glossary{\csname glo@#1@type\endcsname}{%
5841     \string\glossaryentry{\csname glo@#1@index\endcsname
5842     \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5843 \fi
5844 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5845 \def\@gls@fixbraces#1#2#3\@nil{%
5846   \ifx#2[\relax
5847     \@gls@fixbraces#1#2#3\@end@fixbraces
5848   \else
5849     \def#1{{#2#3}}%
5850   \fi

```

5851 }

@@gls@fixbraces

```
5852 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%  
5853   \def#1{[#2]{#3}}%  
5854 }
```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```
5855 \newrobustcmd*{\glssee}[3][\seename]{%  
5856   \@do@seeglossary{#2}{#1}{#3}}  
5857 \newcommand*{\@glssee}[3][\seename]{%  
5858   \glssee[#1]{#3}{#2}}
```

`\glsseeformat` The first argument specifies what tag to use (e.g. “see”), the second argument is a comma-separated list of labels. The final argument (the location) is ignored.

```
5859 \newrobustcmd*{\glsseeformat}[3][\seename]{%  
5860   \emph{#1} \glsseelist{#2}}
```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```
5861 \newrobustcmd*{\glsseelist}[1]{%
```

If there is only one item in the list, set the last separator to do nothing.

```
5862   \let\@gls@dolast\relax
```

Don’t display separator on the first iteration of the loop

```
5863   \let\@gls@donext\relax
```

Iterate through the labels

```
5864   \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
5865     \ifx\@xfor@nextelement\@nnil  
5866       \@gls@dolast  
5867     \else  
5868       \@gls@donext  
5869     \fi
```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```
5870     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5871     \let\@gls@dolast\glsseelastsep  
5872     \let\@gls@donext\glsseesep  
5873   }%  
5874 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5875 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
5876 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5877 \newrobustcmd*{\glsseeitem}[1]{\gls hyperlink[\glsseeitemformat{#1}]{#1}}
```

`\glsseeitemformat` As from v3.0, default is to use `\glsentrytext` instead of `\glsentryname`. (To avoid problems with the name key being sanitized, although this is no longer a problem now.)

```
5878 \newcommand*{\glsseeitemformat}[1]{\glsentrytext{#1}}
```

1.16 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[<key-val list>]`. If the type key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`\save@numberlist` Provide command to store number list.

```
5879 \newcommand*{\gls@save@numberlist}[1]{%
5880   \ifglssavenumberlist
5881     \toks@{#1}%
5882     \edef\@do@writeaux@info{%
5883       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5884     }%
5885     \@onelevel@sanitize\@do@writeaux@info
5886     \protected@write\@auxout{}\@do@writeaux@info{%
5887       \fi
5888 }
```

`\noprintglossary` Warn the user if they have forgotten `\printglossaries` or `\printglossary`. (Will be suppressed if there is at least one occurrence of `\printglossary`. There is no check to ensure that there is a `\printglossary` for each defined glossary.)

```
5889 \newcommand*{\warn@noprintglossary}{}%
```

`\printglossary` The TOC title needs to be processed in a different manner to the main title in case the translator and hyperref packages are both being used.

```
5890 \ifcsundef{printglossary}{}%
5891 {%
  If \printglossary is already defined, issue a warning and undefine it.
5892   \@gls@warnonglossdefined
5893   \undef\printglossary
5894 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5895 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{}%
```

```

5896 \@printglossary{#1}{\@print@glossary}%
5897 }

```

The `\printglossaries` command will do `\printglossary` for each glossary type that has been defined. It is better to use `\printglossaries` rather than individual `\printglossary` commands to ensure that you don't forget any new glossaries you may have created. It also makes it easier to chop and change the value of the `acronym` package option. However, if you want to list the glossaries in a different order, or if you want to set the title or table of contents entry, or if you want to use different glossary styles for each glossary, you will need to use `\printglossary` explicitly for each glossary type.

`\printglossaries`

```

5898 \newcommand*{\printglossaries}{%
5899   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5900 }

```

`\printnoidxglossary` Provide an alternative to `\printglossary` that doesn't require an external indexing application. Entries won't be sorted and the location list will be empty.

```

5901 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5902   \@printglossary{#1}{\@print@noidx@glossary}%
5903 }

```

`\printnoidxglossaries` Analogous to `\printglossaries`

```

5904 \newcommand*{\printnoidxglossaries}{%
5905   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5906 }

```

`\printgloss@setsort` Initialise to do nothing.

```

5907 \newcommand*{\@printgloss@setsort}{}

```

`\preglossaryhook`

```

5908 \newcommand*{\@gls@preglossaryhook}{}

```

`\@printglossary` Sets up the glossary for either `\printglossary` or `\printnoidxglossary`. The first argument is the options list, the second argument is the handler macro that deals with the actual glossary.

```

5909 \newcommand{\@printglossary}[2]{%
  Set up defaults.
5910   \def\@glo@type{\glsdefaulttype}%
5911   \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%
5912   \def\glossarytoctitle{\glossarytitle}%
5913   \let\org@glossarytitle\glossarytitle
5914   \def\@glossarystyle{%
5915     \ifx\@glossary@default@style\relax
5916       \GlossariesWarning{No default glossary style provided \MessageBreak

```

```

5917         for the glossary ‘\@glo@type’. \MessageBreak
5918         Using deprecated fallback. \MessageBreak
5919         To fix this set the style with \MessageBreak
5920         \string\setglossarystyle\space or use the \MessageBreak
5921         style key=value option}%
5922     \fi
5923 }%
5924 \def\gls@dotoc@title{\gls@set@toc@title{\@glo@type}}%

    Store current value of \glossaryentrynumbers. (This may be changed via the optional ar-
    gument)
5925 \let\org@glossaryentrynumbers\glossaryentrynumbers

    Localise the effects of the optional argument
5926 \bgroup

    Activate or deactivate sort key:
5927 \@printgloss@setsort

    Determine settings specified in the optional argument.
5928 \setkeys{printgloss}{#1}%

    Does the glossary exist?
5929 \ifglossaryexists{\@glo@type}%
5930 {%

    If title has been set, but toctitle hasn't, make toctitle the same as given title (rather than the
    title used when the glossary was defined)
5931 \ifx\glossarytitle\org@glossarytitle
5932 \else
5933 \expandafter\let\csname @glo@type @title\endcsname
5934 \glossarytitle
5935 \fi

    Allow a high-level user command to indicate the current glossary
5936 \let\currentglossary\@glo@type

    Enable individual number lists to be suppressed.
5937 \let\org@glossaryentrynumbers\glossaryentrynumbers
5938 \let\glsnonextpages\@glsnonextpages

    Enable individual number list to be activated:
5939 \let\glsnextpages\@glsnextpages

    Enable suppression of description terminators.
5940 \let\nopostdesc\@nopostdesc

    Set up the entry for the TOC
5941 \gls@dotoc@title

    Set the glossary style
5942 \@glossarystyle

```

Added a way to fetch the current entry label (v3.08 updated for new `\glossentry` and `\subglossentry`, but this is now only needed for backward compatibility):

```

5943 \let\gls@org@glossaryentryfield\glossentry
5944 \let\gls@org@glossarysubentryfield\subglossentry
5945 \renewcommand{\glossentry}[1]{%
5946   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5947   \gls@org@glossaryentryfield{##1}%
5948 }%
5949 \renewcommand{\subglossentry}[2]{%
5950   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5951   \gls@org@glossarysubentryfield{##1}{##2}%
5952 }%

```

```

5953 \@gls@preglossaryhook

```

Now do the handler macro that deals with the actual glossary:

```

5954   #2%
5955 }%
5956 {\GlossariesWarning{Glossary ‘\@glo@type’ doesn’t exist}}%

```

End the current scope

```

5957 \egroup

```

Reset `\glossaryentrynumbers`

```

5958 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers

```

Suppress warning about no `\printglossary`

```

5959 \global\let\warn@noprintglossary\relax
5960 }

```

`@print@glossary` Internal workings of `\printglossary` dealing with reading the external file.

```

5961 \newcommand{\@print@glossary}{%

```

Some macros may end up being expanded into internals in the glossary, so need to make `@` a letter. (Unlikely to be a problem since v3.08a but kept for backward compatibility.)

```

5962 \makeatletter

```

Input the glossary file, if it exists.

```

5963 \@input@{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%

```

If the glossary file doesn't exist, do `\null`. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```

5964 \IfFileExists{\jobname.\csname @glo@type@\@glo@type @in\endcsname}%
5965 {}%
5966 {\null}%

```

If `xindy` is being used, need to write the language dependent information to the `.aux` file for `makeglossaries`.

```

5967 \ifglxindy
5968   \ifcsundef{\@glo@type @language}%

```

```

5969   {%
5970     \edef\@do@auxoutstuff{%
5971       \noexpand\AtEndDocument{%

```

If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

5972         \noexpand\immediate\noexpand\write\@auxout{%
5973           \string\providecommand\string\@xdylanguage[2]{}}%
5974         \noexpand\immediate\noexpand\write\@auxout{%
5975           \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5976       }%
5977   }%
5978 }%
5979 {%
5980   \edef\@do@auxoutstuff{%
5981     \noexpand\AtEndDocument{%
5982       \noexpand\immediate\noexpand\write\@auxout{%
5983         \string\providecommand\string\@xdylanguage[2]{}}%
5984       \noexpand\immediate\noexpand\write\@auxout{%
5985         \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5986           @language\endcsname}}%
5987     }%
5988   }%
5989 }%
5990 \@do@auxoutstuff
5991 \edef\@do@auxoutstuff{%
5992   \noexpand\AtEndDocument{%

```

If the user removes the glossaries package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

5993         \noexpand\immediate\noexpand\write\@auxout{%
5994           \string\providecommand\string\@gls@codepage[2]{}}%
5995         \noexpand\immediate\noexpand\write\@auxout{%
5996           \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5997       }%
5998   }%
5999   \@do@auxoutstuff
6000 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

6001 \renewcommand*{\@warn@nomakeglossaries}{%
6002   \GlossariesWarningNoLine{\string\makeglossaries\space
6003     hasn't been used,^^Jthe glossaries will not be updated}%
6004 }%
6005 }

```

The sort macros all have the syntax:

`\@glo@sortmacro@<order>{<type>}`

where `<order>` is the sort order as specified by the sort key and `<type>` is the glossary type. (The referenced entry list is stored in `\@glsref@<type>`. The actual sorting is done by `\@glo@sortentries{<handler>}{<type>}`).

`glo@sortentries`

```

6006 \newcommand*{\@glo@sortentries}[2]{%
6007   \glosortentrieswarning
6008   \def\@glo@sortinglist{}%
6009   \def\@glo@sortinghandler{#1}%
6010   \edef\@glo@type{#2}%
6011   \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
6012   \csdef{\@glsref@#2}{}%
6013   \@for\@this@label:=\@glo@sortinglist\do{%
      Has this entry already been added?
6014     \xifinlistcs{\@this@label}{\@glsref@#2}%
6015     {}%
6016     {%
6017       \listcsxadd{\@glsref@#2}{\@this@label}%
6018     }%
6019     \ifcsdef{\@glo@sortingchildren@\@this@label}%
6020     {%
6021       \@glo@addchildren{#2}{\@this@label}%
6022     }%
6023     {}%
6024   }%
6025 }
```

`glo@addchildren`

`\@glo@addchildren{<type>}{<parent>}`

```

6026 \newcommand*{\@glo@addchildren}[2]{%
      Scope to allow nesting.
6027   \bgroup
6028   \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
6029   \@for\@this@childlabel:=\@glo@childlist\do
6030   {%
      Check this label hasn't already been added.
6031     \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
6032     {}%
6033     {%
6034       \listcsxadd{\@glsref@#1}{\@this@childlabel}%
6035     }%
      Does this child have children?
6036     \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%

```

```

6037      {%
6038      \glo@addchildren{#1}{\@this@childlabel}%
6039      }%
6040      {%
6041      }%
6042      }%
6043 \egroup
6044 }

@do@sortentries

6045 \newcommand*{\@glo@do@sortentries}[1]{%
6046   \ifglshasparent{#1}%
6047   {%
        This entry has a parent, so add it to the child list
6048   \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
6049   \ifcsundef{glo@sortingchildren@\@glo@parent}%
6050   {%
6051     \csdef{glo@sortingchildren@\@glo@parent}{}%
6052   }%
6053   {}%
6054   \expandafter\@glo@sortedinsert
6055   \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
        Has the parent been added?
6056   \xifinlistcs{\@glo@parent}{@glstref@\@glo@type}%
6057   {%
        Yes, it has so do nothing.
6058   }%
6059   {%
        No, it hasn't so add it now.
6060   \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
6061   }%
6062   }%
6063   {%
6064   \@glo@sortedinsert{\@glo@sortinglist}{#1}%
6065   }%
6066   }

```

```

\lo@sortedinsert \glo@sortedinsert{<list>}{<entry label>}

```

Insert into list.

```

6067 \newcommand*{\@glo@sortedinsert}[2]{%
6068   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
6069 }%

```

The sort handlers need to be in the form required by datatool's `\dtl@sortlist` macro. These must set the count register `\dtl@sortresult` to either `-1` (`#1` less than `#2`), `0` (`#1 = #2`) or `+1` (`#1` greater than `#2`).

orthandler@word

```
6070 \newcommand*{\@glo@sorthandler@word}[2]{%
6071   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
6072   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
6073   \edef\glo@do@compare{%
6074     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
6075     {\expandonce\@gls@sort@B}%
6076     {\expandonce\@gls@sort@A}%
6077   }%
6078   \glo@do@compare
6079 }
```

thandler@letter

```
6080 \newcommand*{\@glo@sorthandler@letter}[2]{%
6081   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
6082   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
6083   \edef\glo@do@compare{%
6084     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
6085     {\expandonce\@gls@sort@B}%
6086     {\expandonce\@gls@sort@A}%
6087   }%
6088   \glo@do@compare
6089 }
```

orthandler@case Case-sensitive sort.

```
6090 \newcommand*{\@glo@sorthandler@case}[2]{%
6091   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
6092   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
6093   \edef\glo@do@compare{%
6094     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
6095     {\expandonce\@gls@sort@B}%
6096     {\expandonce\@gls@sort@A}%
6097   }%
6098   \glo@do@compare
6099 }
```

thandler@nocase Case-insensitive sort.

```
6100 \newcommand*{\@glo@sorthandler@nocase}[2]{%
6101   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
6102   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
6103   \edef\glo@do@compare{%
6104     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
6105     {\expandonce\@gls@sort@B}%
6106     {\expandonce\@gls@sort@A}%
6107   }%
6108   \glo@do@compare
6109 }
```

```

@sortmacro@word  Sort macro for ‘word’
6110 \newcommand*{\@glo@sortmacro@word}[1]{%
6111   \ifdefstring{\@glo@default@sorttype}{standard}%
6112   {%
6113     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
6114   }%
6115   {%
6116     \PackageError{glossaries}{Conflicting sort options:^^J
6117       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6118       \string\printnoidxglossary[sort=word]}{}}%
6119   }%
6120 }

ortmacro@letter  Sort macro for ‘letter’
6121 \newcommand*{\@glo@sortmacro@letter}[1]{%
6122   \ifdefstring{\@glo@default@sorttype}{standard}%
6123   {%
6124     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
6125   }%
6126   {%
6127     \PackageError{glossaries}{Conflicting sort options:^^J
6128       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6129       \string\printnoidxglossary[sort=letter]}{}}%
6130   }%
6131 }

tmacro@standard  Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)
6132 \newcommand*{\@glo@sortmacro@standard}[1]{%
6133   \ifdefstring{\@glo@default@sorttype}{standard}%
6134   {%
6135     \ifcsdef{\@glo@sorthandler@\glsorder}%
6136     {%
6137       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
6138     }%
6139     {%
6140       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
6141     }%
6142   }%
6143   {%
6144     \PackageError{glossaries}{Conflicting sort options:^^J
6145       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6146       \string\printnoidxglossary[sort=standard]}{}}%
6147   }%
6148 }

@sortmacro@case  Sort macro for ‘case’
6149 \newcommand*{\@glo@sortmacro@case}[1]{%
6150   \ifdefstring{\@glo@default@sorttype}{standard}%
6151   {%

```

```

6152 \glo@sortentries{\glo@sorthandler@case}{#1}%
6153 }%
6154 {%
6155 \PackageError{glossaries}{Conflicting sort options:^^J
6156 \string\usepackage[sort=\glo@default@sorttype]{glossaries}^^J
6157 \string\printnoidxglossary[sort=case]}{}%
6158 }%
6159 }

```

ortmacro@nocase Sort macro for ‘nocase’

```

6160 \newcommand*\glo@sortmacro@nocase}[1]{%
6161 \ifdefstring{\glo@default@sorttype}{standard}%
6162 {%
6163 \glo@sortentries{\glo@sorthandler@nocase}{#1}%
6164 }%
6165 {%
6166 \PackageError{glossaries}{Conflicting sort options:^^J
6167 \string\usepackage[sort=\glo@default@sorttype]{glossaries}^^J
6168 \string\printnoidxglossary[sort=nocase]}{}%
6169 }%
6170 }

```

o@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@{*type*}.

```

6171 \newcommand*\glo@sortmacro@def}[1]{%
6172 \def\glo@sortinglist{}%
6173 \for@gl@sentries[#1]{\gls@thislabel}%
6174 {%
6175 \xifinlistcs{\gls@thislabel}{\gls@ref@#1}%
6176 {%
6177 \listadd{\glo@sortinglist}{\gls@thislabel}%
6178 }%
6179 }%

```

Hasn’t been referenced.

```

6180 }%
6181 }%
6182 \cslet{\gls@ref@#1}{\glo@sortinglist}%
6183 }

```

ortmacro@def@do This won’t include parent entries that haven’t been referenced.

```

6184 \newcommand*\glo@sortmacro@def@do}[1]{%
6185 \ifinlistcs{#1}{\gls@ref@\glo@type}%
6186 {}%
6187 {%
6188 \listcsadd{\gls@ref@\glo@type}{#1}%
6189 }%
6190 \ifcsdef{\glo@sortingchildren@#1}%
6191 {%
6192 \glo@addchildren{\glo@type}{#1}%

```

```

6193 }%
6194 {}%
6195 }

```

`@sortmacro@use` Sort macro for ‘use’. (No sorting is required, as the entries are already in order of use, so do nothing.)

```

6196 \newcommand*{\@glo@sortmacro@use}[1]{}

```

`@noidx@glossary` Glossary handler for `\printnoidxglossary` which doesn’t use an indexing application. Since `\printnoidxglossary` may occur at the start of the document, we can’t just check if an entry has been used. Instead, the first pass needs to write information to the aux file every time an entry is referenced. This needs to be read in on the second run and stored in a list corresponding to the appropriate glossary.

```

6197 \newcommand*{\@print@noidx@glossary}{%
6198   \ifcsdef{@glsref@{\@glo@type}}%
6199   {%

```

Sort the entries:

```

6200   \ifcsdef{@glo@sortmacro@{\@glo@sorttype}}%
6201   {%
6202     \csuse{@glo@sortmacro@{\@glo@sorttype}}{\@glo@type}%
6203   }%
6204   {%
6205     \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
6206   }%

```

Do the glossary heading and preamble

```

6207   \glossarysection[\glossarytoctitle]{\glossarytitle}%
6208   \glossarypreamble

```

The glossary style might use a tabular-like environment, which may cause scoping problems when setting the current letter group. The predefined tabular-like styles don’t support letter group headings, but there’s nothing to stop the user from defining their own custom style that might, so any redefinition of this command within `theglossary` will have to be done globally.

```

6209   \def\@gls@currentlettergroup{%
6210     \begin{theglossary}%
6211     \glossaryheader
6212     \glsresetentrylist

```

Iterate through the entries.

```

6213   \forlistcsloop{\@gls@noidx@do}{@glsref@{\@glo@type}}%

```

Finally end the glossary and do the postamble:

```

6214   \end{theglossary}%
6215   \glossarypostamble
6216 }%
6217 {%
6218   \@gls@noref@warn{\@glo@type}%
6219 }%
6220 }

```

\glo@grabfirst

```
6221 \def\glo@grabfirst#1#2\@nil{%
6222   \def\@gls@firsttok{#1}%
6223   \ifdefempty\@gls@firsttok
6224     {%
6225       \def\@glo@thislettergrp{0}%
6226     }%
6227     {%
```

Sanitize it:

```
6228   \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
6229   \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
6230 }%
6231 }
```

\@glo@grabfirst

```
6232 \def\@glo@grabfirst#1#2\@nil{%
6233   \ifdefempty\@glo@thislettergrp
6234     {%
6235       \def\@glo@thislettergrp{glssymbols}%
6236     }%
6237     {%
6238       \count@=\uccode'#1\relax
6239       \ifnum\count@=0\relax
6240         \def\@glo@thislettergrp{glssymbols}%
6241       \else
6242         \ifdefstring\@glo@sorttype{case}%
6243           {%
6244             \count@='#1\relax
6245           }%
6246           {%
6247             }%
6248           \edef\@glo@thislettergrp{\the\count@}%
6249         \fi
6250       }%
6251 }
```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label.
This only allows one sublevel.

```
6252 \newcommand{\@gls@noidx@do}[1]{%
```

Get this entry's location list

```
6253   \global\letcs{\@gls@loclist}{\glo@\glsdetoklabel{#1}@loclist}%
```

Does this entry have a parent?

```
6254   \ifglshasparent{#1}%
6255   {%
```

Has a parent.

```
6256 \gls@level=\csuse{glo@glsdetoklabel{#1}@level}\relax
6257 \ifdefvoid{\@gls@loclist}
6258 {%
6259   \subglossentry{\gls@level}{#1}{}%
6260 }%
6261 {%
6262   \subglossentry{\gls@level}{#1}%
6263   {%
6264     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
6265   }%
6266 }%
6267 }%
6268 {%
```

Doesn't have a parent Get this entry's sort key

```
6269 \letcs{\@gls@sort}{glo@glsdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
6270 \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
6271 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
6272 {}%
6273 {%
```

Do the group header:

```
6274 \ifdefempty{\@gls@currentlettergroup}{}%
6275 {%
```

The group skip may start a new scope, so make a global assignment.

```
6276 \global\let\@glo@thislettergrp\@glo@thislettergrp
6277 \glsgroupskip
6278 }%
6279 \glsgroupheading{\@glo@thislettergrp}%
6280 }%

6281 \global\let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
6282 \ifdefvoid{\@gls@loclist}
6283 {%
6284   \glossentry{#1}{}%
6285 }%
6286 {%
6287   \glossentry{#1}%
6288   {%
6289     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
6290   }%
6291 }%
6292 }%
6293 }
```


glsnoidxloclist `\glsnoidxloclist{<list cs>}`

Display location list.

```
6294 \newcommand*{\glsnoidxloclist}[1]{%
6295   \def\@gls@noidxloclist@sep{}%
6296   \def\@gls@noidxloclist@prev{}%
6297   \forlistloop{\glsnoidxloclisthandler}{#1}%
6298 }
```

xloclisthandler Handler for location list iterator.

```
6299 \newcommand*{\glsnoidxloclisthandler}[1]{%
6300   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6301   {%
```

Same as previous location so skip.

```
6302   }%
6303   {%
6304     \@gls@noidxloclist@sep
6305     #1%
6306     \def\@gls@noidxloclist@sep{\delimN}%
6307     \def\@gls@noidxloclist@prev{#1}%
6308   }%
6309 }
```

yloclisthandler Handler for location list iterator when used with \glsdisplaynumberlist.

```
6310 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
6311   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6312   {%
```

Same as previous location so skip.

```
6313   }%
6314   {%
6315     \@gls@noidxloclist@sep
6316     \@gls@noidxloclist@prev
6317     \def\@gls@noidxloclist@prev{#1}%
6318   }%
6319 }
```

noidxdisplayloc `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
6320 \newcommand*{\glsnoidxdisplayloc}[4]{%
6321   \setentrycounter[#1]{#2}%
6322   \csuse{#3}{#4}%
6323 }
```

`\@gls@reference` `\@gls@reference{<type>}{<label>}{<loc>}`

Identifies that a reference has been used (for use in the aux file). All entries must be defined in the preamble.

```
6324 \newcommand*{\@gls@reference}[3]{%
```

Add to label list

```
6325 \glsdoifexistsorwarn{#2}%
6326 {%
6327 \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
6328 \ifinlistcs{#2}{@glsref@#1}%
6329 {}%
6330 {\listcsgadd{@glsref@#1}{#2}}}%
```

Add to location list

```
6331 \ifcsundef{glo@glstdetoklabel{#2}@loclist}%
6332 {\csgdef{glo@glstdetoklabel{#2}@loclist}{}}%
6333 {}%
6334 \listcsgadd{glo@glstdetoklabel{#2}@loclist}{#3}%
6335 }%
6336 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```
6337 \define@key{printgloss}{type}{\def@glo@type{#1}}
```

The title key sets the title used in the glossary section header. This overrides the title used in `\newglossary`.

```
6338 \define@key{printgloss}{title}{%
6339 \def@glossarytitle{#1}%
6340 \let@gls@dotoc@title\relax
6341 }
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
6342 \define@key{printgloss}{toctitle}{%
6343 \def@glossarytoctitle{#1}%
6344 \let@gls@dotoc@title\relax
6345 }
```

The style key sets the glossary style (but only for the given glossary).

```
6346 \define@key{printgloss}{style}{%
6347 \ifcsundef{@glsstyle@#1}%
6348 {%
6349 \PackageError{glossaries}%
6350 {Glossary style ‘#1’ undefined}{}%
6351 }%
6352 {%
6353 \def@glossarystyle{\setglossentrycompatibility
6354 \csname @glsstyle@#1\endcsname}%
6355 }%
6356 }
```

The `numberedsection` key determines if this glossary should be in a numbered section.

```

6357 \define@choicekey{printgloss}{numberedsection}%
6358 [\gls@numberedsection@val\gls@numberedsection@nr]%
6359 {false,nolabel,autolabel,nameref}[nolabel]%
6360 {%
6361   \ifcase\gls@numberedsection@nr\relax
6362     \renewcommand*{\@@glossarysecstar}{*}%
6363     \renewcommand*{\@@glossaryseclabel}{}%
6364   \or
6365     \renewcommand*{\@@glossarysecstar}{}%
6366     \renewcommand*{\@@glossaryseclabel}{}%
6367   \or
6368     \renewcommand*{\@@glossarysecstar}{}%
6369     \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
6370   \or
6371     \renewcommand*{\@@glossarysecstar}{*}%
6372     \renewcommand*{\@@glossaryseclabel}{%
6373       \protected@edef\@currentlabelname{\glossarytoctitle}%
6374       \label{\glsautoprefix\@glo@type}}%
6375   \fi
6376 }

```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

6377 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
6378   \csuse{glsnogroupskip#1}%
6379 }

```

The `nopostdot` key has the same effect as the package option of the same name.

```

6380 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
6381   \csuse{glsnopostdot#1}%
6382 }

```

CounterLabelPrefix Make it easier to redefine the label prefix.

```

6383 \newcommand*{\GlsEntryCounterLabelPrefix}{glsentry-}

```

The conditionals have been moved inside the appropriate commands to make it easier for the user to redefine them in the preamble and selectively switch the counter display on and off. Previously the helper commands were redefined by the `entrycounter` option, which would counteract any earlier customisation.

The `entrycounter` key is the same as the package option but localised to the current glossary.

```

6384 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
6385   \csuse{glsentrycounter#1}%
6386   \@gls@define@glossaryentrycounter
6387 }

```

The `subentrycounter` key is the same as the package option but localised to the current glossary. Note that this doesn't affect the master/slave counter attributes, which occurs if `subentrycounter` and `entrycounter` package options are set to true.

```

6388 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6389   \csuse{glssubentrycounter#1}%
6390   \@gls@define@glossarysubentrycounter
6391 }

```

The `nonumberlist` key determines if this glossary should have a number list.

```

6392 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
6393 \ifglsnonumberlist
6394   \def\glossaryentrynumbers##1{%
6395 \else
6396   \def\glossaryentrynumbers##1{##1}%
6397 \fi}

```

The `sort` key sets the glossary sort handler (`\printnoidxglossary` only).

```

6398 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

`@assign@sortkey` Issue error if used with `\printglossary`

```

6399 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6400   \PackageError{glossaries}{'sort' key not permitted with
6401   \string\printglossary}%
6402   {The 'sort' key may only be used with \string\printnoidxglossary}%
6403 }

```

`@assign@sortkey` For use with `\printnoidxglossary`

```

6404 \newcommand*{\@glo@assign@sortkey}[1]{%
6405   \def\@glo@sorttype{#1}%
6406 }

```

`@glsnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonextpages` is placed in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6407 \newcommand*{\@glsnonextpages}{%
6408   \gdef\glossaryentrynumbers##1{%
6409     \glsresetentrylist
6410   }%
6411 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is placed in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6412 \newcommand*{\@glsnextpages}{%
6413   \gdef\glossaryentrynumbers##1{%
6414     ##1\glsresetentrylist}}

```

`sresetentrylist` Resets `\glossaryentrynumbers`

```

6415 \newcommand*{\glsresetentrylist}{%
6416   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```

6417 \newcommand*{\glsnonextpages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

6418 \newcommand*{\glsnextpages}{}

```

Process entrycounter and then subentrycounter options (this ensures the sub-counter can pick up the main counter as the master if required):

```

6419 \@gls@define@glossaryentrycounter
6420 \@gls@define@glossarysubentrycounter

```

`subentrycounter` Resets the `glossarysubentry` counter.

```

6421 \newcommand*{\glsresetsubentrycounter}{%
6422   \ifglssubentrycounter
6423     \setcounter{glossarysubentry}{0}%
6424   \fi
6425 }

```

`subentrycounter` Resets the `glossaryentry` counter.

```

6426 \newcommand*{\glsresetentrycounter}{%
6427   \ifglsentrycounter
6428     \setcounter{glossaryentry}{0}%
6429   \fi
6430 }

```

`\glsstepentry` Advance the `glossaryentry` counter if in use. The argument is the label associated with the entry.

```

6431 \newcommand*{\glsstepentry}[1]{%
6432   \ifglsentrycounter
6433     \refstepcounter{glossaryentry}%
6434     \label{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6435   \fi
6436 }

```

`glsstepsubentry` Advance the `glossarysubentry` counter if in use. The argument is the label associated with the subentry.

```

6437 \newcommand*{\glsstepsubentry}[1]{%
6438   \ifglssubentrycounter
6439     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6440     \refstepcounter{glossarysubentry}%
6441     \label{\GlsEntryCounterLabelPrefix\currentglssubentry}%
6442   \fi
6443 }

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```
6444 \newcommand*{\glsrefentry}[1]{%
6445   \ifglentrycounter
6446     \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6447   \else
6448     \ifglssubentrycounter
6449       \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6450     \else
6451       \gls{#1}%
6452     \fi
6453   \fi
6454 }
```

`entrycounterlabel` Defines how to display the glossaryentry counter.

```
6455 \newcommand*{\glentrycounterlabel}{%
6456   \ifglentrycounter
6457     \theglossaryentry.\space
6458   \fi
6459 }
```

`entrycounterlabel` Defines how to display the glossarysubentry counter.

```
6460 \newcommand*{\glssubentrycounterlabel}{%
6461   \ifglssubentrycounter
6462     \theglossarysubentry)\space
6463   \fi
6464 }
```

`\glentryitem` Step and display glossaryentry counter, if appropriate.

```
6465 \newcommand*{\glentryitem}[1]{%
6466   \ifglentrycounter
6467     \glsstepentry{#1}\glentrycounterlabel
6468   \else
6469     \glsresetsubentrycounter
6470   \fi
6471 }
```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```
6472 \newcommand*{\glssubentryitem}[1]{%
6473   \ifglssubentrycounter
6474     \glsstepsubentry{#1}\glssubentrycounterlabel
6475   \fi
6476 }
```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
6477 \ifcsundef{theglossary}%
6478 {%
6479   \newenvironment{theglossary}{}{}}%
```

```

6480 }%
6481 {%
6482   \@gls@warnontheglossdefined
6483   \renewenvironment{theglossary}{-}{-}%
6484 }

```

The glossary header is given by `\glossaryheader`. This forms part of the glossary style, and must indicate what should appear immediately after the start of the `theglossary` environment. (For example, if the glossary uses a tabular-like environment, it may be used to set the header row.) Note that if you don't want a header row, the glossary style must redefine `\glossaryheader` to do nothing.

`\glossaryheader`

```

6485 \newcommand*{\glossaryheader}{}

```

`\glstarget`

```
\glstarget{<label>}{<name>}
```

Provide user interface to `\glstarget` to make it easier to modify the glossary style in the document.

```

6486 \newcommand*{\glstarget}[2]{\@glstarget{\glo@linkprefix#1}{#2}}

```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

`\compatibleglossentry`

```
\glossentry{<label>}{<page-list>}
```

```

6487 \providecommand*{\compatibleglossentry}[2]{%
6488   \toks@{#2}%
6489   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6490     {\noexpand\glsnamefont
6491       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6492     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6493     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6494     {\the\toks@}%
6495   }%
6496   \@do@glossentry
6497 }

```

`\glossentryname`

```

6498 \newcommand*{\glossentryname}[1]{%
6499   \glsdoifexistsorwarn{#1}%
6500   {%
6501     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6502     \expandafter\glsnamefont\expandafter{\glo@name}%

```

```

6503 }%
6504 }

```

`\Glossentryname`

```

6505 \newcommand*{\Glossentryname}[1]{%
6506   \glsdoifexistsorwarn{#1}%
6507   {%
6508     \glsnamefont{\Glsentryname{#1}}%
6509   }%
6510 }

```

`\glossentrydesc`

```

6511 \newcommand*{\glossentrydesc}[1]{%
6512   \glsdoifexistsorwarn{#1}%
6513   {%
6514     \glsentrydesc{#1}%
6515   }%
6516 }

```

`\Glossentrydesc`

```

6517 \newcommand*{\Glossentrydesc}[1]{%
6518   \glsdoifexistsorwarn{#1}%
6519   {%
6520     \Glsentrydesc{#1}%
6521   }%
6522 }

```

`\glossentrysymbol`

```

6523 \newcommand*{\glossentrysymbol}[1]{%
6524   \glsdoifexistsorwarn{#1}%
6525   {%
6526     \glsentrysymbol{#1}%
6527   }%
6528 }

```

`\Glossentrysymbol`

```

6529 \newcommand*{\Glossentrysymbol}[1]{%
6530   \glsdoifexistsorwarn{#1}%
6531   {%
6532     \Glsentrysymbol{#1}%
6533   }%
6534 }

```

`\lesubglossentry`

`\subglossentry{<level>}{<label>}{<page-list>}`

```

6535 \providecommand*{\compatiblesubglossentry}[3]{%
6536   \toks@{#3}%

```



```

6537 \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6538 {#2}}%
6539 {\noexpand\glsnamefont
6540 {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6541 {\expandafter\expandonce\csname glo@#2@desc\endcsname}}%
6542 {\expandafter\expandonce\csname glo@#2@symbol\endcsname}}%
6543 {\the\toks@}%
6544 }%
6545 \@do@subglossentry
6546 }

```

rycompatibility

```

6547 \newcommand*\setglossentrycompatibility{%
6548 \let\glossentry\compatibleglossentry
6549 \let\subglossentry\compatiblesubglossentry
6550 }
6551 \setglossentrycompatibility

```

ssaryentryfield

```

\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}
{<page-list>}

```

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```

6552 \newcommand{\glossaryentryfield}[5]{%
6553 \GlossariesWarning
6554 {Deprecated use of \string\glossaryentryfield.^^J
6555 I recommend you change to \string\glossentry.^^J
6556 If you've just upgraded, try removing your gls auxiliary
6557 files^^J and recompile}%
6558 \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

rysubentryfield

```

\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}
{<page-list>}

```

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore *<symbol>*. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```

6559 \newcommand*\glossarysubentryfield}[6]{%
6560 \GlossariesWarning
6561 {Deprecated use of \string\glossarysubentryfield.^^J
6562 I recommend you change to \string\subglossentry.^^J
6563 If you've just upgraded, try removing your gls auxiliary
6564 files^^J and recompile}%
6565 \glstarget{#2}{\strut}#4. #6\par}

```

Within each glossary, the entries form distinct groups which are determined by the first character of the sort key. When using `makeindex`, there will be a maximum of 28 groups: symbols, numbers, and the 26 alphabetical groups A, ..., Z. If you use `xindy` the groups will depend on whatever alphabet is used. This is determined by the language or custom alphabets can be created in the `xindy` style file. The command `\glsgroupskip` specifies what to do between glossary groups. Glossary styles must redefine this command. (Note that `\glsgroupskip` only occurs between groups, not at the start or end of the glossary.)

`\glsgroupskip`

```
6566 \newcommand*{\glsgroupskip}{}

```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgroupheading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glssymbols`, `glsnumbers`, A, ..., Z. Glossary styles must redefine this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
6567 \newcommand*{\glsgroupheading}[1]{}

```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an a, while entries belonging to another group could be defined so that the sort key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgetgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by *<label>*. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

`\glsgetgrouptitle`

```
6568 \newcommand*{\glsgetgrouptitle}[1]{%
6569   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6570   \@gls@grptitle
6571 }
```

`\@gls@getgrouptitle`

Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
6572 \newcommand*{\@gls@getgrouptitle}[2]{%
```

Even if the argument appears to be a single letter, it won't be considered a single letter by `\dtl@ifsingle` if it's an active character.

```

6573 \dtl@ifsingle{#1}%
6574 {%
6575   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6576 }%
6577 {%
6578   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6579               or test{\ifstrequal{#1}{glsnumbers}}}%
6580   {%
6581     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6582   }%
6583   {%
6584     \def#2{#1}%
6585   }%
6586 }%
6587 }

```

`x@getgrouptitle` Version for the no-indexing app option:

```

6588 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6589   \DTLifint{#1}%
6590   {\edef#2{\char#1\relax}}%
6591   {%
6592     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6593   }%
6594 }

```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`lsgetgrouplabel`

```

6595 \newcommand*{\glsgetgrouplabel}[1]{%
6596   \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6597   \ifthenelse{\equal{#1}{\glsnumbersgroupname}}{\glsnumbers}{#1}}

```

The command `\setentrycounter` sets the entry's associated counter (required by `\glshypernumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

`setentrycounter`

```

6598 \newcommand*{\setentrycounter}[2][ ]{%
6599   \def\@glo@counterprefix{#1}%
6600   \ifx\@glo@counterprefix\empty
6601     \def\@glo@counterprefix{.}%
6602   \else

```

```

6603 \def\@glo@counterprefix{.#1.}%
6604 \fi
6605 \def\glsentrycounter{#2}%
6606 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

6607 \newcommand*{\setglossarystyle}[1]{%
6608 \ifcsundef{@glsstyle@#1}%
6609 {%
6610 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6611 }%
6612 {%
6613 \csname @glsstyle@#1\endcsname
6614 }%

```

Set the default style if it's not already set.

```

6615 \ifx\@glossary@default@style\relax
6616 \protected@edef\@glossary@default@style{#1}%
6617 \fi
6618 }

```

`\glossarystyle`

```

6619 \newcommand*{\glossarystyle}[1]{%
6620 \ifcsundef{@glsstyle@#1}%
6621 {%
6622 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6623 }%
6624 {%
6625 \GlossariesWarning
6626 {Deprecated command \string\glossarystyle.^~J
6627 I recommend you switch to \string\setglossarystyle\space unless
6628 you want to maintain backward compatibility}%
6629 \setglossentrycompatibility
6630 \csname @glsstyle@#1\endcsname

6631 \ifcsdef{@glscompstyle@#1}%
6632 {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6633 }%
6634 }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6635 \ifx\@glossary@default@style\relax
6636 \protected@edef\@glossary@default@style{#1}%
6637 \fi
6638 }

```

`\newglossarystyle` New glossary styles can be defined using:

`\newglossarystyle{<name>}{<definition>}`

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [section 1.19](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```
6639 \newcommand{\newglossarystyle}[2]{%
6640   \ifcsundef{@glsstyle@#1}%
6641   {%
6642     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6643   }%
6644   {%
6645     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6646   }%
6647 }
```

`\ewglossarystyle` Code for this macro supplied by Marco Daniel.

```
6648 \newcommand{\renewglossarystyle}[2]{%
6649   \ifcsundef{@glsstyle@#1}%
6650   {%
6651     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6652   }%
6653   {%
6654     \csdef{@glsstyle@#1}{#2}%
6655   }%
6656 }
```

Glossary entries are encoded so that the second argument to `\glossaryentryfield` is always specified as `\glsnamefont{<name>}`. This allows the user to change the font used to display the name term without having to redefine `\glossaryentryfield`. The default uses the surrounding font, so in the list type styles (which place the name in the optional argument to `\item`) the name will appear in bold.

`\glsnamefont`

```
6657 \newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the format key in commands like `\glslink`. The default format is given by `\glslinknumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn’t have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for

the page counter, but this code needs to be more general. So I have adapted the code used in the package.

`\glshypernumber`

```
6658 \ifcsundef{hyperlink}%
6659 {%
6660   \def\glshypernumber#1{#1}%
6661 }%
6662 {%
6663   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6664 }
```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6665 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6666   \ifx\#1\%
6667   \else
6668     \@delimR#1\delimR\delimR\%
6669   \fi
6670   \ifx\#2\%
6671   \else
6672     #2%
6673   \fi
6674   \ifx\#3\%
6675   \else
6676     \@glshypernumber#3\@nil
6677   \fi
6678 }
```

`\@delimR` displays a range of numbers for the counter whose name is given by `\@gls@counter` (which must be set prior to using `\glshypernumber`).

`\@delimR`

```
6679 \def\@delimR#1\delimR #2\delimR #3\%
6680 \ifx\#2\%
6681   \@delimN{#1}%
6682 \else
6683   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6684 \fi}
```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```
6685 \def\@delimN#1{\@delimN#1\delimN \delimN\%
6686 \def\@delimN#1\delimN #2\delimN#3\%
6687 \ifx\#3\%
6688   \@gls@numberlink{#1}%
6689 \else
6690   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6691 \fi
6692 }
```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

6693 \def\@gls@numberlink#1{%
6694 \begingroup
6695 \toks@={}%
6696 \@gls@removespaces#1 \@nil
6697 \endgroup}

6698 \def\@gls@removespaces#1 #2\@nil{%
6699 \toks@=\expandafter{\the\toks@#1}%
6700 \ifx\#2\%
6701 \edef\x{\the\toks@}%
6702 \ifx\x\empty
6703 \else

6704 \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
6705 {\the\toks@}%
6706 \fi
6707 \else
6708 \@gls@ReturnAfterFi{%
6709 \@gls@removespaces#2\@nil
6710 }%
6711 \fi
6712 }
6713 \long\def\@gls@ReturnAfterFi#1\fi{\fi#1}

```

The following commands will switch to the appropriate font, and create a hyperlink, if hyperlinks are supported. If hyperlinks are not supported, they will just display their argument in the appropriate font.

\hyperrm

```
6714 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}
```

\hypersf

```
6715 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}
```

\hypertt

```
6716 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}
```

\hyperbf

```
6717 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}
```

\hypermd

```
6718 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}
```

\hyperit

```
6719 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}
```

\hypersl

```
6720 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6721 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6722 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6723 \newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}
```

1.17 Acronyms

`\oldacronym`

```
\oldacronym[⟨label⟩]{⟨abbrv⟩}{⟨long⟩}{⟨key-val list⟩}
```

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym [⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}` and it additionally defines the command `\⟨label⟩` which is equivalent to `\gls{⟨label⟩}` (thus `⟨label⟩` must only contain alphabetical characters). If `⟨label⟩` is omitted, `⟨abbrv⟩` is used. This only emulates the syntax of the old package. The way the acronyms appear in the list of acronyms is determined by the definition of `\newacronym` and the glossary style.

Note that `\⟨label⟩` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\⟨label⟩[⟨insert⟩]` but you can't do `\⟨label⟩[⟨key-val list⟩]`. For example if you define the acronym `svm`, then you can do `\svm['s]` but you can't do `\svm[format=textbf]`. If the package is loaded, `\svm['s]` will appear as `svm ['s]` which is unlikely to be the desired result. In this case, you will need to use `\gls` explicitly, e.g. `\gls{svm}['s]`. Note that it is up to the user to load if desired.

```
6724 \newcommand{\oldacronym}[4][\gls@label]{%
6725   \def\gls@label{#2}%
6726   \newacronym[#4]{#1}{#2}{#3}%
6727   \ifcsundef{xspace}%
6728     {%
6729       \expandafter\edef\csname#1\endcsname{%
6730         \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6731       }%
6732     }%
6733   {%
6734     \expandafter\edef\csname#1\endcsname{%
6735       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6736         \noexpand\gls{#1}\noexpand\xspace}%
6737     }%
6738   }%
6739 }
```

```
\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrv⟩}{⟨long⟩}
```

This is a quick way of defining acronyms, using `\newglossaryentry` with the appropriate

values. It sets the glossary type to `\acronymtype` which will be acronym if the package option `acronym` has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

This is just a stub. It's redefined by commands like `\SetDefaultAcronymStyle`.

`\newacronym`

```
6740 \newcommand{\newacronym}[4] [] {}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix`

Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, `ABCS` looks as though the “s” is part of the acronym, but `ABCS` looks as though the “s” is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is need to cancel it out.

```
6741 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6742 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6743 \newcommand*{\glsshortkey}{short}
```

`\shortpluralkey`

```
6744 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6745 \newcommand*{\glslongkey}{long}
```

`\longpluralkey`

```
6746 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6747 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}
```

```
6748 \newcommand*\ns@acrfull[2] [] {%
```

```
6749 \new@ifnextchar[{\@acrfull{#1}{#2}}%
```

```
6750 {\@acrfull{#1}{#2} []}%
```

```
6751 }
```

`\@acrfull` Low-level macro:

```
6752 \def\@acrfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6753 \acrfullfmt{#1}{#2}{#3}%  
6754 }
```

Using `\acrlinkfullformat` and `\acrfullformat` is now deprecated as it can cause complications with the first letter upper case variants, but the package needs to provide backward compatibility support.

`\acrfullfmt` No case change full format.

```
6755 \newcommand*{\acrfullfmt}[3]{%  
6756 \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%  
6757 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```
6758 \newcommand{\acrlinkfullformat}[5]{%  
6759 \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%  
6760 }
```

`\acrfullformat` Default full form is `<long>` (`<short>`).

```
6761 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
6762 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
6763 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}  
  
6764 \newcommand*{\ns@Acrfull}[2][]{%  
6765 \new@ifnextchar[{\@Acrfull{#1}{#2}}%  
6766 {\@Acrfull{#1}{#2}[]}%  
6767 }
```

Low-level macro:

```
6768 \def\@Acrfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6769 \Acrfullfmt{#1}{#2}{#3}%  
6770 }
```

`\Acrfullfmt` First letter upper case full format.

```
6771 \newcommand*{\Acrfullfmt}[3]{%  
6772 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%  
6773 }
```

`\ACRfull`

```
6774 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}
```

```

6775 \newcommand*\ns@ACRfull[2] [] {%
6776   \new@ifnextchar[{\@ACRfull{#1}{#2}}{%
6777     {\@ACRfull{#1}{#2} []}%
6778 }

```

Low-level macro:

```

6779 \def\@ACRfull#1#2[#3] {%

```

Make it easier for acronym styles to change this:

```

6780   \ACRfullfmt{#1}{#2}{#3}%
6781 }

```

`\ACRfullfmt` All upper case full format.

```

6782 \newcommand*\@ACRfullfmt[3] {%
6783   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
6784 }

```

Plural:

`\acrfullpl`

```

6785 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}

6786 \newcommand*\ns@acrfullpl[2] [] {%
6787   \new@ifnextchar[{\@acrfullpl{#1}{#2}}{%
6788     {\@acrfullpl{#1}{#2} []}%
6789 }

```

Low-level macro:

```

6790 \def\@acrfullpl#1#2[#3] {%

```

Make it easier for acronym styles to change this:

```

6791   \acrfullplfmt{#1}{#2}{#3}%
6792 }

```

`\acrfullplfmt` No case change plural full format.

```

6793 \newcommand*\@acrfullplfmt[3] {%
6794   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6795 }

```

`\Acrfullpl`

```

6796 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}

6797 \newcommand*\ns@Acrfullpl[2] [] {%
6798   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}{%
6799     {\@Acrfullpl{#1}{#2} []}%
6800 }

```

Low-level macro:

```

6801 \def\@Acrfullpl#1#2[#3] {%

```

Make it easier for acronym styles to change this:

```
6802 \Acrfullplfmt{#1}{#2}{#3}%  
6803 }
```

\Acrfullplfmt First letter upper case plural full format.

```
6804 \newcommand*{\Acrfullplfmt}[3]{%  
6805 \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6806 }
```

\ACRfullpl

```
6807 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\@ns@ACRfullpl}  
  
6808 \newcommand*\ns@ACRfullpl[2][{}]{%  
6809 \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%  
6810 {\@ACRfullpl{#1}{#2}[]}%  
6811 }
```

Low-level macro:

```
6812 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6813 \ACRfullplfmt{#1}{#2}{#3}%  
6814 }
```

\ACRfullplfmt All upper case plural full format.

```
6815 \newcommand*{\ACRfullplfmt}[3]{%  
6816 \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6817 }
```

1.18 Predefined acronym styles

\acronymfont This is only used with the additional acronym styles:

```
6818 \newcommand{\acronymfont}[1]{#1}
```

\firstacronymfont This is only used with the additional acronym styles:

```
6819 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}
```

\acrnameformat The styles that allow an additional description use \acrnameformat{<short>}{<long>} to determine what information is displayed in the name.

```
6820 \newcommand*{\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by \newacronym:

\glskeylisttok

```
6821 \newtoks\glskeylisttok
```

\glslabeltok

```
6822 \newtoks\glslabeltok
```

```

\glsshorttok
6823 \newtoks\glsshorttok

\glslongtok
6824 \newtoks\glslongtok

\newacronymhook  Provide a hook for \newacronym:
6825 \newcommand*{\newacronymhook}{}

\newacronymhook  New improved version of setting the acronym style.
6826 \newcommand*{\SetGenericNewAcronym}{%
    Change the behaviour of \Glsentryname to workaround expansion issues that cause a prob-
    lem for \makefirstuc
6827   \let\@Gls@entryname\@Gls@acrentryname
    Change the way acronyms are defined:
6828   \renewcommand{\newacronym}[4][]{%
6829     \ifdefempty{\@glsacronymlists}%
6830     {%
6831       \def\@glo@type{\acronymtype}%
6832       \setkeys{glossentry}{##1}%
6833       \DeclareAcronymList{\@glo@type}%
6834     }%
6835     {}%
6836     \glskeylisttok{##1}%
6837     \glslabeltok{##2}%
6838     \glsshorttok{##3}%
6839     \glslongtok{##4}%
6840     \newacronymhook
6841     \protected@edef\@do@newglossaryentry{%
6842       \noexpand\newglossaryentry{\the\glslabeltok}%
6843       {%
6844         type=\acronymtype,%
6845         name={\expandonce{\acronymentry{##2}}},%
6846         sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6847         text={\the\glsshorttok},%
6848         short={\the\glsshorttok},%
6849         shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6850         long={\the\glslongtok},%
6851         longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6852         \GenericAcronymFields,%
6853         \the\glskeylisttok
6854       }%
6855     }%
6856     \@do@newglossaryentry
6857   }%
    Make sure that \acrfull etc reflects the new style:
6858   \renewcommand{\acrfullfmt}[3]{%

```

```

6859 \glslink{##1}{##2}{\genacrfullformat{##2}{##3}}}%
6860 \renewcommand*{\Acrfullfmt}[3]{%
6861 \glslink{##1}{##2}{\Genacrfullformat{##2}{##3}}}%
6862 \renewcommand*{\ACRfullfmt}[3]{%
6863 \glslink{##1}{##2}{%
6864 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6865 \renewcommand*{\acrfullplfmt}[3]{%
6866 \glslink{##1}{##2}{\genplacrfullformat{##2}{##3}}}%
6867 \renewcommand*{\Acrfullplfmt}[3]{%
6868 \glslink{##1}{##2}{\Genplacrfullformat{##2}{##3}}}%
6869 \renewcommand*{\ACRfullplfmt}[3]{%
6870 \glslink{##1}{##2}{%
6871 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that `\glsentryfull` etc reflects the new style:

```

6872 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6873 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6874 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6875 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6876 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6877 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry`

```
\acronymentry{<label>}
```

Display style for the name field in the list of acronyms.

```

6878 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort`

```
\acronymsort{<short>}{<long>}
```

Default sort format for acronyms.

```

6879 \newcommand*{\acronymsort}[2]{#1}

```

`\setacronymstyle`

```
\setacronymstyle{<style name>}
```

```

6880 \newcommand*{\setacronymstyle}[1]{%
6881 \ifcsundef{@glsacr@dispstyle@#1}
6882 {%
6883 \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
6884 }%
6885 {%
6886 \ifdefempty{\@glsacronymlists}%
6887 {%
6888 \DeclareAcronymList{\acronymtype}%

```

```

6889 }%
6890 {}%
6891 \SetGenericNewAcronym
6892 \GlsUseAcrStyleDefs{#1}%
6893 \@for\@gls@type:=\@glsacronymlists\do{%
6894     \defglentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6895 }%
6896 }%
6897 }

```

`\newacronymstyle`

```

\newacronymstyle{<style name>}{<entry format definition>}{<display
definitions>}

```

Defines a new acronym style called *<style name>*.

```

6898 \newcommand*{\newacronymstyle}[3]{%
6899     \ifcsdef{@glsacr@dispstyle@#1}%
6900     {%
6901         \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6902     }%
6903     {%
6904         \csdef{@glsacr@dispstyle@#1}{#2}%
6905         \csdef{@glsacr@styledefs@#1}{#3}%
6906     }%
6907 }

```

`\renewacronymstyle` Redefines the given acronym style.

```

6908 \newcommand*{\renewacronymstyle}[3]{%
6909     \ifcsdef{@glsacr@dispstyle@#1}%
6910     {%
6911         \csdef{@glsacr@dispstyle@#1}{#2}%
6912         \csdef{@glsacr@styledefs@#1}{#3}%
6913     }%
6914     {%
6915         \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6916     }%
6917 }

```

`\rEntryDispStyle`

```

6918 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

`\UseAcrStyleDefs`

```

6919 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

`long-short` *<long>* (*<short>*) acronym style.

```

6920 \newacronymstyle{long-short}%
6921 {%

```

Check for long form in case this is a mixed glossary.

```

6922 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6923 }%
6924 {%
6925 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6926 \renewcommand*{\genacrfullformat}[2]{%
6927 \glentrylong{##1}##2\space
6928 (\protect\firstacronymfont{\glentryshort{##1}})%
6929 }%
6930 \renewcommand*{\Genacrfullformat}[2]{%
6931 \Glentrylong{##1}##2\space
6932 (\protect\firstacronymfont{\glentryshort{##1}})%
6933 }%
6934 \renewcommand*{\genplacrfullformat}[2]{%
6935 \glentrylongpl{##1}##2\space
6936 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6937 }%
6938 \renewcommand*{\Genplacrfullformat}[2]{%
6939 \Glentrylongpl{##1}##2\space
6940 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6941 }%
6942 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6943 \renewcommand*{\acronymsort}[2]{##1}%
6944 \renewcommand*{\acronymfont}[1]{##1}%
6945 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6946 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6947 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6948 \newacronymstyle{long-sp-short}%
6949 {%

```

Check for long form in case this is a mixed glossary.

```

6950 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6951 }%
6952 {%
6953 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6954 \renewcommand*{\genacrfullformat}[2]{%
6955 \glentrylong{##1}##2\glspacespace{##1}%
6956 (\protect\firstacronymfont{\glentryshort{##1}})%
6957 }%
6958 \renewcommand*{\Genacrfullformat}[2]{%
6959 \Glentrylong{##1}##2\glspacespace{##1}%
6960 (\protect\firstacronymfont{\glentryshort{##1}})%
6961 }%
6962 \renewcommand*{\genplacrfullformat}[2]{%
6963 \glentrylongpl{##1}##2\glspacespace{##1}%
6964 (\protect\firstacronymfont{\glentryshortpl{##1}})%

```



```

6965 }%
6966 \renewcommand*{\Genplacrfullformat}[2]{%
6967   \Glsentrylongpl{##1}##2\glsacspace{##1}%
6968   (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6969 }%
6970 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6971 \renewcommand*{\acronymsort}[2]{##1}%
6972 \renewcommand*{\acronymfont}[1]{##1}%
6973 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6974 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6975 }

```

`\glsacspace` Space between long and short form for the above style. This uses a non-breakable space if the short form is less than 3em, otherwise it uses a regular space.

```

6976 \newcommand*{\glsacspace}[1]{%
6977   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{##1}})}%
6978   \ifdim\dimen@<3em~\else\space\fi
6979 }

```

`short-long` (*short*) (*long*) acronym style.

```

6980 \newacronymstyle{short-long}%
6981 {%
  Check for long form in case this is a mixed glossary.
6982   \ifglshaslong{\glslabel}{\glsacacfmt}{\glsacentryfmt}%
6983 }%
6984 {%
6985   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6986   \renewcommand*{\genacrfullformat}[2]{%
6987     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6988     (\glsentrylong{##1})%
6989   }%
6990   \renewcommand*{\Genacrfullformat}[2]{%
6991     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6992     (\glsentrylong{##1})%
6993   }%
6994   \renewcommand*{\genplacrfullformat}[2]{%
6995     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6996     (\glsentrylongpl{##1})%
6997   }%
6998   \renewcommand*{\Genplacrfullformat}[2]{%
6999     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
7000     (\glsentrylongpl{##1})%
7001   }%
7002   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
7003   \renewcommand*{\acronymsort}[2]{##1}%
7004   \renewcommand*{\acronymfont}[1]{##1}%
7005   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
7006   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

7007 }

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```
7008 \newacronymstyle{long-sc-short}%
7009 {%
7010   \GlsUseAcrEntryDisplayStyle{long-short}%
7011 }%
7012 {%
7013   \GlsUseAcrStyleDefs{long-short}%
7014   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
7015   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7016 }
```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```
7017 \newacronymstyle{long-sm-short}%
7018 {%
7019   \GlsUseAcrEntryDisplayStyle{long-short}%
7020 }%
7021 {%
7022   \GlsUseAcrStyleDefs{long-short}%
7023   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
7024   \renewcommand*{\acrpluralsuffix}{\glssacrpluralsuffix}%
7025 }
```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```
7026 \newacronymstyle{sc-short-long}%
7027 {%
7028   \GlsUseAcrEntryDisplayStyle{short-long}%
7029 }%
7030 {%
7031   \GlsUseAcrStyleDefs{short-long}%
7032   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
7033   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7034 }
```

sm-short-long *<short>* (\textsmaller{<long>}) acronym style.

```
7035 \newacronymstyle{sm-short-long}%
7036 {%
7037   \GlsUseAcrEntryDisplayStyle{short-long}%
7038 }%
7039 {%
7040   \GlsUseAcrStyleDefs{short-long}%
7041   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
7042   \renewcommand*{\acrpluralsuffix}{\glssacrpluralsuffix}%
7043 }
```

long-short-desc *<long>* ({<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

7044 \newacronymstyle{long-short-desc}%
7045 {%
7046   \GlsUseAcrEntryDisplayStyle{long-short}%
7047 }%
7048 {%
7049   \GlsUseAcrStyleDefs{long-short}%
7050   \renewcommand*{\GenericAcronymFields}{}%
7051   \renewcommand*{\acronymsort}[2]{##2}%
7052   \renewcommand*{\acronymentry}[1]{%
7053     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7054 }

```

g-sp-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply). The space between the long and short form is given by \glsacspace.

```

7055 \newacronymstyle{long-sp-short-desc}%
7056 {%
7057   \GlsUseAcrEntryDisplayStyle{long-sp-short}%
7058 }%
7059 {%
7060   \GlsUseAcrStyleDefs{long-sp-short}%
7061   \renewcommand*{\GenericAcronymFields}{}%
7062   \renewcommand*{\acronymsort}[2]{##2}%
7063   \renewcommand*{\acronymentry}[1]{%
7064     \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
7065 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

7066 \newacronymstyle{long-sc-short-desc}%
7067 {%
7068   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
7069 }%
7070 {%
7071   \GlsUseAcrStyleDefs{long-sc-short}%
7072   \renewcommand*{\GenericAcronymFields}{}%
7073   \renewcommand*{\acronymsort}[2]{##2}%
7074   \renewcommand*{\acronymentry}[1]{%
7075     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7076 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

7077 \newacronymstyle{long-sm-short-desc}%
7078 {%
7079   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
7080 }%
7081 {%
7082   \GlsUseAcrStyleDefs{long-sm-short}%
7083   \renewcommand*{\GenericAcronymFields}{}%

```

```

7084 \renewcommand*{\acronymsort}[2]{##2}%
7085 \renewcommand*{\acronymentry}[1]{%
7086     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7087 }

```

short-long-desc *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

7088 \newacronymstyle{short-long-desc}%
7089 {%
7090     \GlsUseAcrEntryDisplayStyle{short-long}%
7091 }%
7092 {%
7093     \GlsUseAcrStyleDefs{short-long}%
7094     \renewcommand*{\GenericAcronymFields}{}%
7095     \renewcommand*{\acronymsort}[2]{##2}%
7096     \renewcommand*{\acronymentry}[1]{%
7097         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7098 }

```

short-long-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

7099 \newacronymstyle{sc-short-long-desc}%
7100 {%
7101     \GlsUseAcrEntryDisplayStyle{sc-short-long}%
7102 }%
7103 {%
7104     \GlsUseAcrStyleDefs{sc-short-long}%
7105     \renewcommand*{\GenericAcronymFields}{}%
7106     \renewcommand*{\acronymsort}[2]{##2}%
7107     \renewcommand*{\acronymentry}[1]{%
7108         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7109 }

```

short-long-desc *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

7110 \newacronymstyle{sm-short-long-desc}%
7111 {%
7112     \GlsUseAcrEntryDisplayStyle{sm-short-long}%
7113 }%
7114 {%
7115     \GlsUseAcrStyleDefs{sm-short-long}%
7116     \renewcommand*{\GenericAcronymFields}{}%
7117     \renewcommand*{\acronymsort}[2]{##2}%
7118     \renewcommand*{\acronymentry}[1]{%
7119         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7120 }

```

dua *<long>* only acronym style.

```

7121 \newacronymstyle{dua}%
7122 {%

```

Check for long form in case this is a mixed glossary.

```

7123 \ifdefempty\glscustomtext
7124 {%
7125 \ifglshaslong{\glslabel}%
7126 {%
7127 \glsifplural
7128 {%

```

Plural form:

```

7129 \glscapscase
7130 {%

```

Plural form, don't adjust case:

```

7131 \glentrylongpl{\glslabel}\glsinsert
7132 }%
7133 {%

```

Plural form, make first letter upper case:

```

7134 \Glentrylongpl{\glslabel}\glsinsert
7135 }%
7136 {%

```

Plural form, all caps:

```

7137 \mfirstucMakeUppercase
7138 {\glentrylongpl{\glslabel}\glsinsert}%
7139 }%
7140 }%
7141 {%

```

Singular form

```

7142 \glscapscase
7143 {%

```

Singular form, don't adjust case:

```

7144 \glentrylong{\glslabel}\glsinsert
7145 }%
7146 {%

```

Subsequent singular form, make first letter upper case:

```

7147 \Glentrylong{\glslabel}\glsinsert
7148 }%
7149 {%

```

Subsequent singular form, all caps:

```

7150 \mfirstucMakeUppercase
7151 {\glentrylong{\glslabel}\glsinsert}%
7152 }%
7153 }%
7154 }%
7155 {%

```

Not an acronym:

```
7156      \glsgenentryfmt
7157      }%
7158      }%
7159      {\glscustomtext\glsinsert}%
7160 }%
7161 {%
7162   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

7163   \renewcommand*{\acrfullfmt}[3]{%
7164     \glslink{##1}{##2}{\glsentrylong{##2}##3\space
7165       (\acronymfont{\glsentryshort{##2}})}}%
7166   \renewcommand*{\Acrfullfmt}[3]{%
7167     \glslink{##1}{##2}{\Glsentrylong{##2}##3\space
7168       (\acronymfont{\glsentryshort{##2}})}}%
7169   \renewcommand*{\ACRfullfmt}[3]{%
7170     \glslink{##1}{##2}{%
7171       \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
7172         (\acronymfont{\glsentryshort{##2}})}}}%

7173   \renewcommand*{\acrfullplfmt}[3]{%
7174     \glslink{##1}{##2}{\glsentrylongpl{##2}##3\space
7175       (\acronymfont{\glsentryshortpl{##2}})}}%

7176   \renewcommand*{\Acrfullplfmt}[3]{%
7177     \glslink{##1}{##2}{\Glsentrylongpl{##2}##3\space
7178       (\acronymfont{\glsentryshortpl{##2}})}}%
7179   \renewcommand*{\ACRfullplfmt}[3]{%
7180     \glslink{##1}{##2}{%
7181       \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
7182         (\acronymfont{\glsentryshortpl{##2}})}}}%
7183   \renewcommand*{\glsentryfull}[1]{%
7184     \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
7185   }%
7186   \renewcommand*{\Glsentryfull}[1]{%
7187     \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
7188   }%
7189   \renewcommand*{\glsentryfullpl}[1]{%
7190     \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
7191   }%
7192   \renewcommand*{\Glsentryfullpl}[1]{%
7193     \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
7194   }%
7195   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
7196   \renewcommand*{\acronymsort}[2]{##1}%
7197   \renewcommand*{\acronymfont}[1]{##1}%
7198   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7199 }
```

dua-desc <long> only acronym style with user-supplied description.

```
7200 \newacronymstyle{dua-desc}%
7201 {%
7202   \GlsUseAcrEntryDispStyle{dua}%
7203 }%
7204 {%
7205   \GlsUseAcrStyleDefs{dua}%
7206   \renewcommand*{\GenericAcronymFields}{}%

7207   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
7208   \renewcommand*{\acronymsort}[2]{##2}%
7209 }%
```

footnote <short>\footnote{<long>} acronym style.

```
7210 \newacronymstyle{footnote}%
7211 {%

  Check for long form in case this is a mixed glossary.

7212   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
7213 }%
7214 {%
7215   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
7216 \glshyperfirstfalse
7217 \renewcommand*{\genacrfullformat}[2]{%
7218   \protect\firstacronymfont{\glentryshort{##1}}##2%
7219   \protect\footnote{\glentrylong{##1}}%
7220 }%
7221 \renewcommand*{\Genacrfullformat}[2]{%
7222   \firstacronymfont{\Glentryshort{##1}}##2%
7223   \protect\footnote{\glentrylong{##1}}%
7224 }%
7225 \renewcommand*{\genplacrfullformat}[2]{%
7226   \protect\firstacronymfont{\glentryshortpl{##1}}##2%
7227   \protect\footnote{\glentrylongpl{##1}}%
7228 }%
7229 \renewcommand*{\Genplacrfullformat}[2]{%
7230   \protect\firstacronymfont{\Glentryshortpl{##1}}##2%
7231   \protect\footnote{\glentrylongpl{##1}}%
7232 }%
7233 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
7234 \renewcommand*{\acronymsort}[2]{##1}%
7235 \renewcommand*{\acronymfont}[1]{##1}%
7236 \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
```

Don't use footnotes for \acrfull:

```
7237 \renewcommand*{\acrfullfmt}[3]{%
7238   \glslink[##1]{##2}{\acronymfont{\glentryshort{##2}}##3\space
7239   (\glentrylong{##2})}%
```

```

7240 \renewcommand*{\Acrfullfmt}[3]{%
7241   \glslink{##1}{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
7242   (\Glsentrylong{##2})}%
7243 \renewcommand*{\ACRfullfmt}[3]{%
7244   \glslink{##1}{##2}{%
7245     \mfirstucMakeUppercase{\acronymfont{\Glsentryshort{##2}}##3\space
7246     (\Glsentrylong{##2})}}}%
7247 \renewcommand*{\acrfullplfmt}[3]{%
7248   \glslink{##1}{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
7249   (\Glsentrylongpl{##2})}%
7250 \renewcommand*{\Acrfullplfmt}[3]{%
7251   \glslink{##1}{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
7252   (\Glsentrylongpl{##2})}%
7253 \renewcommand*{\ACRfullplfmt}[3]{%
7254   \glslink{##1}{##2}{%
7255     \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{##2}}##3\space
7256     (\Glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

7257 \renewcommand*{\glsentryfull}[1]{%
7258   \acronymfont{\Glsentryshort{##1}}\space(\Glsentrylong{##1})}%
7259 \renewcommand*{\Glsentryfull}[1]{%
7260   \acronymfont{\Glsentryshort{##1}}\space(\Glsentrylong{##1})}%
7261 \renewcommand*{\glsentryfullpl}[1]{%
7262   \acronymfont{\Glsentryshortpl{##1}}\space(\Glsentrylongpl{##1})}%
7263 \renewcommand*{\Glsentryfullpl}[1]{%
7264   \acronymfont{\Glsentryshortpl{##1}}\space(\Glsentrylongpl{##1})}%
7265 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

7266 \newacronymstyle{footnote-sc}%
7267 {%
7268   \GlsUseAcrEntryDisplayStyle{footnote}%
7269 }%
7270 {%
7271   \GlsUseAcrStyleDefs{footnote}%
7272   \renewcommand{\acronymentry}[1]{\acronymfont{\Glsentryshort{##1}}}
7273   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
7274   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7275 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

7276 \newacronymstyle{footnote-sm}%
7277 {%
7278   \GlsUseAcrEntryDisplayStyle{footnote}%
7279 }%
7280 {%
7281   \GlsUseAcrStyleDefs{footnote}%
7282   \renewcommand{\acronymentry}[1]{\acronymfont{\Glsentryshort{##1}}}
7283   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%

```



```

7284 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7285 }%

```

footnote-desc *<short>*\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

7286 \newacronymstyle{footnote-desc}%
7287 {%
7288 \GlsUseAcrEntryDisplayStyle{footnote}%
7289 }%
7290 {%
7291 \GlsUseAcrStyleDefs{footnote}%
7292 \renewcommand*{\GenericAcronymFields}{}%
7293 \renewcommand*{\acronymsort}[2]{##2}%
7294 \renewcommand*{\acronymentry}[1]{%
7295 \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
7296 }

```

footnote-sc-desc \textsc{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

7297 \newacronymstyle{footnote-sc-desc}%
7298 {%
7299 \GlsUseAcrEntryDisplayStyle{footnote-sc}%
7300 }%
7301 {%
7302 \GlsUseAcrStyleDefs{footnote-sc}%
7303 \renewcommand*{\GenericAcronymFields}{}%
7304 \renewcommand*{\acronymsort}[2]{##2}%
7305 \renewcommand*{\acronymentry}[1]{%
7306 \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
7307 }

```

footnote-sm-desc \textsmaller{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

7308 \newacronymstyle{footnote-sm-desc}%
7309 {%
7310 \GlsUseAcrEntryDisplayStyle{footnote-sm}%
7311 }%
7312 {%
7313 \GlsUseAcrStyleDefs{footnote-sm}%
7314 \renewcommand*{\GenericAcronymFields}{}%
7315 \renewcommand*{\acronymsort}[2]{##2}%
7316 \renewcommand*{\acronymentry}[1]{%
7317 \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
7318 }

```

AcronymSynonyms

```

7319 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

`\acs`
 7320 `\let\acs\acrshort`
 First letter uppercase short form

`\Acs`
 7321 `\let\Acs\Acrshort`
 Plural short form

`\acsp`
 7322 `\let\acsp\acrshortpl`
 First letter uppercase plural short form

`\Acsp`
 7323 `\let\Acsp\Acrshortpl`
 Long form

`\acl`
 7324 `\let\acl\acrlong`
 Plural long form

`\aclp`
 7325 `\let\aclp\acrlongpl`
 First letter upper case long form

`\Acl`
 7326 `\let\Acl\Acrlong`
 First letter upper case plural long form

`\Aclp`
 7327 `\let\Aclp\Acrlongpl`
 Full form

`\acf`
 7328 `\let\acf\acrfull`
 Plural full form

`\acfp`
 7329 `\let\acfp\acrfullpl`
 First letter upper case full form

`\Acf`
 7330 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

```
7331 \let\Acfp\Acrfullpl
```

Standard form

`\ac`

```
7332 \let\ac\gls
```

First upper case standard form

`\Ac`

```
7333 \let\Ac\Gls
```

Standard plural form

`\acp`

```
7334 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
7335 \let\Acp\Glspl
```

```
7336 }
```

Define synonyms if required

```
7337 \ifglsacrshortcuts
```

```
7338 \DefineAcronymSynonyms
```

```
7339 \fi
```

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\glsDisplayStyle` Sets the default acronym display style for given glossary.

```
7340 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
```

```
7341 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
7342 }
```

`\glsNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\gslongtok` and `\glskeylisttok`.

```
7343 \newcommand*{\DefaultNewAcronymDef}{%
```

```
7344 \edef\@do@newglossaryentry{%
```

```
7345 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
7346 {%
```

```
7347 type=\acronymtype,%
```

```
7348 name={\the\glsshorttok},%
```

```
7349 sort={\the\glsshorttok},%
```

```
7350 text={\the\glsshorttok},%
```

```
7351 first={\acrfullformat{\the\gslongtok}{\the\glsshorttok}},%
```

```
7352 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
```

```

7353     firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7354                               {\noexpand\expandonce\noexpand\@glo@shortpl}},%
7355     short={\the\glsshorttok},%
7356     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7357     long={\the\glslongtok},%
7358     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7359     description={\the\glslongtok},%
7360     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

Remaining options specified by the user:

```

7361     \the\glskeylisttok
7362   }%
7363 }%
7364 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7365 \let\@org@gls@assign@plural\gls@assign@plural
7366 \let\@org@gls@assign@descplural\gls@assign@descplural
7367 \def\gls@assign@firstpl##1##2{%
7368   \@gls@expand@field{##1}{firstpl}{##2}%
7369 }%
7370 \def\gls@assign@plural##1##2{%
7371   \@gls@expand@field{##1}{plural}{##2}%
7372 }%
7373 \def\gls@assign@descplural##1##2{%
7374   \@gls@expand@field{##1}{descplural}{##2}%
7375 }%
7376 \@do@newglossaryentry
7377 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7378 \let\gls@assign@plural\@org@gls@assign@plural
7379 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7380 }

```

ultAcronymStyle Set up the default acronym style:

```

7381 \newcommand*{\SetDefaultAcronymStyle}{%

```

Set the display style:

```

7382   \@for\@gls@type:=\@gls@acronymlists\do{%
7383     \SetDefaultAcronymDisplayStyle{\@gls@type}%
7384   }%

```

Set up the definition of `\newacronym`:

```

7385   \renewcommand{\newacronym}[4][{}]{%

```

If user is just using the main glossary and hasn't identified it as a list of acronyms, then update.

(This is done to ensure backwards compatibility with versions prior to 2.04).

```

7386     \ifx\@gls@acronymlists\@empty
7387       \def\@glo@type{\acronymtype}%
7388       \setkeys{glossentry}{##1}%
7389       \DeclareAcronymList{\@glo@type}%
7390       \SetDefaultAcronymDisplayStyle{\@glo@type}%
7391     \fi
7392     \glskeylisttok{##1}%

```

```

7393 \glslabeltok{##2}%
7394 \glsshorttok{##3}%
7395 \glslongtok{##4}%
7396 \newacronymhook
7397 \DefaultNewAcronymDef
7398 }%
7399 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7400 }

```

`\acrfootnote` Used by the footnote acronym styles.

```

7401 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}

```

`acrlinkfootnote`

```

7402 \newcommand*{\acrlinkfootnote}[3]{%
7403 \footnote{\glslink[#1]{#2}{#3}}%
7404 }

```

`crnolinkfootnote`

```

7405 \newcommand*{\acrnolinkfootnote}[3]{%
7406 \footnote{#3}%
7407 }

```

`acronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```

7408 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7409 \defglsentryfmt[#1]{%
7410 \ifdefempty\glscustomtext
7411 {%
7412 \ifglsused{\glslabel}%
7413 {%
7414 \acronymfont{\glsgenentryfmt}%
7415 }%
7416 {%
7417 \firstacronymfont{\glsgenentryfmt}%
7418 \ifgls hassymbol{\glslabel}%
7419 {%
7420 \expandafter\protect\expandafter\acrfootnote\expandafter
7421 {\@gls@link@opts}{\@gls@link@label}%
7422 {%
7423 \glsifplural
7424 {\glsentrysymbolplural{\glslabel}}%
7425 {\glsentrysymbol{\glslabel}}%
7426 }%
7427 }%
7428 }%
7429 }%
7430 {\glscustomtext\glsinsert}%
7431 }%
7432 }

```

teNewAcronymDef

```

7433 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7434   \edef\@do@newglossaryentry{%
7435     \noexpand\newglossaryentry{\the\glslabeltok}%
7436     {%
7437       type=\acronymtype,%
7438       name={\noexpand\acronymfont{\the\glsshorttok}},%
7439       sort={\the\glsshorttok},%
7440       first={\the\glsshorttok},%
7441       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7442       text={\the\glsshorttok},%
7443       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7444       short={\the\glsshorttok},%
7445       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7446       long={\the\glslongtok},%
7447       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7448       symbol={\the\glslongtok},%
7449       symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7450       \the\glskeylisttok
7451     }%
7452   }%
7453   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7454   \let\@org@gls@assign@plural\gls@assign@plural
7455   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7456   \def\gls@assign@firstpl##1##2{%
7457     \@@gls@expand@field{##1}{firstpl}{##2}%
7458   }%
7459   \def\gls@assign@plural##1##2{%
7460     \@@gls@expand@field{##1}{plural}{##2}%
7461   }%
7462   \def\gls@assign@symbolplural##1##2{%
7463     \@@gls@expand@field{##1}{symbolplural}{##2}%
7464   }%
7465   \@do@newglossaryentry
7466   \let\gls@assign@plural\@org@gls@assign@plural
7467   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7468   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7469 }

```

oteAcronymStyle

If a description and footnote are both required, store the long form in the symbol key. Store the short form in text key. Note that since the long form is stored in the symbol key, if you want the long form to appear in the list of acronyms, you need to use a glossary style that displays the symbol key.

```

7470 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7471   \renewcommand{\newacronym}[4][\]{%
7472     \ifx\@glsacronymlists\@empty
7473       \def\@glo@type{\acronymtype}%
7474       \setkeys{glossentry}{##1}%
7475       \DeclareAcronymList{\@glo@type}%

```

```

7476     \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7477     \fi
7478     \glskeylisttok{##1}%
7479     \glslabeltok{##2}%
7480     \glsshorttok{##3}%
7481     \glslongtok{##4}%
7482     \newacronymhook
7483     \DescriptionFootnoteNewAcronymDef
7484 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7485 \@for\@gls@type:=\@glsacronymlists\do{%
7486     \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7487 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7488 \ifglsacrsmallcaps
7489     \renewcommand*\acronymfont}[1]{\textsc{##1}}%
7490     \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7491 \else
7492     \ifglsacrsmaller
7493         \renewcommand*\acronymfont}[1]{\textsmaller{##1}}%
7494     \fi
7495 \fi

```

Check for package option clash

```

7496 \ifglsacrdua
7497     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7498     can’t both be set}{}%
7499 \fi
7500 }%

```

`nymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.

```

7501 \newcommand*\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7502     \defglsentryfmt[#1]{\glsngenentryfmt}%
7503 }

```

`UANewAcronymDef`

```

7504 \newcommand*\DescriptionDUANewAcronymDef{%
7505     \edef\@do@newglossaryentry{%
7506         \noexpand\newglossaryentry{\the\glslabeltok}%
7507         {%
7508             type=\acronymtype,%
7509             name={\the\glslongtok},%
7510             sort={\the\glslongtok},%
7511             text={\the\glslongtok},%
7512             first={\the\glslongtok},%

```

```

7513 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7514 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7515 short={\the\glsshorttok},%
7516 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7517 long={\the\glslongtok},%
7518 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7519 symbol={\the\glsshorttok},%
7520 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7521 \the\glskeylisttok
7522 }%
7523 }%
7524 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7525 \let\@org@gls@assign@plural\gls@assign@plural
7526 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7527 \def\gls@assign@firstpl##1##2{%
7528   \@gls@expand@field{##1}{firstpl}{##2}%
7529 }%
7530 \def\gls@assign@plural##1##2{%
7531   \@gls@expand@field{##1}{plural}{##2}%
7532 }%
7533 \def\gls@assign@symbolplural##1##2{%
7534   \@gls@expand@field{##1}{symbolplural}{##2}%
7535 }%
7536 \do@newglossaryentry
7537 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7538 \let\gls@assign@plural\@org@gls@assign@plural
7539 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7540 }

```

DUAACronymStyle Description, don't use acronym and no footnote. Note that the short form is stored in the symbol key, so if the short form needs to be displayed in the glossary, use a style the displays the symbol.

```

7541 \newcommand*{\SetDescriptionDUAACronymStyle}{%
7542   \ifglsacrsmallcaps
7543     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7544       can't both be set}{}%
7545   \else
7546     \ifglsacrsmaller
7547       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7548         can't both be set}{}%
7549     \fi
7550   \fi
7551   \renewcommand{\newacronym}[4][{}]{%
7552     \ifx\@glsacronymlists\@empty
7553       \def\@glo@type{\acronymtype}%
7554       \setkeys{glossentry}{##1}%
7555       \DeclareAcronymList{\@glo@type}%
7556       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
7557     \fi

```



```

7558 \glskeylisttok{##1}%
7559 \glslabeltok{##2}%
7560 \glsshorttok{##3}%
7561 \glslongtok{##4}%
7562 \newacronymhook
7563 \DescriptionDUANewAcronymDef
7564 }%

```

Set display.

```

7565 \@for\@gls@type:=\@glsacronymlists\do{%
7566 \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
7567 }%
7568 }%

```

nymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7569 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7570 \defglsentryfmt[#1]{%

7571 \ifdefempty\glscustomtext
7572 {%
7573 \ifglsused{\glslabel}%
7574 {%

```

Move the inserted text outside of \acronymfont

```

7575 \let\gls@org@insert\glsinsert
7576 \let\glsinsert\@empty
7577 \acronymfont{\glsentryfmt}\gls@org@insert
7578 }%
7579 {%
7580 \glsentryfmt
7581 \ifgls hassymbol{\glslabel}%
7582 {%
7583 \glsifplural
7584 {%
7585 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7586 }%
7587 {%
7588 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7589 }%
7590 \space(\protect\firstacronymfont
7591 {\glsupcase
7592 {\@glo@symbol}
7593 {\@glo@symbol}
7594 {\mfirstucMakeUppercase{\@glo@symbol}}})%
7595 }%
7596 {}%
7597 }%
7598 }%
7599 {\glscustomtext\glsinsert}%

```

```

7600 }%
7601 }

```

onNewAcronymDef

```

7602 \newcommand*{\DescriptionNewAcronymDef}{%
7603   \edef\@do@newglossaryentry{%
7604     \noexpand\newglossaryentry{\the\glslabeltok}%
7605     {%
7606       type=\acronymtype,%
7607       name={\noexpand
7608         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
7609       sort={\the\glsshorttok},%
7610       first={\the\glslongtok},%
7611       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7612       text={\the\glsshorttok},%
7613       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7614       short={\the\glsshorttok},%
7615       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7616       long={\the\glslongtok},%
7617       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7618       symbol={\noexpand\@glo@text},%
7619       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7620       \the\glskeylisttok}%
7621   }%
7622   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7623   \let\@org@gls@assign@plural\gls@assign@plural
7624   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7625   \def\gls@assign@firstpl##1##2{%
7626     \@@gls@expand@field{##1}{firstpl}{##2}%
7627   }%
7628   \def\gls@assign@plural##1##2{%
7629     \@@gls@expand@field{##1}{plural}{##2}%
7630   }%
7631   \def\gls@assign@symbolplural##1##2{%
7632     \@@gls@expand@field{##1}{symbolplural}{##2}%
7633   }%
7634   \@do@newglossaryentry
7635   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7636   \let\gls@assign@plural\@org@gls@assign@plural
7637   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7638 }

```

ionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using \acrnameformat to allow the user to override the way the name is displayed in the list of acronyms.

```

7639 \newcommand*{\SetDescriptionAcronymStyle}{%
7640   \renewcommand{\newacronym}[4][ ]{%
7641     \ifx\@glsacronymlists\empty
7642       \def\@glo@type{\acronymtype}%

```

```

7643     \setkeys{glossentry}{##1}%
7644     \DeclareAcronymList{\@glo@type}%
7645     \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7646     \fi
7647     \glskeylisttok{##1}%
7648     \glslabeltok{##2}%
7649     \glsshorttok{##3}%
7650     \glslongtok{##4}%
7651     \newacronymhook
7652     \DescriptionNewAcronymDef
7653 }%

```

Set display.

```

7654 \@for\@gls@type:=\@glsacronymlists\do{%
7655     \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7656 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7657 \ifglsacrsmallcaps
7658     \renewcommand{\acronymfont}[1]{\textsc{##1}}
7659     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7660 \else
7661     \ifglsacrsmaller
7662         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7663     \fi
7664 \fi
7665 }%

```

`\acronymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7666 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7667     \defglsentryfmt[#1]{%
7668         \ifdefempty\glscustomtext
7669             {%

```

Move the inserted text outside of `\acronymfont`

```

7670         \let\gls@org@insert\glsinsert
7671         \let\glsinsert\@empty
7672         \ifglsused{\glslabel}%
7673         {%
7674             \acronymfont{\glsentryfmt}\gls@org@insert
7675         }%
7676         {%
7677             \firstacronymfont{\glsentryfmt}\gls@org@insert
7678             \ifglsashaslong{\glslabel}%
7679             {%
7680                 \expandafter\protect\expandafter\acrfootnote\expandafter
7681                 {\@gls@link@opts}{\@gls@link@label}%

```

```

7682         {%
7683         \glsifplural
7684         {\glsentrylongpl{\glslabel}}}%
7685         {\glsentrylong{\glslabel}}}%
7686     }%
7687 }%

7688     {%
7689     }%
7690 }%
7691 {\glscustomtext\glsinsert}%
7692 }%
7693 }

```

teNewAcronymDef

```

7694 \newcommand*{\FootnoteNewAcronymDef}{%
7695 \edef\@do@newglossaryentry{%
7696 \noexpand\newglossaryentry{\the\glslabeltok}%
7697 {%
7698     type=\acronymtype,%
7699     name={\noexpand\acronymfont{\the\glsshorttok}},%
7700     sort={\the\glsshorttok},%
7701     text={\the\glsshorttok},%
7702     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7703     first={\the\glsshorttok},%
7704     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7705     short={\the\glsshorttok},%
7706     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7707     long={\the\glslongtok},%
7708     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7709     description={\the\glslongtok},%
7710     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7711     \the\glskeylisttok
7712 }%
7713 }%
7714 \let\@org@gls@assign@plural\gls@assign@plural
7715 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7716 \let\@org@gls@assign@descplural\gls@assign@descplural
7717 \def\gls@assign@firstpl##1##2{%
7718 \@@gls@expand@field{##1}{firstpl}{##2}%
7719 }%
7720 \def\gls@assign@plural##1##2{%
7721 \@@gls@expand@field{##1}{plural}{##2}%
7722 }%
7723 \def\gls@assign@descplural##1##2{%
7724 \@@gls@expand@field{##1}{descplural}{##2}%
7725 }%
7726 \@do@newglossaryentry
7727 \let\gls@assign@plural\@org@gls@assign@plural
7728 \let\gls@assign@firstpl\@org@gls@assign@firstpl

```

```

7729 \let\gls@assign@descplural\@org@gls@assign@descplural
7730 }

```

`oteAcronymStyle` If footnote package option is specified, set the first use to append the long form (stored in description) as a footnote. Use the description key to store the long form.

```

7731 \newcommand*{\SetFootnoteAcronymStyle}{%
7732   \renewcommand{\newacronym}[4][]{%
7733     \ifx\@glsacronymlists\@empty
7734       \def\@glo@type{\acronymtype}%
7735       \setkeys{glossentry}{##1}%
7736       \DeclareAcronymList{\@glo@type}%
7737       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7738     \fi
7739     \glskeylisttok{##1}%
7740     \glslabeltok{##2}%
7741     \glsshorttok{##3}%
7742     \glslongtok{##4}%
7743     \newacronymhook
7744     \FootnoteNewAcronymDef
7745   }%

```

Set display

```

7746 \@for\@gls@type:=\@glsacronymlists\do{%
7747   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7748 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7749 \ifglsacrsmallcaps
7750   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7751   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7752 \else
7753   \ifglsacrsmaller
7754     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7755   \fi
7756 \fi

```

Check for option clash

```

7757 \ifglsacrdua
7758   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7759     can’t both be set}{}%
7760 \fi
7761 }%

```

`parenifnotempty` Do a space followed by the argument if the argument doesn't expand to empty or `\relax`. If argument isn't empty (or `\relax`), apply the macro to it given in the second argument.

```

7762 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7763   \protected@edef\gls@tmp{##1}%
7764   \ifdefempty\gls@tmp
7765     {}%

```

```

7766 {%
7767   \ifx\gls@tmp\@gls@default@value
7768   \else
7769     \space (#2{#1})%
7770   \fi
7771 }%
7772 }

```

nymDisplayStyle Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7773 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7774   \defglsentryfmt[#1]{%

```

```

7775     \ifdefempty\glscustomtext
7776     {%

```

Move the inserted text outside of \acronymfont

```

7777     \let\gls@org@insert\glsinsert
7778     \let\glsinsert\@empty
7779     \ifglsused{\glslabel}%
7780     {%
7781       \acronymfont{\glsentryfmt}\gls@org@insert
7782     }%
7783     {%
7784       \glsentryfmt
7785       \ifgls hassymbol{\glslabel}%
7786       {%
7787         \glsifplural
7788         {%
7789           \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7790         }%
7791         {%
7792           \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7793         }%
7794         \space
7795         (\glscapscase
7796         {\firstacronymfont{\@glo@symbol}}%
7797         {\firstacronymfont{\@glo@symbol}}%
7798         {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7799       }%
7800     }%
7801   }%
7802 }%
7803 {\glscustomtext\glsinsert}%
7804 }%
7805 }

```

l1NewAcronymDef

```

7806 \newcommand*{\SmallNewAcronymDef}{%

```

```

7807 \edef\@do@newglossaryentry{%
7808   \noexpand\newglossaryentry{\the\glslabeltok}%
7809   {%
7810     type=\acronymtype,%
7811     name={\noexpand\acronymfont{\the\glsshorttok}},%
7812     sort={\the\glsshorttok},%
7813     text={\the\glsshorttok},%

Default to the short plural.
7814     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7815     first={\the\glslongtok},%

Default to the long plural.
7816     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7817     short={\the\glsshorttok},%
7818     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7819     long={\the\glslongtok},%
7820     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7821     description={\noexpand\@glo@first},%
7822     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7823     symbol={\the\glsshorttok},%

Default to the short plural.
7824     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7825     \the\glskeylisttok
7826   }%
7827 }%
7828 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7829 \let\@org@gls@assign@plural\gls@assign@plural
7830 \let\@org@gls@assign@descplural\gls@assign@descplural
7831 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7832 \def\gls@assign@firstpl##1##2{%
7833   \@@gls@expand@field{##1}{firstpl}{##2}%
7834 }%
7835 \def\gls@assign@plural##1##2{%
7836   \@@gls@expand@field{##1}{plural}{##2}%
7837 }%
7838 \def\gls@assign@descplural##1##2{%
7839   \@@gls@expand@field{##1}{descplural}{##2}%
7840 }%
7841 \def\gls@assign@symbolplural##1##2{%
7842   \@@gls@expand@field{##1}{symbolplural}{##2}%
7843 }%
7844 \@do@newglossaryentry
7845 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7846 \let\gls@assign@plural\@org@gls@assign@plural
7847 \let\gls@assign@descplural\@org@gls@assign@descplural
7848 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7849 }

```

`\allAcronymStyle` Neither footnote nor description required, but smallcaps or smaller specified. Use the symbol

key to store the short form and first to store the long form.

```

7850 \newcommand*{\SetSmallAcronymStyle}{%
7851   \renewcommand{\newacronym}[4][1]{%
7852     \ifx\@glsacronymlists\@empty
7853       \def\@glo@type{\acronymtype}%
7854       \setkeys{glossentry}{##1}%
7855       \DeclareAcronymList{\@glo@type}%
7856       \SetSmallAcronymDisplayStyle{\@glo@type}%
7857     \fi
7858     \glskeylisttok{##1}%
7859     \glslabeltok{##2}%
7860     \glsshorttok{##3}%
7861     \glslongtok{##4}%
7862     \newacronymhook
7863     \SmallNewAcronymDef
7864   }%

```

Change the display since first only contains long form.

```

7865   \@for\@gls@type:=\@glsacronymlists\do{%
7866     \SetSmallAcronymDisplayStyle{\@gls@type}%
7867   }%

```

Redefine \acronymfont if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7868   \ifglsacrsmallcaps
7869     \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7870     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7871   \else
7872     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
7873   \fi

```

check for option clash

```

7874   \ifglsacrdua
7875     \ifglsacrsmallcaps
7876       \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7877         can’t both be set}{}%
7878     \else
7879       \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7880         can’t both be set}{}%
7881     \fi
7882   \fi
7883 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7884 \newcommand*{\SetDUADisplayStyle}[1]{%
7885   \defglsentryfmt[1]{\glsentryfmt}%
7886 }

```

UANewAcronymDef

```

7887 \newcommand*{\DUANewAcronymDef}{%

```



```

7888 \edef\@do@newglossaryentry{%
7889   \noexpand\newglossaryentry{\the\glslabeltok}%
7890   {%
7891     type=\acronymtype,%
7892     name={\the\glsshorttok},%
7893     text={\the\glslongtok},%
7894     first={\the\glslongtok},%
7895     plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7896     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7897     short={\the\glsshorttok},%
7898     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7899     long={\the\glslongtok},%
7900     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7901     description={\the\glslongtok},%
7902     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7903     symbol={\the\glsshorttok},%
7904     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7905     \the\glskeylisttok
7906   }%
7907 }%
7908 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7909 \let\@org@gls@assign@plural\gls@assign@plural
7910 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7911 \let\@org@gls@assign@descplural\gls@assign@descplural
7912 \def\gls@assign@firstpl##1##2{%
7913   \@gls@expand@field{##1}{firstpl}{##2}%
7914 }%
7915 \def\gls@assign@plural##1##2{%
7916   \@gls@expand@field{##1}{plural}{##2}%
7917 }%
7918 \def\gls@assign@symbolplural##1##2{%
7919   \@gls@expand@field{##1}{symbolplural}{##2}%
7920 }%
7921 \def\gls@assign@descplural##1##2{%
7922   \@gls@expand@field{##1}{descplural}{##2}%
7923 }%
7924 \@do@newglossaryentry
7925 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7926 \let\gls@assign@plural\@org@gls@assign@plural
7927 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7928 \let\gls@assign@descplural\@org@gls@assign@descplural
7929 }

```

\SetDUASStyle Always expand acronyms.

```

7930 \newcommand*{\SetDUASStyle}{%
7931   \renewcommand{\newacronym}[4][]{%
7932     \ifx\@glsacronymlists\empty
7933       \def\@glo@type{\acronymtype}%
7934       \setkeys{glossentry}{##1}%

```

```

7935     \DeclareAcronymList{\@glo@type}%
7936     \SetDUADisplayStyle{\@glo@type}%
7937     \fi
7938     \glskeylisttok{##1}%
7939     \glslabeltok{##2}%
7940     \glsshorttok{##3}%
7941     \gslongtok{##4}%
7942     \newacronymhook
7943     \DUANewAcronymDef
7944 }%

Set the display
7945 \@for\@gls@type:=\@glsacronymlists\do{%
7946     \SetDUADisplayStyle{\@gls@type}%
7947 }%
7948 }

```

SetAcronymStyle

```

7949 \newcommand*{\SetAcronymStyle}{%
7950     \SetDefaultAcronymStyle
7951     \ifglsacrdescription
7952         \ifglsacrfootnote
7953             \SetDescriptionFootnoteAcronymStyle
7954         \else
7955             \ifglsacrdua
7956                 \SetDescriptionDUAAcronymStyle
7957             \else
7958                 \SetDescriptionAcronymStyle
7959             \fi
7960         \fi
7961     \else
7962         \ifglsacrfootnote
7963             \SetFootnoteAcronymStyle
7964         \else
7965             \ifthenelse{\boolean{glsacrsmalldcaps}\OR
7966                 \boolean{glsacrsmaller}}{%
7967                 {%
7968                     \SetSmallAcronymStyle
7969                 }%
7970             }%
7971             \ifglsacrdua
7972                 \SetDUASyle
7973             \fi
7974         }%
7975     \fi
7976 \fi
7977 }

```

Set the acronym style according to the package options

```

7978 \SetAcronymStyle

```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\SetCustomDisplayStyle` Sets the acronym display style.

```
7979 \newcommand*{\SetCustomDisplayStyle}[1]{%
7980   \def\glsentryfmt[#1]{\glsentryfmt}%
7981 }
```

`\CustomAcronymFields`

```
7982 \newcommand*{\CustomAcronymFields}{%
7983   name={\the\glsshorttok},%
7984   description={\the\glslongtok},%
7985   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7986   firstplural={\acrfullformat
7987     {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7988     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%
7989   text={\the\glsshorttok},%
7990   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7991 }
```

`\CustomNewAcronymDef`

```
7992 \newcommand*{\CustomNewAcronymDef}{%
7993   \protected@edef\@do@newglossaryentry{%
7994     \noexpand\newglossaryentry{\the\glslabeltok}%
7995     {%
7996       type=\acronymtype,%
7997       short={\the\glsshorttok},%
7998       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7999       long={\the\glslongtok},%
8000       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
8001       user1={\the\glsshorttok},%
8002       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
8003       user3={\the\glslongtok},%
8004       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
8005       \CustomAcronymFields,%
8006       \the\glskeylisttok
8007     }%
8008   }%
8009   \@do@newglossaryentry
8010 }
```

`\SetCustomStyle`

```
8011 \newcommand*{\SetCustomStyle}{%
8012   \renewcommand{\newacronym}[4][\the\glsshorttok]{%
8013     \ifx\@glsacronymlists\empty
8014       \def\@glo@type{\acronymtype}%
8015     \fi
8016   }
```

```

8015     \setkeys{glossentry}{##1}%
8016     \DeclareAcronymList{\@glo@type}%
8017     \SetCustomDisplayStyle{\@glo@type}%
8018     \fi
8019     \glskeylisttok{##1}%
8020     \glslabeltok{##2}%
8021     \glsshorttok{##3}%
8022     \glslongtok{##4}%
8023     \newacronymhook
8024     \CustomNewAcronymDef
8025 }%
    Set the display
8026     \@for\@gls@type:=\@glsacronymlists\do{%
8027         \SetCustomDisplayStyle{\@gls@type}%
8028     }%
8029 }

```

1.19 Predefined Glossary Styles

The glossaries bundle comes with some predefined glossary styles. These need to be loaded now for the style option to use them.

First, the glossary hyper-navigation commands need to be loaded.

```
8030 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
8031 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `nolong` package option is used.

```
8032 \@gls@loadlong
```

The styles that use the `supertabular` environment. These are not loaded if the `nosuper` package option is used or if the package isn't installed.

```
8033 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
8034 \@gls@loadtree
```

The default glossary style is set according to the `style` package option, but can be overridden by `\glossarystyle`. The required style must be defined at this point.

```

8035 \ifx\@glossary@default@style\relax
8036 \else
8037   \setglossarystyle{\@glossary@default@style}
8038 \fi

```

1.20 Debugging Commands

\showgloparent

\showgloparent{<label>}

```
8039 \newcommand*{\showgloparent}[1]{%
8040   \expandafter\show\csname glo@glstetoklabel{#1}@parent\endcsname
8041 }
```

\showglolevel

\showglolevel{<label>}

```
8042 \newcommand*{\showglolevel}[1]{%
8043   \expandafter\show\csname glo@glstetoklabel{#1}@level\endcsname
8044 }
```

\showglotext

\showglotext{<label>}

```
8045 \newcommand*{\showglotext}[1]{%
8046   \expandafter\show\csname glo@glstetoklabel{#1}@text\endcsname
8047 }
```

\showgloplural

\showgloplural{<label>}

```
8048 \newcommand*{\showgloplural}[1]{%
8049   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
8050 }
```

\showglofirst

\showglofirst{<label>}

```
8051 \newcommand*{\showglofirst}[1]{%
8052   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
8053 }
```

\showglofirstpl

\showglofirstpl{<label>}

```
8054 \newcommand*{\showglofirstpl}[1]{%
8055   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
8056 }
```

\showglotype \showglotype{<label>}

```
8057 \newcommand*{\showglotype}[1]{%
8058   \expandafter\show\csname glo@glstdetoklabel{#1}@type\endcsname
8059 }
```

\showglocounter \showglocounter{<label>}

```
8060 \newcommand*{\showglocounter}[1]{%
8061   \expandafter\show\csname glo@glstdetoklabel{#1}@counter\endcsname
8062 }
```

\showglouserii \showglouserii{<label>}

```
8063 \newcommand*{\showglouserii}[1]{%
8064   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
8065 }
```

\showglouseriii \showglouseriii{<label>}

```
8066 \newcommand*{\showglouseriii}[1]{%
8067   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
8068 }
```

\showglouseriiii \showglouseriiii{<label>}

```
8069 \newcommand*{\showglouseriiii}[1]{%
8070   \expandafter\show\csname glo@glstdetoklabel{#1}@useriiii\endcsname
8071 }
```

\showglouseriv \showglouseriv{<label>}

```
8072 \newcommand*{\showglouseriv}[1]{%
8073   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
8074 }
```

\showglouserv \showglouserv{<label>}

```
8075 \newcommand*{\showglouserv}[1]{%
8076   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname
8077 }
```

\showglouservi \showglouservi{<label>}

```
8078 \newcommand*{\showglouservi}[1]{%
8079   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
8080 }
```

\showgloname \showgloname{<label>}

```
8081 \newcommand*{\showgloname}[1]{%
8082   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
8083 }
```

\showglodesc \showglodesc{<label>}

```
8084 \newcommand*{\showglodesc}[1]{%
8085   \expandafter\show\csname glo@glstdetoklabel{#1}@desc\endcsname
8086 }
```

owglodescplural \showglodescplural{<label>}

```
8087 \newcommand*{\showglodescplural}[1]{%
8088   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
8089 }
```

\showglosort \showglosort{<label>}

```
8090 \newcommand*{\showglosort}[1]{%
8091   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
8092 }
```

\showglosymbol \showglosymbol{<label>}

```
8093 \newcommand*{\showglosymbol}[1]{%
8094   \expandafter\show\csname glo@glstetoklabel{#1}@symbol\endcsname
8095 }
```

glosymbolplural \showglosymbolplural{<label>}

```
8096 \newcommand*{\showglosymbolplural}[1]{%
8097   \expandafter\show\csname glo@glstetoklabel{#1}@symbolplural\endcsname
8098 }
```

\showgloshort \showgloshort{<label>}

```
8099 \newcommand*{\showgloshort}[1]{%
8100   \expandafter\show\csname glo@glstetoklabel{#1}@short\endcsname
8101 }
```

\showglolong \showglolong{<label>}

```
8102 \newcommand*{\showglolong}[1]{%
8103   \expandafter\show\csname glo@glstetoklabel{#1}@long\endcsname
8104 }
```

\showgloindex \showgloindex{<label>}

```
8105 \newcommand*{\showgloindex}[1]{%
8106   \expandafter\show\csname glo@glstetoklabel{#1}@index\endcsname
8107 }
```

\showgloflag \showgloflag{<label>}

```
8108 \newcommand*{\showgloflag}[1]{%
8109   \expandafter\show\csname ifglo@glstetoklabel{#1}@flag\endcsname
8110 }
```


`\showgloloclist`

`\showgloloclist{<label>}`

```
8111 \newcommand*{\showgloloclist}[1]{%
8112   \expandafter\show\csname glo@glstetoklabel{#1}@loclist\endcsname
8113 }
```

`\showglofield`

`\showglofield{<label>}{<field>}`

```
8114 \newcommand*{\showglofield}[2]{%
8115   \cshow{glo@glstetoklabel{#1}@#2}%
8116 }
```

`\showacronymlists`

`\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
8117 \newcommand*{\showacronymlists}{%
8118   \show\@glsacronymlists
8119 }
```

`\showglossaries`

`\showglossaries`

Show list of defined glossaries.

```
8120 \newcommand*{\showglossaries}{%
8121   \show\@glo@types
8122 }
```

`\showglossaryin`

`\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
8123 \newcommand*{\showglossaryin}[1]{%
8124   \expandafter\show\csname @glotype@#1@in\endcsname
8125 }
```

`\showglossaryout`

`\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
8126 \newcommand*{\showglossaryout}[1]{%
8127   \expandafter\show\csname @glotype@#1@out\endcsname
8128 }
```

owglossarytitle

```
\showglossarytitle{<glossary-label>}
```

Show the title for the given glossary.

```
8129 \newcommand*{\showglossarytitle}[1]{%
8130   \expandafter\show\csname @glotype@#1@title\endcsname
8131 }
```

glossarycounter

```
\showglossarycounter{<glossary-label>}
```

Show the counter for the given glossary.

```
8132 \newcommand*{\showglossarycounter}[1]{%
8133   \expandafter\show\csname @glotype@#1@counter\endcsname
8134 }
```

glossaryentries

```
\showglossaryentries{<glossary-label>}
```

Show the list of entry labels for the given glossary.

```
8135 \newcommand*{\showglossaryentries}[1]{%
8136   \expandafter\show\csname glolist@#1\endcsname
8137 }
```

1.21 Compatibility with version 2.07 and below

In order to fix some bugs in v3.0, it was necessary to change the way information is written to the glo file, which also meant a change in the format of the Xindy style file. The compatibility option is meant for documents that use a customised Xindy style file with `\noist`. With the compatibility option, hopefully xindy will still be able to process the old document, but the bugs will remain. The issues in versions 2.07 and below:

- With xindy, the counter used by the entry was hard-coded into the Xindy style file. This meant that you couldn't use the counter to swap counters.
- With both xindy and makeindex, if used with hyperref and `\theH<counter>` was different to `\thecounter`, the link in the location number would be undefined.

```
8138 \csname ifglcompatible-2.07\endcsname
8139 \RequirePackage{glossaries-compatible-207}
8140 \fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`\gls{<label>}`” on first use but use “`\an\gls{<label>}`” on subsequent use.

```
8141 \NeedsTeXFormat{LaTeX2e}
```

```
8142 \ProvidesPackage{glossaries-prefix}[2020/02/13 v4.45 (NLCT)]
```

Pass all options to glossaries:

```
8143 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
8144 \ProcessOptions
```

Load glossaries:

```
8145 \RequirePackage{glossaries}
```

Add the new keys:

```
8146 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
8147 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
8148 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
8149 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\gls@keymap`:

```
8150 \appto\@gls@keymap{,%
```

```
8151   {prefixfirst}{prefixfirst},%
```

```
8152   {prefixfirstplural}{prefixfirstplural},%
```

```
8153   {prefix}{prefix},%
```

```
8154   {prefixplural}{prefixplural}}%
```

```
8155 }
```

Set the default values:

```
8156 \appto\@newglossaryentryprehook{%
```

```
8157   \def\@glo@entryprefix{}}%
```

```
8158   \def\@glo@entryprefixplural{}}%
```

```
8159   \let\@glo@entryprefixfirst\@gls@default@value
```

```
8160   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
8161 }
```

Set the assignment code:

```
8162 \appto\@newglossaryentryposthook{%
```

```
8163   \gls@assign@field{ }\@glo@label{prefix}{\@glo@entryprefix}}%
```

```
8164   \gls@assign@field{ }\@glo@label{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
8165   \expandafter\gls@assign@field\expandafter
```

```
8166     {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
8167     {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```
8168 \expandafter\gls@assign@field\expandafter
8169 {\csname glo@\glo@label @prefixplural\endcsname}{\@glo@label}%
8170 {prefixfirstplural}{\@glo@entryprefixfirstplural}%
8171 }
```

Define commands to access these fields:

entryprefixfirst

```
8172 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@\glsdetoklabel{#1}@prefixfirst}}
```

entryfirstplural

```
8173 \newcommand*{\glsentryprefixfirstplural}[1]{%
8174 \csuse{glo@\glsdetoklabel{#1}@prefixfirstplural}}
```

\glsentryprefix

```
8175 \newcommand*{\glsentryprefix}[1]{\csuse{glo@\glsdetoklabel{#1}@prefix}}
```

entryprefixplural

```
8176 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@\glsdetoklabel{#1}@prefixplural}}
```

Now for the initial upper case variants:

entryprefixfirst

```
8177 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
8178 \protected@edef\@glo@text{\csname glo@\glsdetoklabel{#1}@prefixfirst\endcsname}%
8179 \xmakefirstuc\@glo@text
8180 }
```

entryfirstplural

```
8181 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
8182 \protected@edef\@glo@text{\csname glo@\glsdetoklabel{#1}@prefixfirstplural\endcsname}%
8183 \xmakefirstuc\@glo@text
8184 }
```

\Glsentryprefix

```
8185 \newrobustcmd*{\Glsentryprefix}[1]{%
8186 \protected@edef\@glo@text{\csname glo@\glsdetoklabel{#1}@prefix\endcsname}%
8187 \xmakefirstuc\@glo@text
8188 }
```

entryprefixplural

```
8189 \newrobustcmd*{\Glsentryprefixplural}[1]{%
8190 \protected@edef\@glo@text{\csname glo@\glsdetoklabel{#1}@prefixplural\endcsname}%
8191 \xmakefirstuc\@glo@text
8192 }
```

Define commands to determine if the prefix keys have been set:

`\ifglshasprefix`

```
8193 \newcommand*{\ifglshasprefix}[3]{%
8194   \ifcempty{glo@glstdetoklabel{#1}@prefix}%
8195   {#3}%
8196   {#2}%
8197 }
```

`hasprefixplural`

```
8198 \newcommand*{\ifglshasprefixplural}[3]{%
8199   \ifcempty{glo@glstdetoklabel{#1}@prefixplural}%
8200   {#3}%
8201   {#2}%
8202 }
```

`shasprefixfirst`

```
8203 \newcommand*{\ifglshasprefixfirst}[3]{%
8204   \ifcempty{glo@glstdetoklabel{#1}@prefixfirst}%
8205   {#3}%
8206   {#2}%
8207 }
```

`efixfirstplural`

```
8208 \newcommand*{\ifglshasprefixfirstplural}[3]{%
8209   \ifcempty{glo@glstdetoklabel{#1}@prefixfirstplural}%
8210   {#3}%
8211   {#2}%
8212 }
```

`fix@record@hook` Need to take into account the possibility that glossaries-extra might be loaded with the record option.

```
8213 \providecommand{\@glsprefix@record@hook}[2]{%
8214   \ifdef\@glsextr@record
8215   {\@glsextr@record{#1}{#2}{glslink}}%
8216   {}%
8217 }
```

`\glsprefixsep` Separator between prefix and term. Does nothing by default.

```
8218 \newcommand{\glsprefixsep}{}%
```

Define commands that insert the prefix before commands like `\gls`:

`\pgls`

```
8219 \newrobustcmd{\pgls}{\@glshyp@opt\@pgls}
```

`\@pgls` Unstarred version.

```
8220 \newcommand*{\@pgls}[2][ ]{%
8221   \new@ifnextchar[%
8222   {\@pgls@{#1}{#2}}%
```

```

8223 {\@pgls@{#1}{#2}[]}%
8224 }

```

\@pgls@ Read in the final optional argument:

```

8225 \def\@pgls@#1#2[#3]{%
8226   \@glsprefix@record@hook{#1}{#2}%
8227   \glsdoifexists{#2}%
8228   {%
8229     \ifglsused{#2}%
8230     {%
8231       \ifglshasprefix{#2}{\glstryentryprefix{#2}\glsprefixsep}{}%
8232     }%
8233     {%
8234       \ifglshasprefixfirst{#2}{\glstryentryprefixfirst{#2}\glsprefixsep}{}%
8235     }%
8236     \@gls@{#1}{#2}[#3]%
8237   }%
8238 }

```

Similarly for the plural version:

```

\pglsp1
8239 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

8240 \newcommand*{\@pglsp1}[2][]{%
8241   \new@ifnextchar[%
8242     {\@pglsp1@{#1}{#2}}%
8243     {\@pglsp1@{#1}{#2}[]}%
8244 }

```

\@pglsp1@ Read in the final optional argument:

```

8245 \def\@pglsp1@#1#2[#3]{%
8246   \@glsprefix@record@hook{#1}{#2}%
8247   \glsdoifexists{#2}%
8248   {%
8249     \ifglsused{#2}%
8250     {%
8251       \ifglshasprefixplural{#2}{\glstryentryprefixplural{#2}\glsprefixsep}{}%
8252     }%
8253     {%
8254       \ifglshasprefixfirstplural{#2}%
8255         {\glstryentryprefixfirstplural{#2}\glsprefixsep}{}%
8256     }%
8257     \@glspl@{#1}{#2}[#3]%
8258   }%
8259 }

```

Now for the first letter upper case versions:

```
\Pgl's
8260 \newrobustcmd{\Pgl's}{\@gls@hyp@opt\@Pgl's}
```

\@Pgl's Unstarred version.

```
8261 \newcommand*{\@Pgl's}[2] [] {%
8262   \new@ifnextchar[%
8263     {\@Pgl's@{#1}{#2}}%
8264     {\@Pgl's@{#1}{#2} []}%
8265 }
```

\@Pgl's@ Read in the final optional argument:

```
8266 \def\@Pgl's@#1#2[#3] {%
8267   \@gls@prefix@record@hook{#1}{#2}%
8268   \glsdoifexists{#2}%
8269   {%
8270     \ifglsused{#2}%
8271     {%
8272       \ifglshasprefix{#2}%
8273       {%
8274         \Glsentryprefix{#2}%
8275         \gls@prefixsep
8276         \@gls@{#1}{#2}[#3]%
8277       }%
8278       {\@Gls@{#1}{#2}[#3]}%
8279     }%
8280     {%
8281       \ifglshasprefixfirst{#2}%
8282       {%
8283         \Glsentryprefixfirst{#2}%
8284         \gls@prefixsep
8285         \@gls@{#1}{#2}[#3]%
8286       }%
8287       {\@Gls@{#1}{#2}[#3]}%
8288     }%
8289   }%
8290 }
```

Similarly for the plural version:

```
\Pgl'spl
8291 \newrobustcmd{\Pgl'spl}{\@gls@hyp@opt\@Pgl'spl}
```

\@Pgl'spl Unstarred version.

```
8292 \newcommand*{\@Pgl'spl}[2] [] {%
8293   \new@ifnextchar[%
8294     {\@Pgl'spl@{#1}{#2}}%
8295     {\@Pgl'spl@{#1}{#2} []}%
8296 }
```

\@Pglsp1@ Read in the final optional argument:

```

8297 \def\@Pglsp1@#1#2[#3]{%
8298   \@glsprefix@record@hook{#1}{#2}%
8299   \glsoifexists{#2}%
8300   {%
8301     \ifglused{#2}%
8302     {%
8303       \ifglshasprefixplural{#2}%
8304       {%
8305         \Glsentryprefixplural{#2}%
8306         \glsprefixsep
8307         \@glsp1@{#1}{#2}[#3]%
8308       }%
8309       {\@Glspl@{#1}{#2}[#3]}%
8310     }%
8311     {%
8312       \ifglshasprefixfirstplural{#2}%
8313       {%
8314         \Glsentryprefixfirstplural{#2}%
8315         \glsprefixsep
8316         \@glsp1@{#1}{#2}[#3]%
8317       }%
8318       {\@Glspl@{#1}{#2}[#3]}%
8319     }%
8320   }%
8321 }
```

Finally the all upper case versions:

\PGLS

```

8322 \newrobustcmd{\PGLS}{\@glshyp@opt\@PGLS}
```

\@PGLS Unstarred version.

```

8323 \newcommand*{\@PGLS}[2][ ]{%
8324   \new@ifnextchar[%
8325   {\@PGLS@{#1}{#2}}%
8326   {\@PGLS@{#1}{#2}[ ]}%
8327 }
```

\@PGLS@ Read in the final optional argument:

```

8328 \def\@PGLS@#1#2[#3]{%
8329   \@glsprefix@record@hook{#1}{#2}%
8330   \glsoifexists{#2}%
8331   {%
8332     \ifglused{#2}%
8333     {%
8334       \ifglshasprefix{#2}%
8335       {\mfirstucMakeUppercase{\Glsentryprefix{#2}\glsprefixsep}}{}}%
```



```

8336 }%
8337 {%
8338     \ifglshasprefixfirst{#2}%
8339     {\mfirstucMakeUppercase{\glstryentryprefixfirst{#2}\glsprefixsep}}{}%
8340 }%
8341 \@GLS@{#1}{#2}[#3]%
8342 }%
8343 }

```

Plural version:

\PGLSp1

```

8344 \newrobustcmd{\PGLSp1}{\@glshyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

8345 \newcommand*{\@PGLSp1}[2][ ]{%
8346     \new@ifnextchar[%
8347     {\@PGLSp1@{#1}{#2}}%
8348     {\@PGLSp1@{#1}{#2}[ ]}%
8349 }

```

\@PGLSp1@ Read in the final optional argument:

```

8350 \def\@PGLSp1@#1#2[#3]{%
8351     \@glsprefix@record@hook{#1}{#2}%
8352     \glsdoidexists{#2}%
8353     {%
8354         \ifglused{#2}%
8355         {%
8356             \ifglshasprefixplural{#2}%
8357             {\mfirstucMakeUppercase{\glstryentryprefixplural{#2}\glsprefixsep}}{}%
8358         }%
8359         {%
8360             \ifglshasprefixfirstplural{#2}%
8361             {\mfirstucMakeUppercase{\glstryentryprefixfirstplural{#2}\glsprefixsep}}{}%
8362         }%
8363         \@GLSp1@{#1}{#2}[#3]%
8364     }%
8365 }

```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8366 \ProvidesPackage{glossary-hypernav}[2020/02/13 v4.45 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [section 1.16](#).) `\printglossary` (and `\printglossaries`) set `\@glo@type` to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

`\glsnavhyperlink[⟨type⟩]{⟨label⟩}{⟨text⟩}`

This command makes `⟨text⟩` a hyperlink to the glossary group whose label is given by `⟨label⟩` for the glossary given by `⟨type⟩`.

`glsnavhyperlink`

```
8367 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8368   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8369   \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

`avhyperlinkname`

Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
8370 \newcommand*{\glsnavhyperlinkname}[2]{\glsn:#1@#2}
```

`snavhypertarget`

`\glsnavhypertarget[⟨type⟩]{⟨label⟩}{⟨text⟩}`

This command makes `⟨text⟩` a hypertarget for the glossary group whose label is given by `⟨label⟩` in the glossary given by `⟨type⟩`. If `⟨type⟩` is omitted, `\@glo@type` is used which is set by `\printglossary` to the current glossary label.

```
8371 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
8372   \@glsnavhypertarget{#1}{#2}{#3}%
8373 }
```

The actual code is now in an internal command that doesn't have an optional argument, which makes it easier to save and restore the original behaviour.

`snavhypertarget`

```
8374 \newcommand*{\@glsnavhypertarget}[3]{%}
```

Add this group to the aux file for re-run check.

```
8375 \protected@write\auxout{}\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
8376 \@glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8377 \expandafter\let
```

```
8378 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8379 \@for\@gls@elem:=\@gls@list\do{%
```

```
8380 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}%}
```

Check if list terminated prematurely.

```
8381 \if@endfor
```

```
8382 \else
```

This group was not included in the list, so issue a warning.

```
8383 \GlossariesWarningNoLine{Navigation panel
```

```
8384 for glossary type ‘#1’~Jmissing group ‘#2’}%
```

```
8385 \gdef\gls@hypergroup@rerun{%
```

```
8386 \GlossariesWarningNoLine{Navigation panel
```

```
8387 has changed. Rerun LaTeX}}%
```

```
8388 \fi
```

```
8389 }
```

`\hypergroup@rerun` Give a warning at the end if re-run required

```
8390 \let\gls@hypergroup@rerun\relax
```

```
8391 \AtEndDocument{\gls@hypergroup@rerun}
```

`\@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergroup@list@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
8392 \newcommand*{\@gls@hypergroup}[2]{%
```

```
8393 \@ifundefined{\@gls@hypergroup@list@#1}{%
```

```
8394 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}%
```

```
8395 }{%
```

```
8396 \expandafter\let\expandafter\@gls@tmp
```

```
8397 \csname @gls@hypergroup@list@#1\endcsname
```

```
8398 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
```

```
8399 \@gls@tmp,#2}%
```

```
8400 }%
```

```
8401 }
```

The `\glsnavigation` command displays a simple glossary group navigation. The symbol and number elements are defined separately, so that they can be suppressed if need be. (In earlier versions this command will produce a link to all 28 groups, but some groups may not be defined if there are groups that do not contain any terms, in which case you will get an undefined hyperlink warning. Version 1.14 changed this to only use labels for groups that are present.) Now for the whole navigation bit:

`\glsnavigation`

```
8402 \newcommand*{\glsnavigation}{%
8403   \def\@gls@between{}%
8404   \ifcsundef\@gls@hypergroup\list\@glo@type}%
8405   {%
8406     \def\@gls@list{}%
8407   }%
8408   {%
8409     \expandafter\let\expandafter\@gls@list
8410       \csname @gls@hypergroup\list\@glo@type\endcsname
8411   }%
8412   \@for\@gls@tmp:=\@gls@list\do{%
8413     \@gls@between

8414     \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8415     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8416     \let\@gls@between\glshypernavsep
8417   }%
8418 }
```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8419 \newcommand*{\glshypernavsep}{\space\textbar\space}
```

The `\glssymbolnav` produces a simple navigation set of links for just the symbol and number groups. This used to be used at the start of `\glsnavigation`. This command is no longer needed.

`\glssymbolnav`

```
8420 \newcommand*{\glssymbolnav}{%
8421   \glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}}%
8422   \glshypernavsep
8423   \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
8424   \glshypernavsep
8425 }
```

3.2 In-line Style (glossary-inline.sty)

This defines an in-line style where the entries are comma-separated with just the name and description displayed.

```
8426 \ProvidesPackage{glossary-inline}[2020/02/13 v4.45 (NLCT)]
```

`inline` Define the inline style.

```
8427 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then changed by
    \glossentry)
8428   \renewenvironment{theglossary}%
8429   {%
```

```

8430     \def\gls@inlinesep{}%
8431     \def\gls@inlinesubsep{}%
8432     \def\gls@inlinepostchild{}%
8433     }%
8434     {\glspostinline}%

```

No header:

```
8435 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
8436 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```

8437 \renewcommand{\glossentry}[2]{%
8438   \glsinlinedopostchild
8439   \gls@inlinesep
8440   \glsentryitem{##1}%
8441   \glsinlinenameformat{##1}{%
8442     \glossentryname{##1}%
8443   }%
8444   \ifglstdescsuppressed{##1}%
8445   {%
8446     \glsinlineemptydescformat
8447     {%
8448       \glossentrysymbol{##1}%
8449     }%
8450     {%
8451       ##2%
8452     }%
8453   }%
8454   {%
8455     \ifglshasdesc{##1}%
8456     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8457     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8458   }%
8459   \ifglshaschildren{##1}%
8460   {%
8461     \glsresetsubentrycounter
8462     \glsinlineparentchildseparator
8463     \def\gls@inlinesubsep{}%
8464     \def\gls@inlinepostchild{\glsinlinepostchild}%
8465   }%
8466   {}%
8467   \def\gls@inlinesep{\glsinlineseparator}%
8468 }%

```

Sub-entries display description:

```

8469 \renewcommand{\subglossentry}[3]{%
8470   \gls@inlinesubsep%
8471   \glsinlinesubnameformat{##2}{%

```

```

8472     \glossentryname{##2}}}%
8473     \glssubentryitem{##2}%
8474     \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8475     \def\gls@inlinesubsep{\glsinlinesubseparator}%
8476 }%

```

Nothing special between groups:

```

8477 \renewcommand*{\glsgroupskip}{}%
8478 }

```

linedopostchild

```

8479 \newcommand*{\glsinlinedopostchild}{%
8480     \gls@inlinepostchild
8481     \def\gls@inlinepostchild{}}%
8482 }

```

inlineseparator Separator to use between entries.

```

8483 \newcommand*{\glsinlineseparator}{;\space}

```

inlinesubseparator Separator to use between sub-entries.

```

8484 \newcommand*{\glsinlinesubseparator}{,\space}

```

parentchildseparator Separator to use between parent and children.

```

8485 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

inlinepostchild Hook to use between child and next entry

```

8486 \newcommand*{\glsinlinepostchild}{}

```

\glspostinline Terminator for inline glossary.

```

8487 \newcommand*{\glspostinline}{\glspostdescription\space}

```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```

8488 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

inlinedescformat Formats the entry's description, symbol and location list:

```

8489 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```

8490 \newcommand*{\glsinlineemptydescformat}[2]{}

```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```

8491 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}

```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```

8492 \newcommand*{\glsinlinesubdescformat}[3]{#1}

```

3.3 List Style (glossary-list.sty)

The style file defines glossary styles that use the description environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

8493 \ProvidesPackage{glossary-list}[2020/02/13 v4.45 (NLCT)]

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8494 \providecommand{\indexspace}{%
8495   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8496 }
```

`tgrouphaderfmt` Provide a way of adjusting the format of the group headings.

```
8497 \newcommand*{\glslistgrouphaderfmt}[1]{#1}
```

`tnavigationitem` Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of `item` to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```
8498 \newcommand*{\glslistnavigationitem}[1]{\item[#1]}
```

`list` The list glossary style uses the description environment. The group separator `\glsgrpskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the glossaries package.

```
8499 \newglossarystyle{list}{%
```

Use description environment:

```
8500 \renewenvironment{theglossary}%
8501   {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8502 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8503 \renewcommand*{\glsgrpskip}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8504 \renewcommand*{\glossentry}[2]{%
8505   \item[\glsentryitem{##1}%
8506     \glstarget{##1}{\glossentryname{##1}}]
8507   \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
8508 \renewcommand*{\subglossentry}[3]{%
8509   \glssubentryitem{##2}%
```

```

8510 \glstarget{##2}{\strut}\space
8511 \glossentrydesc{##2}\glspostdescription\space ##3.}%

Add vertical space between groups:

8512 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8513 }

```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```

8514 \newglossarystyle{listgroup}{%
    Base it on the list style:
8515 \setglossarystyle{list}%

    Each group has a heading:
8516 \renewcommand*{\glsgroupheading}[1]{%
8517 \item[\glslistgroupheaderfmt{\glsgrouptitle{##1}}]}

```

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```

8518 \newglossarystyle{listhypergroup}{%
    Base it on the list style:
8519 \setglossarystyle{list}%

    Add navigation links at the start of the environment.
8520 \renewcommand*{\glossaryheader}{%
8521 \glslistnavigationitem{\glslnavigation}}%

    Each group has a heading with a hypertarget:
8522 \renewcommand*{\glsgroupheading}[1]{%
8523 \item[\glslistgroupheaderfmt
8524 {\glslnavhypertarget{##1}{\glsgrouptitle{##1}}]}

```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```

8525 \newglossarystyle{altlist}{%
    Base it on the list style:
8526 \setglossarystyle{list}%

    Main (level 0) entries start a new item in the list with a line break after the entry name:
8527 \renewcommand*{\glossentry}[2]{%
8528 \item[\glsentryitem{##1}%
8529 \glstarget{##1}{\glssentryname{##1}}]}

```

Version 3.04 changed `\newline` to the following paragraph break stuff (thanks to Daniel Gebhardt for supplying the fix) to prevent a page break occurring at this point.

```

8530 \mbox{}\par\nobreak\@afterheading
8531 \glossentrydesc{##1}\glspostdescription\space ##2}%

```


Sub-entries start a new paragraph:

```
8532 \renewcommand{\subglossentry}[3]{%  
8533   \par  
8534   \glssubentryitem{##2}%  
8535   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%  
8536 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8537 \newglossarystyle{altlistgroup}{%  
  Base it on the altlist style:  
8538   \setglossarystyle{altlist}%  
  Each group has a heading:  
8539   \renewcommand*{\glsgroupeheading}[1]{%  
8540     \item[\glslistgroupheaderfmt{\glsgroupeheading{##1}}]}
```

altlisthypergroup The altlisthypergroup glossary style is like the altlistgroup style, but has a set of links to the groups at the start of the glossary.

```
8541 \newglossarystyle{altlisthypergroup}{%  
  Base it on the altlist style:  
8542   \setglossarystyle{altlist}%  
  Add navigation links at the start of the environment.  
8543   \renewcommand*{\glossaryheader}{%  
8544     \glslistnavigationitem{\glslistnavigation}}%  
  Each group has a heading with a hypertarget:  
8545   \renewcommand*{\glsgroupeheading}[1]{%  
8546     \item[\glslistgroupheaderfmt  
8547       {\glslistnavigationitem{##1}{\glsgroupeheading{##1}}}]}
```

listdotted The listdotted glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
8548 \newglossarystyle{listdotted}{%  
  Base it on the list style:  
8549   \setglossarystyle{list}%  
  Each main (level 0) entry starts a new item:  
8550   \renewcommand*{\glossentry}[2]{%  
8551     \item[\makebox[\glslistdottedwidth][l]{%  
8552       \glssentryitem{##1}%  
8553       \glstarget{##1}{\glossentryname{##1}}}%  
8554     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```

8555 \renewcommand*{\subglossentry}[3]{%
8556   \item[\makebox[\glslstdottedwidth][l]{%
8557     \glssubentryitem{##2}%
8558     \glstarget{##2}{\glossentryname{##2}}}%
8559   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8560 }
```

`listdottedwidth`

```

8561 \newlength\glslstdottedwidth
8562 \setlength{\glslstdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8563 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
8564 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```

8565 \renewcommand*{\glossentry}[2]{%
8566   \item[\glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
8567 }
```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8568 \ProvidesPackage{glossary-long}[2020/02/13 v4.45 (NLCT)]
```

Requires the package:

```
8569 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. (There's a chance that the user may specify `nolong` and then load later, in which case `\glsdescwidth` may have already been defined by . The same goes for `\glspagelistwidth`.)

```

8570 \@ifundefined{glsdescwidth}{%
8571   \newlength\glsdescwidth
8572   \setlength{\glsdescwidth}{0.6\hsize}
8573 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column.

```

8574 \@ifundefined{glspagelistwidth}{%
8575   \newlength\glspagelistwidth
8576   \setlength{\glspagelistwidth}{0.1\hsize}
8577 }{}
```

long The long glossary style command which uses the longtable environment:

```
8578 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
8579 \renewenvironment{theglossary}{%
8580     {\begin{longtable}\lp{\glstdescwidth}}}%
8581     {\end{longtable}}}%
```

Do nothing at the start of the environment:

```
8582 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8583 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8584 \renewcommand{\glossentry}[2]{%
8585     \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8586     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8587 }%
```

Sub entries displayed on the following row without the name:

```
8588 \renewcommand{\subglossentry}[3]{%
8589     &
8590     \glssubentryitem{##2}%
8591     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8592     ##3\tabularnewline
8593 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip`
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8594 \ifglsgroupskip
8595     \renewcommand*{\glsgroupskip}{}%
8596 \else
8597     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8598 \fi
8599 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
8600 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
8601 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
8602 \renewenvironment{theglossary}{%
8603     \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8604 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8605 }
```

longheader The longheader style is like the long style but with a header:

```
8606 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8607 \setglossarystyle{long}%
```

Set the table's header:

```
8608 \renewcommand*{\glossaryheader}{%
8609 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8610 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8611 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8612 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8613 \renewcommand*{\glossaryheader}{%
8614 \hline\bfseries \entryname & \bfseries
8615 \descriptionname\tabularnewline\hline
8616 \endhead
8617 \hline\endfoot}%
8618 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
8619 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
8620 \renewenvironment{theglossary}%
8621 {\begin{longtable}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8622 {\end{longtable}}%
```

No table header:

```
8623 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8624 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8625 \renewcommand{\glossentry}[2]{%
8626 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8627 \glossentrydesc{##1} & ##2\tabularnewline
8628 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8629 \renewcommand{\subglossentry}[3]{%
8630 &
8631 \glssubentryitem{##2}%
8632 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8633 ##3\tabularnewline
8634 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8635 \ifglsgroupskip
8636 \renewcommand*{\glsgroupskip}{}%
8637 \else
8638 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8639 \fi
8640 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
8641 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8642 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8643 \renewenvironment{theglossary}%
8644 {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8645 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8646 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8647 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
8648 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
8649 \setglossarystyle{long3col}%
    Set the table's header:
8650 \renewcommand*{\glossaryheader}{%
8651 \bfseries\entryname&\bfseries\descriptionname&
8652 \bfseries\pagelistname\tabularnewline\endhead}%
8653 }
```

colheaderborder The long3colheaderborder style is like the above but with a border

```
8654 \newglossarystyle{long3colheaderborder}{%
    Base it on the glostylelong3colborder style:
8655 \setglossarystyle{long3colborder}%
    Set the table's header and add horizontal line at table's foot:
8656 \renewcommand*{\glossaryheader}{%
8657 \hline
8658 \bfseries\entryname&\bfseries\descriptionname&
8659 \bfseries\pagelistname\tabularnewline\hline\endhead
8660 \hline\endfoot}%
8661 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8662 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8663 \renewenvironment{theglossary}{%
```

```
8664 {\begin{longtable}{l l l l}}%
```

```
8665 {\end{longtable}}}%
```

No table header:

```
8666 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8667 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a single row (name in first column, description in second column, symbol in third column, page list in last column):

```
8668 \renewcommand{\glossentry}[2]{%
```

```
8669 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
8670 \glossentrydesc{##1} &
```

```
8671 \glossentrysymbol{##1} &
```

```
8672 ##2\tabularnewline
```

```
8673 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8674 \renewcommand{\subglossentry}[3]{%
```

```
8675 &
```

```
8676 \glssubentryitem{##2}%
```

```
8677 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
8678 \glossentrysymbol{##2} & ##3\tabularnewline
```

```
8679 }%
```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8680 \ifglsgnogroupskip
```

```
8681 \renewcommand*{\glsgroupskip}{}%
```

```
8682 \else
```

```
8683 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
```

```
8684 \fi
```

```
8685 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8686 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8687 \setglossarystyle{long4col}%
```

Table has a header:

```
8688 \renewcommand*{\glossaryheader}{%
```

```
8689 \bfseries\entryname&\bfseries\descriptionname&
```

```
8690 \bfseries \symbolname&
```

```

8691 \bfseries\pagelistname\tabularnewline\endhead}%
8692 }

```

long4colborder The long4colborder style is like long4col but with a border.

```

8693 \newglossarystyle{long4colborder}{%
    Base it on the glostylelong4col style:
8694 \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8695 \renewenvironment{theglossary}%
8696 {\begin{longtable}{|l|l|l|l|}}%
8697 {\end{longtable}}%
    Add horizontal lines to the head and foot of the table:
8698 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8699 }

```

colheaderborder The long4colheaderborder style is like the above but with a border.

```

8700 \newglossarystyle{long4colheaderborder}{%
    Base it on the glostylelong4col style:
8701 \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8702 \renewenvironment{theglossary}%
8703 {\begin{longtable}{|l|l|l|l|}}%
8704 {\end{longtable}}%
    Add table header and horizontal line at the table's foot:
8705 \renewcommand*{\glossaryheader}{%
8706 \hline\bfseries\entryname&\bfseries\descriptionname&
8707 \bfseries \symbolname&
8708 \bfseries\pagelistname\tabularnewline\hline\endhead
8709 \hline\endfoot}%
8710 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8711 \newglossarystyle{altlong4col}{%
    Base it on the glostylelong4col style:
8712 \setglossarystyle{long4col}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8713 \renewenvironment{theglossary}%
8714 {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8715 {\end{longtable}}%
8716 }

```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```

8717 \newglossarystyle{altlong4colheader}{%
      Base it on the glostylelong4colheader style:
8718   \setglossarystyle{long4colheader}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8719   \renewenvironment{theglossary}%
8720     {\begin{longtable}{\lp{\glsgdescwidth}\lp{\glspagelistwidth}}}%
8721     {\end{longtable}}}%
8722 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```

8723 \newglossarystyle{altlong4colborder}{%
      Base it on the glostylelong4colborder style:
8724   \setglossarystyle{long4colborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8725   \renewenvironment{theglossary}%
8726     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8727     {\end{longtable}}}%
8728 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```

8729 \newglossarystyle{altlong4colheaderborder}{%
      Base it on the glostylelong4colheaderborder style:
8730   \setglossarystyle{long4colheaderborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8731   \renewenvironment{theglossary}%
8732     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8733     {\end{longtable}}}%
8734 }
```

3.5 Glossary Styles using longtable and booktabs (the `glossary-longbooktabs`) package

The styles here are based on David Carlisle's patch at <http://tex.stackexchange.com/a/56890>

```

8735 \ProvidesPackage{glossary-longbooktabs}[2020/02/13 v4.45 (NLCT)]
```

Requires `booktabs` package:

```

8736 \RequirePackage{booktabs}
```


and the base packages for long styles:

```
8737 \RequirePackage{glossary-long}
8738 \RequirePackage{glossary-longragged}
```

(longtable and array loaded by those packages).

long-booktabs The long-booktabs style is similar to the longheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8739 \newglossarystyle{long-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8740 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8741 \setglossarystyle{long}{%
```

Add a header with rules.

```
8742 \renewcommand*{\glossaryheader}{%
8743 \toprule \bfseries \entryname & \bfseries
8744 \descriptionname\tabularnewline\midrule\endhead
8745 \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8746 \ifglsgnogroupskip
8747 \renewcommand*{\glsgroupskip}{}%
8748 \else
8749 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8750 \fi
8751 }
```

long3col-booktabs The long3col-booktabs style is similar to the long3colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8752 \newglossarystyle{long3col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8753 \glspatchLToutput
```

Use the long3col style as a base.

```
8754 \setglossarystyle{long3col}{%
```

Add a header with rules.

```
8755 \renewcommand*{\glossaryheader}{%
8756 \toprule \bfseries \entryname &
8757 \bfseries \descriptionname &
8758 \bfseries \pagelistname
8759 \tabularnewline\midrule\endhead
8760 \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8761 \ifglsgroupskip
8762 \renewcommand*{\glsgroupskip}{}%
8763 \else
8764 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8765 \fi
8766 }
```

ng4col-booktabs The long4col-booktabs style is similar to the long4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8767 \newglossarystyle{long4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8768 \glspatchLToutput
```

Use the long4col style as a base.

```
8769 \setglossarystyle{long4col}{%
```

Add a header with rules.

```
8770 \renewcommand*{\glossaryheader}{%
8771 \toprule \bfseries \entryname &
8772 \bfseries \descriptionname &
8773 \bfseries \symbolname &
8774 \bfseries \pagelistname
8775 \tabularnewline\midrule\endhead
8776 \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8777 \ifglsgroupskip
8778 \renewcommand*{\glsgroupskip}{}%
8779 \else
8780 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8781 \fi
8782 }
```

ng4col-booktabs The altlong4col-booktabs style is similar to the altlong4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8783 \newglossarystyle{altlong4col-booktabs}{%
```

The patch \glspatchLToutput is already applied in long4col-booktabs and so doesn't need to be here.

```
8784 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8785 \setglossarystyle{long4col-booktabs}{%
```

Change the column specifications:

```
8786 \renewenvironment{theglossary}%  
8787   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%  
8788   {\end{longtable}}}%  
8789 }
```

Ragged styles.

ragged-booktabs The longragged-booktabs style is similar to the longragged style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8790 \newglossarystyle{longragged-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8791 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8792 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8793 \renewenvironment{theglossary}%  
8794   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%  
8795   {\end{longtable}}}%  
8796 }
```

ed3col-booktabs The longragged3col-booktabs style is similar to the longragged3col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8797 \newglossarystyle{longragged3col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8798 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8799 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8800 \renewenvironment{theglossary}%  
8801   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%  
8802     >{\raggedright}p{\glspagelistwidth}}}%  
8803   {\end{longtable}}}%  
8804 }
```

ed4col-booktabs The altlongragged4col-booktabs style is similar to the altlongragged4col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8805 \newglossarystyle{altlongragged4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8806 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8807 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8808 \renewenvironment{theglossary}%  
8809 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8810 >{\raggedright}p{\glspagelistwidth}}}%  
8811 {\end{longtable}}%  
8812 }
```

sLTpenaltycheck

```
8813 \newcommand*{\glslTpenaltycheck}{%  
8814 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8815 }
```

enaltygroupskip

```
8816 \newcommand{\glspenaltygroupskip}{%  
8817 \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8818 \let\@gls@org@LT@output\LT@output  
8819 \newcommand*{\glstoreLToutput}{\let\LT@output\@gls@org@LT@output}
```

This is David's patch, but I've replaced the hard-coded values with \glslTpenaltycheck to make it easier to adjust.

lspatchLToutput

```
8820 \newcommand*{\glspatchLToutput}{%  
8821 \renewcommand*{\LT@output}{%  
8822 \ifnum\outputpenalty < -\@Mi  
8823 \ifnum\outputpenalty > -\LT@end@pen  
8824 \LT@err{floats and marginpars not allowed in a longtable}\@ehc  
8825 \else  
8826 \setbox\z@\vbox{\unvbox\@cclv}%  
8827 \ifdim \ht\LT@lastfoot>\ht\LT@foot  
8828 \dimen@ \pagegoal  
8829 \advance\dimen@-\ht\LT@lastfoot  
8830 \ifdim\dimen@<\ht\z@  
8831 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%  
8832 \@makecol  
8833 \@outputpage  
8834 \setbox\z@\vbox{\box\LT@head\glslTpenaltycheck}%  
8835 \fi  
8836 \fi  
8837 \global\@colroom\@colht  
8838 \global\vsize\@colht  
8839 {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%  
8840 \fi  
8841 \else
```

```

8842 \setbox\@cclv\vbox{\unvbox\@ccclv\copy\LT@foot\vss}%
8843 \@makecol
8844 \@outputpage
8845 \global\ysize\@colroom
8846 \copy\LT@head
8847 \glsLTpenaltycheck
8848 \nobreak
8849 \fi
8850 }%
8851 }

```

3.6 Glossary Styles using longtable (the glossary-longragged package)

The glossary styles defined in the package used the longtable environment in the glossary and use ragged right formatting for the multiline columns.

```
8852 \ProvidesPackage{glossary-longragged}[2020/02/13 v4.45 (NLCT)]
```

Requires the package:

```
8853 \RequirePackage{array}
```

Requires the package:

```
8854 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8855 \@ifundefined{glsdescwidth}{%
8856 \newlength\glsdescwidth
8857 \setlength{\glsdescwidth}{0.6\hsize}
8858 }{}

```

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8859 \@ifundefined{glspagelistwidth}{%
8860 \newlength\glspagelistwidth
8861 \setlength{\glspagelistwidth}{0.1\hsize}
8862 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8863 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8864 \renewenvironment{theglossary}%
8865 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
8866 {\end{longtable}}%

```

Do nothing at the start of the environment:

```
8867 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8868 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8869 \renewcommand{\glossentry}[2]{%
8870   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8871   \glossentrydesc{##1}\glspostdescription\space ##2%
8872   \tabularnewline
8873 }%
```

Sub entries displayed on the following row without the name:

```
8874 \renewcommand{\subglossentry}[3]{%
8875   &
8876   \glssubentryitem{##2}%
8877   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8878   \glspostdescription\space ##3%
8879   \tabularnewline
8880 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip`
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8881 \ifglsgroupskip
8882 \renewcommand*{\glsgroupskip}{}%
8883 \else
8884 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8885 \fi
8886 }
```

`ongraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8887 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8888 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8889 \renewenvironment{theglossary}{%
8890   \begin{longtable}{|l|>{\raggedright}p{\glsglwdwidth}|}%
8891   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8892 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8893 }
```

`ongraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8894 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8895 \setglossarystyle{longragged}%
```

Set the table's header:

```
8896 \renewcommand*{\glossaryheader}{%
8897   \bfseries \entryname & \bfseries \descriptionname
```

```

8898 \tabularnewline\endhead}%
8899 }

```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```

8900 \newglossarystyle{longraggedheaderborder}{%
    Base it on the glostylelongraggedborder style:
8901 \setglossarystyle{longraggedborder}%
    Set the table's header and add horizontal line to table's foot:
8902 \renewcommand*{\glossaryheader}{%
8903 \hline\bfseries \entryname & \bfseries \descriptionname
8904 \tabularnewline\hline
8905 \endhead
8906 \hline\endfoot}%
8907 }

```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```

8908 \newglossarystyle{longragged3col}{%
    Use a longtable with 3 columns:
8909 \renewenvironment{theglossary}%
8910 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8911 >{\raggedright}p{\glspagelistwidth}}}%
8912 {\end{longtable}}%

```

No table header:

```

8913 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```

8914 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8915 \renewcommand{\glossentry}[2]{%
8916 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8917 \glossentrydesc{##1} & ##2\tabularnewline
8918 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8919 \renewcommand{\subglossentry}[3]{%
8920 &
8921 \glssubentryitem{##2}%
8922 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8923 ##3\tabularnewline
8924 }%

```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

8925 \ifglsnogroupskip
8926 \renewcommand*{\glsgroupskip}{}%

```

```

8927 \else
8928 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8929 \fi
8930 }

```

ragged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```

8931 \newglossarystyle{longragged3colborder}{%
    Base it on the glostylelongragged3col style:
8932 \setglossarystyle{longragged3col}%
    Use a longtable with 3 columns with vertical lines around them:
8933 \renewenvironment{theglossary}%
8934 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%
8935 >{\raggedright}p{\glspagelistwidth}|}%
8936 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8937 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8938 }

```

ragged3colheader The longragged3colheader style is like longragged3col but with a header row:

```

8939 \newglossarystyle{longragged3colheader}{%
    Base it on the glostylelongragged3col style:
8940 \setglossarystyle{longragged3col}%
    Set the table's header:
8941 \renewcommand*{\glossaryheader}{%
8942 \bfseries\entryname&\bfseries\descriptionname&
8943 \bfseries\pagelistname\tabularnewline\endhead}%
8944 }

```

colheaderborder The longragged3colheaderborder style is like the above but with a border

```

8945 \newglossarystyle{longragged3colheaderborder}{%
    Base it on the glostylelongragged3colborder style:
8946 \setglossarystyle{longragged3colborder}%
    Set the table's header and add horizontal line at table's foot:
8947 \renewcommand*{\glossaryheader}{%
8948 \hline
8949 \bfseries\entryname&\bfseries\descriptionname&
8950 \bfseries\pagelistname\tabularnewline\hline\endhead
8951 \hline\endfoot}%
8952 }

```

altlongragged4col The altlongragged4col style is like the altlong4col style defined in the package, except that ragged right formatting is used for the description and page list columns.

```

8953 \newglossarystyle{altlongragged4col}{%

```


Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8954 \renewenvironment{theglossary}%
8955   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8956     >{\raggedright}p{\glspagelistwidth}}}%
8957   {\end{longtable}}%
```

No table header:

```
8958 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8959 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a single row (name in first column, description in second column, symbol in third column, page list in last column):

```
8960 \renewcommand{\glossentry}[2]{%
8961   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8962   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8963   ##2\tabularnewline
8964 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8965 \renewcommand{\subglossentry}[3]{%
8966   &
8967   \glssubentryitem{##2}%
8968   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8969   \glossentrysymbol{##2} & ##3\tabularnewline
8970 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8971 \ifglsgroupskip
8972   \renewcommand*{\glsgroupskip}{}%
8973 \else
8974   \renewcommand*{\glsgroupskip}{& & & \tabularnewline}%
8975 \fi
8976 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8977 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8978 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8979 \renewenvironment{theglossary}%
8980   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8981     >{\raggedright}p{\glspagelistwidth}}}%
8982   {\end{longtable}}%
```

Table has a header:

```
8983 \renewcommand*{\glossaryheader}{%
8984 \bfseries\entryname&\bfseries\descriptionname&
8985 \bfseries \symbolname&
8986 \bfseries\pagelistname\tabularnewline\endhead}%
8987 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8988 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8989 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8990 \renewenvironment{theglossary}%
8991 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8992 >{\raggedright}p{\glspagelistwidth}|}%
8993 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8994 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8995 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8996 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8997 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8998 \renewenvironment{theglossary}%
8999 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
9000 >{\raggedright}p{\glspagelistwidth}|}%
9001 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
9002 \renewcommand*{\glossaryheader}{%
9003 \hline\bfseries\entryname&\bfseries\descriptionname&
9004 \bfseries \symbolname&
9005 \bfseries\pagelistname\tabularnewline\hline\endhead
9006 \hline\endfoot}%
9007 }
```

3.7 Glossary Styles using multicol (glossary-mcols.sty)

The style file defines glossary styles that use the `multicol` package. These use the tree-like glossary styles in a `multicol` environment.

```
9008 \ProvidesPackage{glossary-mcols}[2020/02/13 v4.45 (NLCT)]
```

Required packages:

```
9009 \RequirePackage{multicol}
9010 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
9011 \providecommand{\indexspace}{%
9012   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9013 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
9014 \newcommand*{\glsmcols}{2}
```

`mcolindex` Multi-column index style. Same as the index, but puts the glossary in multiple columns. (Ideally the glossary title should go in the optional argument of multicols, but the title isn't part of the glossary style.)

```
9015 \newglossarystyle{mcolindex}{%
9016   \setglossarystyle{index}%
9017   \renewenvironment{theglossary}%
9018     {%
9019       \begin{multicols}{\glsmcols}
9020       \setlength{\parindent}{0pt}%
9021       \setlength{\parskip}{0pt plus 0.3pt}%
9022       \let\item\glstreeitem
9023       \let\subitem\glstreesubitem
9024       \let\subsubitem\glstreesubsubitem
9025     }%
9026     {\end{multicols}}}%
9027 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
9028 \newglossarystyle{mcolindexgroup}{%
9029   \setglossarystyle{mcolindex}%
9030   \renewcommand*{\glsgroupheading}[1]{%
9031     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
9032 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
9033 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
9034   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
9035   \renewcommand*{\glossaryheader}{%
9036     \item\glstreenavigationfmt{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
9037 \renewcommand*{\glsgroupheading}[1]{%
9038   \item\glstreegroupheaderfmt
9039     {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
9040   \indexspace}%
9041 }
```

colindexspannav Similar to **colindexhypergroup**, but puts the navigation line in the optional argument of **multicols**.

```
9042 \newglossarystyle{colindexspannav}{%
9043   \setglossarystyle{index}%
9044   \renewenvironment{theglossary}%
9045     {%
9046       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
9047       \setlength{\parindent}{0pt}%
9048       \setlength{\parskip}{0pt plus 0.3pt}%
9049       \let\item\glstreeitem}%
9050     {\end{multicols}}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
9051 \renewcommand*{\glsgroupheading}[1]{%
9052   \item\glstreegroupheaderfmt
9053     {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
9054   \indexspace}%
9055 }
```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
9056 \newglossarystyle{mcoltree}{%
9057   \setglossarystyle{tree}%
9058   \renewenvironment{theglossary}%
9059     {%
9060       \begin{multicols}{\glsmcols}
9061       \setlength{\parindent}{0pt}%
9062       \setlength{\parskip}{0pt plus 0.3pt}%
9063     }%
9064     {\end{multicols}}%
9065 }
```

mcoltreegroup Like the **mcoltree** style but the glossary groups have headings.

```
9066 \newglossarystyle{mcoltreegroup}{%
  Base it on the glostylemcoltree style:
9067   \setglossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9068 \renewcommand{\glsgroupheading}[1]{\par
9069 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
9070 }
```

ltreehypergroup The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
9071 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the `glostylemcoltree` style:

```
9072 \setglossarystyle{mcoltree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9073 \renewcommand*{\glossaryheader}{%
9074 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9075 \renewcommand*{\glsgroupheading}[1]{%
9076 \par\noindent
9077 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9078 \indexspace}%
9079 }
```

mcoltreespannav Similar to the `mcoltreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
9080 \newglossarystyle{mcoltreespannav}{%
9081 \setglossarystyle{tree}%
9082 \renewenvironment{theglossary}%
9083 {%
9084 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
9085 \setlength{\parindent}{0pt}%
9086 \setlength{\parskip}{0pt plus 0.3pt}%
9087 }%
9088 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9089 \renewcommand*{\glsgroupheading}[1]{%
9090 \par\noindent
9091 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9092 \indexspace}%
9093 }
```

mcoltreenoname Multi-column index style. Same as the `treenoname`, but puts the glossary in multiple columns.

```
9094 \newglossarystyle{mcoltreenoname}{%
9095 \setglossarystyle{treenoname}%
9096 \renewenvironment{theglossary}%
9097 {%
```

```

9098     \begin{multicols}{\glsmcols}
9099     \setlength{\parindent}{0pt}%
9100     \setlength{\parskip}{0pt plus 0.3pt}%
9101 }%
9102 {\end{multicols}}%
9103 }

```

treenonamegroup Like the `mcoltreenoname` style but the glossary groups have headings.

```

9104 \newglossarystyle{mcoltreenonamegroup}{%
    Base it on the glostylemcoltreenoname style:
9105   \setglossarystyle{mcoltreenoname}%
    Give each group a heading:
9106   \renewcommand{\glsgroupheading}[1]{\par
9107     \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
9108 }

```

onamehypergroup The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9109 \newglossarystyle{mcoltreenonamehypergroup}{%
    Base it on the glostylemcoltreenoname style:
9110   \setglossarystyle{mcoltreenoname}%
    Put navigation links to the groups at the start of the theglossary environment:
9111   \renewcommand*{\glossaryheader}{%
9112     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
9113   \renewcommand*{\glsgroupheading}[1]{%
9114     \par\noindent
9115     \glstreegroupheaderfmt{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
9116     \indexspace}%
9117 }

```

eenonamespannav Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

9118 \newglossarystyle{mcoltreenonamespannav}{%
9119   \setglossarystyle{treenoname}%
9120   \renewenvironment{theglossary}%
9121   {%
9122     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
9123     \setlength{\parindent}{0pt}%
9124     \setlength{\parskip}{0pt plus 0.3pt}%
9125   }%
9126   {\end{multicols}}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
9127   \renewcommand*{\glsgroupheading}[1]{%
9128     \par\noindent

```

```

9129 \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9130 \indexspace}%
9131 }

```

mcolalttree Multi-column index style. Same as the alttree, but puts the glossary in multiple columns.

```

9132 \newglossarystyle{mcolalttree}{%
9133 \setglossarystyle{alttree}%
9134 \renewenvironment{theglossary}%
9135 {%
9136 \begin{multicols}{\glsmcols}
9137 \def\@gls@prevlevel{-1}%
9138 \mbox{}\par
9139 }%
9140 {\par\end{multicols}}}%
9141 }

```

colalttreegroup Like the mcolalttree style but the glossary groups have headings.

```

9142 \newglossarystyle{colalttreegroup}{%
    Base it on the glostylemcolalttree style:
9143 \setglossarystyle{mcolalttree}%
    Give each group a heading.
9144 \renewcommand{\glsgroupheading}[1]{\par
9145 \def\@gls@prevlevel{-1}%
9146 \hangindent0pt\relax
9147 \parindent0pt\relax
9148 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
9149 }

```

treehypergroup The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9150 \newglossarystyle{mcolalttreehypergroup}{%
    Base it on the glostylemcolalttree style:
9151 \setglossarystyle{mcolalttree}%
    Put the navigation links in the header
9152 \renewcommand*{\glossaryheader}{%
9153 \par
9154 \def\@gls@prevlevel{-1}%
9155 \hangindent0pt\relax
9156 \parindent0pt\relax
9157 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9158 \renewcommand*{\glsgroupheading}[1]{%
9159 \par
9160 \def\@gls@prevlevel{-1}%
9161 \hangindent0pt\relax

```

```

9162 \parindent0pt\relax
9163 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
9164 \indexspace}%
9165 }

```

`lalttreespannav` Similar to the `mcolalttreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

9166 \newglossarystyle{mcolalttreespannav}{%
9167 \setglossarystyle{alttree}%
9168 \renewenvironment{theglossary}%
9169 {%
9170 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
9171 \def\@gls@prevlevel{-1}%
9172 \mbox{}\par
9173 }%
9174 {\par\end{multicols}}}%

```

Put a hypertarget at the start of each group

```

9175 \renewcommand*{\glsgroupheading}[1]{%
9176 \par
9177 \def\@gls@prevlevel{-1}%
9178 \hangindent0pt\relax
9179 \parindent0pt\relax
9180 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
9181 \indexspace}%
9182 }

```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the `supertabular` environment.

```

9183 \ProvidesPackage{glossary-super}[2020/02/13 v4.45 (NLCT)]

```

Requires the package:

```

9184 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if `has` has been loaded.

```

9185 \@ifundefined{glsdescwidth}{%
9186 \newlength{glsdescwidth}
9187 \setlength{glsdescwidth}{0.6\hsize}
9188 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if `has` has been loaded.

```

9189 \@ifundefined{glspagelistwidth}{%
9190 \newlength{glspagelistwidth}
9191 \setlength{glspagelistwidth}{0.1\hsize}

```


9192 }{}

super The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

9193 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

9194 \renewenvironment{theglossary}{%

9195 {\tablehead{}\tabletail{}}%

9196 \begin{supertabular}{lp{\glsdescwidth}}%

9197 {\end{supertabular}}%

Do nothing at the start of the table:

9198 \renewcommand*{\glossaryheader}{}%

No group headings:

9199 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

9200 \renewcommand{\glossentry}[2]{%

9201 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

9202 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline

9203 }%

Sub entries put in a row (no name, description and page list in second column):

9204 \renewcommand{\subglossentry}[3]{%

9205 &

9206 \glssubentryitem{##2}%

9207 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space

9208 ##3\tabularnewline

9209 }%

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

9210 \ifglsnogroupskip

9211 \renewcommand*{\glsgroupskip}{}%

9212 \else

9213 \renewcommand*{\glsgroupskip}{& \tabularnewline}%

9214 \fi

9215 }

superborder The superborder style is like the above, but with horizontal and vertical lines:

9216 \newglossarystyle{superborder}{%

Base it on the glostylesuper style:

9217 \setglossarystyle{super}%

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

9218 \renewenvironment{theglossary}{%

9219 {\tablehead{\hline}\tabletail{\hline}%

```

9220     \begin{supertabular}{|l|p{\glsdescwidth}|}%
9221     {\end{supertabular}}%
9222 }

```

superheader The superheader style is like the super style, but with a header:

```

9223 \newglossarystyle{superheader}{%
    Base it on the glostylesuper style:
9224     \setglossarystyle{super}%
    Put the glossary in a supertabular environment with two columns, a header and no tail:
9225 \renewenvironment{theglossary}%
9226     {\tablehead{\bfseries \entryname &
9227     \bfseries\descriptionname\tabularnewline}%
9228     \tabletail{}}%
9229     \begin{supertabular}{lp{\glsdescwidth}}%
9230     {\end{supertabular}}%
9231 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```

9232 \newglossarystyle{superheaderborder}{%
    Base it on the glostylesuper style:
9233     \setglossarystyle{super}%
    Put the glossary in a supertabular environment with two columns, a header and horizontal
    lines above and below the table:
9234     \renewenvironment{theglossary}%
9235     {\tablehead{\hline\bfseries \entryname &
9236     \bfseries \descriptionname\tabularnewline\hline}%
9237     \tabletail{\hline}
9238     \begin{supertabular}{|l|p{\glsdescwidth}|}%
9239     {\end{supertabular}}%
9240 }

```

super3col The super3col style is like the super style, but with 3 columns:

```

9241 \newglossarystyle{super3col}{%
    Put the glossary in a supertabular environment with three columns and no head or tail:
9242     \renewenvironment{theglossary}%
9243     {\tablehead{}\tabletail{}}%
9244     \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
9245     {\end{supertabular}}%
    Do nothing at the start of the table:
9246     \renewcommand*{\glossaryheader}{}%
    No group headings:
9247     \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

9248 \renewcommand{\glossentry}[2]{%
9249   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9250   \glossentrydesc{##1} & ##2\tabularnewline
9251 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```

9252 \renewcommand{\subglossentry}[3]{%
9253   &
9254   \glssubentryitem{##2}%
9255   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9256   ##3\tabularnewline
9257 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

9258 \ifglsgroupskip
9259   \renewcommand*{\glsgroupskip}{}%
9260 \else
9261   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9262 \fi
9263 }
```

super3colborder The super3colborder style is like the super3col style, but with a border:

```

9264 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```

9265 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9266 \renewenvironment{theglossary}%
9267   {\tablehead{\hline}\tabletail{\hline}%
9268   \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth|}}%
9269   {\end{supertabular}}%
9270 }
```

super3colheader The super3colheader style is like the super3col style but with a header row:

```

9271 \newglossarystyle{super3colheader}{%
```

Base it on the glostylesuper3col style:

```

9272 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9273 \renewenvironment{theglossary}%
9274   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9275   \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9276   \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
9277   {\end{supertabular}}%
9278 }
```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
9279 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
9280 \setglossarystyle{super3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9281 \renewenvironment{theglossary}{%
9282   {\tablehead{\hline
9283     \bfseries\entryname&\bfseries\descriptionname&
9284     \bfseries\pagelistname\tabularnewline\hline}%
9285   \tabletail{\hline}%
9286   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
9287   {\end{supertabular}}}%
9288 }
```

super4col The super4col glossary style has four columns, where the third column contains the value of the corresponding symbol key used when that entry was defined.

```
9289 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9290 \renewenvironment{theglossary}{%
9291   {\tablehead{}\tabletail{}}%
9292   \begin{supertabular}{|l|l|l|l|}%
9293   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9294 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9295 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row with the name in the first column, description in second column, symbol in third column and page list in last column:

```
9296 \renewcommand{\glossentry}[2]{%
9297   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9298   \glossentrydesc{##1} &
9299   \glossentrysymbol{##1} & ##2\tabularnewline
9300   }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9301 \renewcommand{\subglossentry}[3]{%
9302   &
9303   \glssubentryitem{##2}%
9304   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9305   \glossentrysymbol{##2} & ##3\tabularnewline
9306   }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
9307 \ifglsnogroupskip
9308 \renewcommand*{\glsgroupskip}{}%
9309 \else
9310 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9311 \fi
9312 }
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
9313 \newglossarystyle{super4colheader}{%
  Base it on the glostylesuper4col style:
9314 \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns, a header and no tail:
9315 \renewenvironment{theglossary}%
9316 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9317 \bfseries\symbolname &
9318 \bfseries\pagelistname\tabularnewline}%
9319 \tabletail{}}%
9320 \begin{supertabular}{1111}}%
9321 {\end{supertabular}}%
9322 }
```

super4colborder The super4colborder style is like the super4col but with a border.

```
9323 \newglossarystyle{super4colborder}{%
  Base it on the glostylesuper4col style:
9324 \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns and a horizontal line in the
  head and tail:
9325 \renewenvironment{theglossary}%
9326 {\tablehead{\hline}\tabletail{\hline}%
9327 \begin{supertabular}{11111111}}%
9328 {\end{supertabular}}%
9329 }
```

colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```
9330 \newglossarystyle{super4colheaderborder}{%
  Base it on the glostylesuper4col style:
9331 \setglossarystyle{super4col}%
  Put the glossary in a supertabular environment with four columns and a header bordered by
  horizontal lines and a horizontal line in the tail:
9332 \renewenvironment{theglossary}%
9333 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
9334 \bfseries\symbolname &
```

```

9335      \bfseries\pagelistname\tabularnewline\hline}%
9336      \tabletail{\hline}%
9337      \begin{supertabular}{|l|l|l|l|}%
9338      {\end{supertabular}}}%
9339 }

```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```

9340 \newglossarystyle{altsuper4col}{%
      Base it on the glostylesuper4col style:
9341   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and no head or tail:
9342   \renewenvironment{theglossary}%
9343   {\tablehead{}\tabletail{}}%
9344   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
9345   {\end{supertabular}}}%
9346 }

```

super4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```

9347 \newglossarystyle{altsuper4colheader}{%
      Base it on the glostylesuper4colheader style:
9348   \setglossarystyle{super4colheader}%
      Put the glossary in a supertabular environment with four columns, a header and no tail:
9349   \renewenvironment{theglossary}%
9350   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9351     \bfseries\symbolname &
9352     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9353   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
9354   {\end{supertabular}}}%
9355 }

```

super4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```

9356 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
9357   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a horizontal line in the
      head and tail:
9358   \renewenvironment{theglossary}%
9359   {\tablehead{\hline}\tabletail{\hline}%
9360   \begin{supertabular}%
9361     {|l|lp{\glsgdescwidth}|l|lp{\glspagelistwidth}|}%
9362     {\end{supertabular}}}%
9363 }

```

colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```

9364 \newglossarystyle{altsuper4colheaderborder}{%

```

Base it on the `glostylesuper4colheaderborder` style:

```
9365 \setglossarystyle{super4colheaderborder}%
```

Put the glossary in a `supertabular` environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```
9366 \renewenvironment{theglossary}%
9367   {\tablehead{\hline
9368     \bfseries\entryname &
9369     \bfseries\descriptionname &
9370     \bfseries\symbolname &
9371     \bfseries\pagelistname\tabularnewline\hline}%
9372   \tabletail{\hline}%
9373   \begin{supertabular}%
9374     {lllp{\glsdescwidth}llp{\glspagelistwidth}}}%
9375   {\end{supertabular}}%
9376 }
```

3.9 Glossary Styles using `supertabular` environment (`glossary-superragged` package)

The glossary styles defined in the package use the `supertabular` environment. These styles are like those provided by the package, except that the multiline columns have ragged right justification.

```
9377 \ProvidesPackage{glossary-superragged}[2020/02/13 v4.45 (NLCT)]
```

Requires the package:

```
9378 \RequirePackage{array}
```

Requires the package:

```
9379 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
9380 \@ifundefined{glsdescwidth}{%
9381   \newlength\glsdescwidth
9382   \setlength{\glsdescwidth}{0.6\hsize}
9383 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
9384 \@ifundefined{glspagelistwidth}{%
9385   \newlength\glspagelistwidth
9386   \setlength{\glspagelistwidth}{0.1\hsize}
9387 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
9388 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9389 \renewenvironment{theglossary}%
9390   {\tablehead{}\tabletail{}}%
9391   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9392   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9393 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9394 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9395 \renewcommand{\glossentry}[2]{%
9396   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9397   \glossentrydesc{##1}\glspostdescription\space ##2%
9398   \tabularnewline
9399 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9400 \renewcommand{\subglossentry}[3]{%
9401   &
9402   \glssubentryitem{##2}%
9403   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9404   ##3%
9405   \tabularnewline
9406 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
9407 \ifglsnogroupskip
9408   \renewcommand*{\glsgroupskip}{}%
9409 \else
9410   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9411 \fi
9412 }
```

`\perraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
9413 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylesuperragged` style:

```
9414 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9415 \renewenvironment{theglossary}%
9416   {\tablehead{\hline}\tabletail{\hline}%
9417   \begin{supertabular}{l|>{\raggedright}p{\glsgdescwidth}|}%
9418   {\end{supertabular}}%
9419 }
```


`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
9420 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
9421 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
9422 \renewenvironment{theglossary}%
```

```
9423 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
```

```
9424 \tabularnewline}%
```

```
9425 \tabletail{}}%
```

```
9426 \begin{supertabular}{l>{\raggedright}p{\glstdescwidth}}%
```

```
9427 {\end{supertabular}}%
```

```
9428 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
9429 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
9430 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
9431 \renewenvironment{theglossary}%
```

```
9432 {\tablehead{\hline\bfseries \entryname &
```

```
9433 \bfseries \descriptionname\tablearnewline\hline}%
```

```
9434 \tabletail{\hline}
```

```
9435 \begin{supertabular}{ll|>{\raggedright}p{\glstdescwidth}}%
```

```
9436 {\end{supertabular}}%
```

```
9437 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
9438 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
9439 \renewenvironment{theglossary}%
```

```
9440 {\tablehead{}\tabletail{}}%
```

```
9441 \begin{supertabular}{l>{\raggedright}p{\glstdescwidth}%
```

```
9442 >{\raggedright}p{\glspagelistwidth}}%
```

```
9443 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9444 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9445 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9446 \renewcommand{\glossentry}[2]{%
```

```
9447 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9448 \glossentrydesc{##1} &
```

```

9449     ##2\tabularnewline
9450 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9451 \renewcommand{\subglossentry}[3]{%
9452     &
9453     \glssubentryitem{##2}%
9454     \glstarget{##2}{\strut}\glossentrydesc{##2} &
9455     ##3\tabularnewline
9456 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

9457 \ifglsnogroupskip
9458 \renewcommand*{\glsgroupskip}{}%
9459 \else
9460 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9461 \fi
9462 }

```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```

9463 \newglossarystyle{superragged3colborder}{%

```

Base it on the glostylesuperragged3col style:

```

9464 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9465 \renewenvironment{theglossary}%
9466 {\tablehead{\hline}\tabletail{\hline}%
9467 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}||%
9468 >{\raggedright}p{\glspagelistwidth}||}%
9469 {\end{supertabular}}%
9470 }

```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```

9471 \newglossarystyle{superragged3colheader}{%

```

Base it on the glostylesuperragged3col style:

```

9472 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9473 \renewenvironment{theglossary}%
9474 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9475 \bfseries\pagelistname\tabularnewline}\tabletail{}%
9476 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9477 >{\raggedright}p{\glspagelistwidth}}%
9478 {\end{supertabular}}%
9479 }

```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9480 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9481 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9482 \renewenvironment{theglossary}{%
9483   {\tablehead{\hline
9484     \bfseries\entryname&\bfseries\descriptionname&
9485     \bfseries\pagelistname\tabularnewline\hline}%
9486   \tabletail{\hline}%
9487   \begin{supertabular}{|l|>{\raggedright}p{\glstdescwidth}|%
9488     >{\raggedright}p{\glspagelistwidth}|}%
9489   {\end{supertabular}}}%
9490 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9491 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9492 \renewenvironment{theglossary}{%
9493   {\tablehead{}\tabletail{}%
9494   \begin{supertabular}{l>{\raggedright}p{\glstdescwidth}l%
9495     >{\raggedright}p{\glspagelistwidth}}}%
9496   {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
9497 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9498 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row with the name in the first column, description in second column, symbol in third column and page list in last column:

```
9499 \renewcommand{\glossentry}[2]{%
9500   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9501   \glossentrydesc{##1} &
9502   \glossentrysymbol{##1} & ##2\tabularnewline
9503   }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9504 \renewcommand{\subglossentry}[3]{%
9505   &
9506   \glssubentryitem{##2}%
9507   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9508   \glossentrysymbol{##2} & ##3\tabularnewline
9509   }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
9510 \ifglsgroupskip
9511 \renewcommand*{\glsgroupskip}{}%
9512 \else
9513 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9514 \fi
9515 }
```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
9516 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
9517 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9518 \renewenvironment{theglossary}%
9519 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9520 \bfseries\symbolname &
9521 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9522 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9523 >{\raggedright}p{\glspagelistwidth}}}%
9524 {\end{supertabular}}}%
9525 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
9526 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9527 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9528 \renewenvironment{theglossary}%
9529 {\tablehead{\hline}\tabletail{\hline}%
9530 \begin{supertabular}%
9531 {l|>{\raggedright}p{\glsgdescwidth}l|}%
9532 >{\raggedright}p{\glspagelistwidth}l}}%
9533 {\end{supertabular}}}%
9534 }
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
9535 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9536 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```

9537 \renewenvironment{theglossary}%
9538   {\tablehead{\hline
9539     \bfseries\entryname &
9540     \bfseries\descriptionname &
9541     \bfseries\symbolname &
9542     \bfseries\pagelistname\tabularnewline\hline}%
9543   \tabletail{\hline}%
9544   \begin{supertabular}%
9545     {||>{\raggedright}p{\glsdescwidth}||}%
9546     >{\raggedright}p{\glspagelistwidth}||}%
9547   {\end{supertabular}}%
9548 }

```

3.10 Tree Styles (glossary-tree.sty)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```

9549 \ProvidesPackage{glossary-tree}[2020/02/13 v4.45 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9550 \providecommand{\indexspace}{%
9551   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9552 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command was previously also used to format the group headings.

```

9553 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}

```

`\glstreegroupheaderfmt` Format used to display the group header in the tree styles. Before v4.22, `\glstreenamefmt` was used for the group header, so the default definition uses that to help maintain backward-compatibility, since in previous versions redefining `\glstreenamefmt` would've also affected the group headings.

```

9554 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9555 \newcommand*{\glstreenavigationfmt}[1]{\glstreenamefmt{#1}}

```

Allow the user to adjust the index style without disturbing the index.

`\glstreeitem` Top level item used in index style.

```

9556 \ifdef\@idxitem
9557 {\newcommand{\glstreeitem}{\@idxitem}}
9558 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```
9559 \ifdef\subitem
9560 {\let\glstreesubitem\subitem}
9561 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

`\glstreesubsubitem` Level 1 item used in index style.

```
9562 \ifdef\subsubitem
9563 {\let\glstreesubsubitem\subsubitem}
9564 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `alttree` style).

```
9565 \newcommand{\glstreepredesc}{\space}
```

`\glstreechildpredesc` Allow the user to adjust the space before the description for sub-entries (except for the `treenoname` and `alttree` style).

```
9566 \newcommand{\glstreechildpredesc}{\space}
```

`index` The index glossary style is similar in style to the way indices are usually typeset using `\item`, `\subitem` and `\subsubitem`. The entry name is set in bold. If an entry has a symbol, it is placed in brackets after the name. Then the description is displayed, followed by the number list. This style allows up to three levels.

```
9567 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9568 \renewenvironment{theglossary}%
9569 {\setlength{\parindent}{0pt}}%
9570 {\setlength{\parskip}{0pt plus 0.3pt}}%
9571 \let\item\glstreeitem
9572 \let\subitem\glstreesubitem
9573 \let\subsubitem\glstreesubsubitem
9574 }%
```

```
9575 {\par}}%
```

Do nothing at the start of the environment:

```
9576 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9577 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entry starts a new item with the name in bold followed by the symbol in brackets (if it exists), the description and the page list.

```
9578 \renewcommand*{\glossentry}[2]{%
9579 \item\glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9580 \ifglshassymbol{##1}{\space\glossentrysymbol{##1}}{\}%
9581 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9582 }%
```

Sub entries: level 1 entries use `\subitem`, levels greater than 1 use `\subsubitem`. The level (`##1`) shouldn't be 0, as that's catered by `\glossentry`, but for completeness, if the level is 0, `\item` is used. The name is put in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```

9583 \renewcommand{\subglossentry}[3]{%
9584   \ifcase##1\relax
9585     % level 0
9586     \item
9587   \or
9588     % level 1
9589     \subitem
9590     \glssubentryitem{##2}%
9591   \else
9592     % all other levels
9593     \subsubitem
9594   \fi
9595   \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}%
9596   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9597   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9598 }%
```

Vertical gap between groups is the same as that used by indices:

```

9599 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The `indexgroup` style is like the `index` style but has headings.

```

9600 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```

9601 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9602 \renewcommand*{\glsgroupheading}[1]{%
9603   \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9604   \indexspace
9605 }%
9606 }
```

indexhypergroup The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

9607 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```

9608 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```

9609 \renewcommand*{\glossaryheader}{%
9610   \item\glstreenavigationfmt{\glsnavigation}\indexspace}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

9611 \renewcommand*{\glsgroupheading}[1]{%
9612   \item\glstreegroupheaderfmt

```

```

9613      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
9614      \indexspace}%
9615 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9616 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9617 \renewenvironment{theglossary}%
9618 {\setlength{\parindent}{0pt}%
9619 \setlength{\parskip}{0pt plus 0.3pt}}%
9620 {}%

```

Do nothing at the start of the theglossary environment:

```

9621 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9622 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

9623 \renewcommand{\glossentry}[2]{%
9624 \hangindent0pt\relax
9625 \parindent0pt\relax
9626 \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9627 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9628 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9629 }%

```

Sub entries: level $\langle n \rangle$ is indented by $\langle n \rangle$ times `\glstreeindent`. The name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```

9630 \renewcommand{\subglossentry}[3]{%
9631 \hangindent##1\glstreeindent\relax
9632 \parindent##1\glstreeindent\relax
9633 \ifnum##1=1\relax
9634 \glssubentryitem{##2}%
9635 \fi
9636 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9637 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9638 \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9639 }%

```

Vertical gap between groups is the same as that used by indices:

```

9640 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

9641 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

9642 \setglossarystyle{tree}%

```


Each group has a heading (in bold) followed by a vertical gap):

```
9643 \renewcommand{\glsgroupheading}[1]{\par
9644 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
9645 \indexspace}%
9646 }
```

treehypergroup The **treehypergroup** style is like the **treegroup** style, but has a set of links to the groups at the start of the glossary.

```
9647 \newglossarystyle{treehypergroup}{%
```

Base it on the **glostyletree** style:

```
9648 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the **theglossary** environment:

```
9649 \renewcommand*{\glossaryheader}{%
9650 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9651 \renewcommand*{\glsgroupheading}[1]{%
9652 \par\noindent
9653 \glstreegroupheaderfmt
9654 {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
9655 \indexspace}%
9656 }
```

\glstreeindent Length governing left indent for each level of the tree style.

```
9657 \newlength\glstreeindent
9658 \setlength{\glstreeindent}{10pt}
```

treenoname The **treenoname** glossary style is like the **tree** style, but doesn't print the name or symbol for sub-levels.

```
9659 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9660 \renewenvironment{theglossary}%
9661 {\setlength{\parindent}{0pt}%
9662 \setlength{\parskip}{0pt plus 0.3pt}}%
9663 {}%
```

No header:

```
9664 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9665 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: the name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
9666 \renewcommand{\glossentry}[2]{%
9667 \hangindent0pt\relax
9668 \parindent0pt\relax
9669 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```

9670 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9671 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9672 }%

```

Sub entries: level $\langle n \rangle$ is indented by $\langle n \rangle$ times `\glstreeindent`. The name and symbol are omitted. The description followed by the page list are displayed.

```

9673 \renewcommand{\subglossentry}[3]{%
9674 \hangindent##1\glstreeindent\relax
9675 \parindent##1\glstreeindent\relax
9676 \ifnum##1=1\relax
9677 \glssubentryitem{##2}%
9678 \fi
9679 \glstarget{##2}{\strut}%
9680 \glossentrydesc{##2}\glspostdescription\space##3\par
9681 }%

```

Vertical gap between groups is the same as that used by indices:

```

9682 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9683 }

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```

9684 \newglossarystyle{treenonamegroup}{%

```

Base it on the `glostyletreenoname` style:

```

9685 \setglossarystyle{treenoname}%

```

Give each group a heading:

```

9686 \renewcommand{\glsgroupheading}[1]{\par
9687 \noindent\glstreegroupheaderfmt
9688 {\glsgrouptitle{##1}}\par\indexspace}%
9689 }

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9690 \newglossarystyle{treenonamehypergroup}{%

```

Base it on the `glostyletreenoname` style:

```

9691 \setglossarystyle{treenoname}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

9692 \renewcommand*{\glossaryheader}{%
9693 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9694 \renewcommand*{\glsgroupheading}[1]{%
9695 \par\noindent
9696 \glstreegroupheaderfmt
9697 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9698 \indexspace}%
9699 }

```

`esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```

9700 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9701   \dimen@=0pt\relax
9702   \gls@tmplen=0pt\relax
9703   \forallglossaries[#1]{\@gls@type}%
9704   {%
9705     \forallglsentries[\@gls@type]{\@glo@label}%
9706     {%
9707       \ifglsahasparent{\@glo@label}%
9708       }%
9709     {%
9710       \settowidth{\dimen@}%
9711         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
9712       \ifdim\dimen@>\gls@tmplen
9713         \gls@tmplen=\dimen@
9714       \letcs{\@glswidestname}{glo\glsdetoklabel{\@glo@label}@name}%
9715       \fi
9716     }%
9717   }%
9718 }%
9719 }
```

`\glssetwidest` `\glssetwidest[⟨level⟩]{⟨text⟩}` sets the widest text for the given level. It is used by the alt-tree glossary styles to determine the indentation of each level.

```

9720 \newcommand*{\glssetwidest}[2][0]{%
9721   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9722     #2}%
9723 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```

9724 \newcommand*{\@glswidestname}{}
```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9725 \newcommand*{\glstreenamebox}[2]{%
9726   \makebox[#1][l]{#2}%
9727 }
```

`alttree` The alttree glossary style is similar in style to the tree style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glssetwidest`.

```

9728 \newglossarystyle{alttree}{%
```

Redefine theglossary environment.

```

9729 \renewenvironment{theglossary}%
9730   {\def\@gls@prevlevel{-1}%
9731    \mbox{}\par}%
9732   {\par}%

```

Set the header and group headers to nothing.

```

9733 \renewcommand*{\glossaryheader}{}%
9734 \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```
9735 \renewcommand{\glossentry}[2]{%
9736 \ifnum\@gls@prevlevel=0\relax
9737 \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9738 \settowidth{\glstreeindent}{\glstreenamfmt{\@glswidestname\space}}%
9739 \fi
```

Set the hangindent and paragraph indent.

```
9740 \hangindent\glstreeindent
9741 \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9742 \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
9743 \glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9744 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9745 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9746 \def\@gls@prevlevel{0}%
9747 }%
```

Redefine the way sub-entries are displayed.

```
9748 \renewcommand{\subglossentry}[3]{%
```

Increment and display the sub-entry counter if this is a level 1 entry and the sub-entry counter is in use.

```
9749 \ifnum##1=1\relax
9750 \glssubentryitem{##2}%
9751 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9752 \ifnum\@gls@prevlevel=##1\relax
9753 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9754 \@ifundefined{@glswidestname\romannumeral##1}{%
9755 \settowidth{\gls@tmplen}{\glstreenamfmt{\@glswidestname\space}}}%
9756 \settowidth{\gls@tmplen}{\glstreenamfmt{%
9757 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9758 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9759      \setlength\glstreeindent\gls@tmplen
9760      \addtolength\glstreeindent\parindent
9761      \parindent\glstreeindent
9762      \else
```

Depth has decreased, so subtract width of the widest entry from the previous level to `\glstreeindent`. First determine the width of the widest entry for the previous level and store in `\glstreeindent`.

```
9763      \ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9764      \settowidth{\glstreeindent}{\glstreenamfmt{%
9765      \@glswidestname\space}}}{%
9766      \settowidth{\glstreeindent}{\glstreenamfmt{%
9767      \csname @glswidestname\romannumeral\@gls@prevlevel
9768      \endcsname\space}}}{%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9769      \addtolength\parindent{-\glstreeindent}%
9770      \setlength\glstreeindent\parindent
9771      \fi
9772      \fi
```

Set the hanging indentation.

```
9773      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9774      \makebox[0pt][r]{\glstreenambox{\gls@tmplen}{%
9775      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9776      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9777      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9778      \def\@gls@prevlevel{##1}%
9779      }%
```

Vertical gap between groups is the same as that used by indices:

```
9780      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9781 }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
9782 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
9783 \setglossarystyle{almtree}%
```

Give each group a heading.

```
9784 \renewcommand{\glsgroupheading}[1]{\par
9785 \def\@gls@prevlevel{-1}%
9786 \hangindent0pt\relax
```

```

9787     \parindent0pt\relax
9788     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
9789     \par\indexspace}%
9790 }

```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9791 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9792   \setglossarystyle{alttree}%
    Put the navigation links in the header
9793   \renewcommand*{\glossaryheader}{%
9794     \par
9795     \def\@gls@prevlevel{-1}%
9796     \hangindent0pt\relax
9797     \parindent0pt\relax
9798     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9799   \renewcommand*{\glsgroupheading}[1]{%
9800     \par
9801     \def\@gls@prevlevel{-1}%
9802     \hangindent0pt\relax
9803     \parindent0pt\relax
9804     \glstreegroupheaderfmt
9805     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9806     \indexspace}}

```

4 Backwards Compatibility

4.1 glossaries-compatible-207

Provides compatibility with version 2.07 and below. This uses original glossaries xindy and makeindex formatting, so can be used with old documents that had customized style files, but hyperlinks may not work properly.

```
9807 \NeedsTeXFormat{LaTeX2e}
9808 \ProvidesPackage{glossaries-compatible-207}[2020/02/13 v4.45 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9809 \ifglxsindy
9810   \renewcommand*\GlsAddXdyAttribute[1]{%
9811     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9812     \expandafter\toks@\expandafter{\@xdylocref}%
9813     \edef\@xdylocref{\the\toks@ ^^J}%
9814     (markup-locref
9815     :open \string"\string~n\string\setentrycounter
9816       {\noexpand\glscounter}%
9817       \expandafter\string\csname#1\endcsname
9818       \expandafter\@gobble\string\{\string" ^^J
9819       :close \string"\expandafter\@gobble\string\}\string" ^^J
9820       :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
9821 \fi
```

sAddXdyCounters

```
9822 \renewcommand*\GlsAddXdyCounters[1]{%
9823   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9824     in compatibility mode.}%
9825 }
```

Add predefined attributes

```
9826 \GlsAddXdyAttribute{glsnumberformat}
9827 \GlsAddXdyAttribute{textrm}
9828 \GlsAddXdyAttribute{textsf}
9829 \GlsAddXdyAttribute{texttt}
9830 \GlsAddXdyAttribute{textbf}
9831 \GlsAddXdyAttribute{textmd}
9832 \GlsAddXdyAttribute{textit}
9833 \GlsAddXdyAttribute{textup}
9834 \GlsAddXdyAttribute{textsl}
```

```

9835 \GlsAddXdyAttribute{textsc}
9836 \GlsAddXdyAttribute{emph}
9837 \GlsAddXdyAttribute{glshypernumber}
9838 \GlsAddXdyAttribute{hyperrm}
9839 \GlsAddXdyAttribute{hypersf}
9840 \GlsAddXdyAttribute{hypertt}
9841 \GlsAddXdyAttribute{hyperbf}
9842 \GlsAddXdyAttribute{hypermd}
9843 \GlsAddXdyAttribute{hyperit}
9844 \GlsAddXdyAttribute{hyperup}
9845 \GlsAddXdyAttribute{hypersl}
9846 \GlsAddXdyAttribute{hypersc}
9847 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9848 \ifglxindy
9849 \renewcommand*{\GlsAddXdyLocation}[2]{%
9850 \edef\@xdyuserlocationdefs{%
9851 \@xdyuserlocationdefs ^^J%
9852 (define-location-class \string"#1\string"^^J\space\space
9853 \space(#2))
9854 }%
9855 \edef\@xdyuserlocationnames{%
9856 \@xdyuserlocationnames^^J\space\space\space
9857 \string"#1\string"}%
9858 }
9859 \fi

```

\@do@wrglossary

```

9860 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9861 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9862 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9863 \def\@glo@range{}%
9864 \expandafter\if\@glo@prefix(\relax
9865 \def\@glo@range{:open-range}%
9866 \else
9867 \expandafter\if\@glo@prefix)\relax
9868 \def\@glo@range{:close-range}%
9869 \fi
9870 \fi

  Get the location and escape any special characters
9871 \protected@edef\@glslocref{\theglsentrycounter}%
9872 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9873 \glossary[\csname glo@#1@type\endcsname]{%

```



```

9874 (indexentry :tkey (\csname glo@#1@index\endcsname)
9875 :locoref \string"\@glslocoref\string" %
9876 :attr \string"\@glo@suffix\string" \@glo@range
9877 )
9878 }%
9879 \else

```

Convert the format information into the format required for makeindex

```

9880 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9881 \glossary[\csname glo@#1@type\endcsname]{%
9882 \string\glossaryentry{\csname glo@#1@index\endcsname
9883 \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9884 \fi
9885 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9886 \def\@set@glo@numformat#1#2#3{%
9887 \expandafter\@glo@check@mkidxrangechar#3\@nil
9888 \protected@edef#1{%
9889 \@glo@prefix setentrycounter[]{#2}%
9890 \expandafter\string\csname\@glo@suffix\endcsname
9891 }%
9892 \@gls@checkmkidxchars#1%
9893 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9894 \ifglxindy
9895 \def\writeist{%
9896 \openout\glswrite=\istfilename
9897 \write\glswrite{;; xindy style file created by the glossaries
9898 package in compatible-2.07 mode}%
9899 \write\glswrite{;; for document '\jobname' on
9900 \the\year-\the\month-\the\day}%
9901 \write\glswrite{^^J; required styles^^J}
9902 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9903 \ifx\@xdystyle\@empty
9904 \else
9905 \protected@write\glswrite{{(require
9906 \string"\@xdystyle.xdy\string")}}%
9907 \fi
9908 }%
9909 \write\glswrite{^^J%
9910 ; list of allowed attributes (number formats)^^J}%
9911 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9912 \write\glswrite{^^J; user defined alphabets^^J}%
9913 \write\glswrite{\@xdyuseralphabets}%
9914 \write\glswrite{^^J; location class definitions^^J}%
9915 \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9916     \string"roman-numbers-lowercase\string" :sep \string"}}%
9917 \@onelevel@sanitize\@gls@roman
9918 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9919     :sep \string"}}%
9920 \@onelevel@sanitize\@tmp
9921 \ifx\@tmp\@gls@roman
9922     \write\glswrite{(define-location-class
9923         \string"roman-page-numbers\string"^^J\space\space\space
9924         (\string"roman-numbers-lowercase\string")
9925         :min-range-length \@glsminrange)}}%
9926 \else
9927     \write\glswrite{(define-location-class
9928         \string"roman-page-numbers\string"^^J\space\space\space
9929         (:sep "\@gls@roman")
9930         :min-range-length \@glsminrange)}}%
9931 \fi
9932 \write\glswrite{(define-location-class
9933     \string"Roman-page-numbers\string"^^J\space\space\space
9934     (\string"roman-numbers-uppercase\string")
9935     :min-range-length \@glsminrange)}}%
9936 \write\glswrite{(define-location-class
9937     \string"arabic-page-numbers\string"^^J\space\space\space
9938     (\string"arabic-numbers\string")
9939     :min-range-length \@glsminrange)}}%
9940 \write\glswrite{(define-location-class
9941     \string"alpha-page-numbers\string"^^J\space\space\space
9942     (\string"alpha\string")
9943     :min-range-length \@glsminrange)}}%
9944 \write\glswrite{(define-location-class
9945     \string"Alpha-page-numbers\string"^^J\space\space\space
9946     (\string"ALPHA\string")
9947     :min-range-length \@glsminrange)}}%
9948 \write\glswrite{(define-location-class
9949     \string"Appendix-page-numbers\string"^^J\space\space\space
9950     (\string"ALPHA\string"
9951     :sep \string"\@glsAlphacompositor\string"
9952     \string"arabic-numbers\string")
9953     :min-range-length \@glsminrange)}}%
9954 \write\glswrite{(define-location-class
9955     \string"arabic-section-numbers\string"^^J\space\space\space
9956     (\string"arabic-numbers\string"
9957     :sep \string"\glscompositor\string"
9958     \string"arabic-numbers\string")
9959     :min-range-length \@glsminrange)}}%
9960 \write\glswrite{^^J; user defined location classes}%
9961 \write\glswrite{\@xdyuserlocationdefs}%
9962 \write\glswrite{^^J; define cross-reference class^^J}%
9963 \write\glswrite{(define-crossref-class \string"see\string"
9964     :unverified )}%

```

```

9965 \write\glswrite{(markup-crossref-list
9966 :class \string"see\string"^^J\space\space\space
9967 :open \string"\string\glseeformat\string"
9968 :close \string"{}\string")}%
9969 \write\glswrite{^^J; define the order of the location classes}%
9970 \write\glswrite{(define-location-class-order
9971 (\@xdylocationclassorder))}%
9972 \write\glswrite{^^J; define the glossary markup^^J}%
9973 \write\glswrite{(markup-index^^J\space\space\space
9974 :open \string"\string
9975 \glossarysection[\string\glossarytoctitle]{\string
9976 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9977 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9978 \space\space:close \string"\expandafter\@gobble
9979 \string%\string~n\string
9980 \end{theglossary}\string\glossarypostamble
9981 \string~n\string" ^^J\space\space\space
9982 :tree)}}%
9983 \write\glswrite{(markup-letter-group-list
9984 :sep \string"\string\glsgroupskip\string~n\string")}%
9985 \write\glswrite{(markup-indexentry
9986 :open \string"\string\relax \string\glresetentrylist
9987 \string~n\string")}%
9988 \write\glswrite{(markup-locclass-list :open
9989 \string"\glsoopenbrace\string\glossaryentrynumbers
9990 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9991 :sep \string", \string"
9992 :close \string"\glsclosebrace\glsclosebrace\string")}%
9993 \write\glswrite{(markup-locref-list
9994 :sep \string"\string\delimN\space\string")}%
9995 \write\glswrite{(markup-range
9996 :sep \string"\string\delimR\space\string")}%
9997 \@onelevel@sanitize\gls@suffixF
9998 \@onelevel@sanitize\gls@suffixFF
9999 \ifx\gls@suffixF\@empty
10000 \else
10001 \write\glswrite{(markup-range
10002 :close "\gls@suffixF" :length 1 :ignore-end)}}%
10003 \fi
10004 \ifx\gls@suffixFF\@empty
10005 \else
10006 \write\glswrite{(markup-range
10007 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
10008 \fi
10009 \write\glswrite{^^J; define format to use for locations^^J}%
10010 \write\glswrite{\@xdylocref}%
10011 \write\glswrite{^^J; define letter group list format^^J}%
10012 \write\glswrite{(markup-letter-group-list
10013 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

10014 \write\glswrite{^^J; letter group headings^^J}%
10015 \write\glswrite{(markup-letter-group
10016 :open-head \string"\string\glsgroupheading
10017 \glsoopenbrace\string"^^J\space\space\space
10018 :close-head \string"\glsclosebrace\string")}%
10019 \write\glswrite{^^J; additional letter groups^^J}%
10020 \write\glswrite{\@xdylettergroups}%
10021 \write\glswrite{^^J; additional sort rules^^J}
10022 \write\glswrite{\@xdysortrules}%
10023 \noist}
10024 \else
10025 \edef\@gls@actualchar{\string?}
10026 \edef\@gls@encapchar{\string|}
10027 \edef\@gls@levelchar{\string!}
10028 \edef\@gls@quotechar{\string"}
10029 \def\writeist{\relax
10030 \openout\glswrite=\istfilename
10031 \write\glswrite{\expandafter\@gobble\string\% makeindex style file
10032 created by the glossaries package}
10033 \write\glswrite{\expandafter\@gobble\string\% for document
10034 '\jobname' on \the\year-\the\month-\the\day}
10035 \write\glswrite{actual '\@gls@actualchar'}
10036 \write\glswrite{encap '\@gls@encapchar'}
10037 \write\glswrite{level '\@gls@levelchar'}
10038 \write\glswrite{quote '\@gls@quotechar'}
10039 \write\glswrite{keyword \string"\string\glossaryentry\string"}
10040 \write\glswrite{preamble \string"\string\glossarysection[\string
10041 \glossarytoctitle]{\string\glossarytitle}\string
10042 \glossarypreamble\string\n\string\begin{theglossary}\string
10043 \glossaryheader\string\n\string"}
10044 \write\glswrite{postamble \string"\string%\string\n\string
10045 \end{theglossary}\string\glossarypostamble\string\n
10046 \string"}
10047 \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
10048 \string"}
10049 \write\glswrite{item_0 \string"\string%\string\n\string"}
10050 \write\glswrite{item_1 \string"\string%\string\n\string"}
10051 \write\glswrite{item_2 \string"\string%\string\n\string"}
10052 \write\glswrite{item_01 \string"\string%\string\n\string"}
10053 \write\glswrite{item_x1
10054 \string"\string\relax \string\glsresetentrylist\string\n
10055 \string"}
10056 \write\glswrite{item_12 \string"\string%\string\n\string"}
10057 \write\glswrite{item_x2
10058 \string"\string\relax \string\glsresetentrylist\string\n
10059 \string"}
10060 \write\glswrite{delim_0 \string"\string\{\string
10061 \glossaryentrynumbers\string\{\string\relax \string"}
10062 \write\glswrite{delim_1 \string"\string\{\string

```

```

10063      \glossaryentrynumbers\string\{\string\relax \string}
10064      \write\glswrite{delim_2 \string"\string\{\string
10065      \glossaryentrynumbers\string\{\string\relax \string}
10066      \write\glswrite{delim_t \string"\string\}\string\}\string}
10067      \write\glswrite{delim_n \string"\string\delimN \string}
10068      \write\glswrite{delim_r \string"\string\delimR \string}
10069      \write\glswrite{headings_flag 1}
10070      \write\glswrite{heading_prefix
10071      \string"\string\glsgroupheading\string\{\string}
10072      \write\glswrite{heading_suffix
10073      \string"\string\}\string\relax
10074      \string\glresetentrylist \string}
10075      \write\glswrite{symhead_positive \string"glssymbols\string}
10076      \write\glswrite{numhead_positive \string"glslnumbers\string}
10077      \write\glswrite{page_compositor \string"glscpositor\string}
10078      \@gls@escbsdq\gls@suffixF
10079      \@gls@escbsdq\gls@suffixFF
10080      \ifx\gls@suffixF\@empty
10081      \else
10082      \write\glswrite{suffix_2p \string"\gls@suffixF\string}
10083      \fi
10084      \ifx\gls@suffixFF\@empty
10085      \else
10086      \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
10087      \fi
10088      \noist
10089  }
10090 \fi

```

\noist

```
10091 \renewcommand*{\noist}{\let\writeist\relax}
```

4.2 glossaries-compatible-307

```

10092 \NeedsTeXFormat{LaTeX2e}
10093 \ProvidesPackage{glossaries-compatible-307}[2020/02/13 v4.45 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

10094 \newcommand{\compatglossarystyle}[2]{%
10095   \ifcsundef{@glscompstyle@#1}%
10096   {%
10097     \csdef{@glscompstyle@#1}{#2}%
10098   }%
10099   {%
10100     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
10101   }%
10102 }

```

Backward compatible inline style.

```
10103 \compatglossarystyle{inline}{%
10104   \renewcommand{\glossaryentryfield}[5]{%
10105     \glsinlinedopostchild
10106     \gls@inlinesep
10107     \def\glo@desc{##3}%
10108     \def\@no@post@desc{\nopostdesc}%
10109     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
10110     \ifx\glo@desc\@no@post@desc
10111       \glsinlineemptydescformat{##4}{##5}%
10112     \else
10113       \ifstrepty{##3}%
10114         {\glsinlineemptydescformat{##4}{##5}}%
10115         {\glsinlinedescformat{##3}{##4}{##5}}%
10116     \fi
10117     \ifglshaschildren{##1}%
10118     {%
10119       \glsresetsubentrycounter
10120       \glsinlineparentchildseparator
10121       \def\gls@inlinesubsep{}%
10122       \def\gls@inlinepostchild{\glsinlinepostchild}%
10123     }%
10124   }%
10125   \def\gls@inlinesep{\glsinlineseparator}%
10126 }%
```

Sub-entries display description:

```
10127 \renewcommand{\glossarysubentryfield}[6]{%
10128   \gls@inlinesubsep%
10129   \glsinlinesubnameformat{##2}{##3}%
10130   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
10131   \def\gls@inlinesubsep{\glsinlinesubseparator}%
10132 }%
10133 }
```

Backward compatible list style.

```
10134 \compatglossarystyle{list}{%
10135   \renewcommand*\{\glossaryentryfield}[5]{%
10136     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
10137       ##3\glspostdescription\space ##5}%
10138 }
```

Sub-entries continue on the same line:

```
10138 \renewcommand*\{\glossarysubentryfield}[6]{%
10139   \glssubentryitem{##2}%
10140   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
10141 }
```

Backward compatible listgroup style.

```
10142 \compatglossarystyle{listgroup}{%
10143   \csuse{@glscompstyle@list}%
10144 }%
```

Backward compatible listhypergroup style.

```
10145 \compatglossarystyle{listhypergroup}{%
10146   \csuse{@glscompstyle@list}%
10147 }%
```

Backward compatible altlist style.

```
10148 \compatglossarystyle{altlist}{%
10149   \renewcommand*{\glossaryentryfield}[5]{%
10150     \item[\glentryitem{##1}\glstarget{##1}{##2}]%
10151       \mbox{}\par\nobreak\@afterheading
10152       ##3\glspostdescription\space ##5}%
10153   \renewcommand{\glossarysubentryfield}[6]{%
10154     \par
10155     \glssubentryitem{##2}%
10156     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
10157 }%
```

Backward compatible altlistgroup style.

```
10158 \compatglossarystyle{altlistgroup}{%
10159   \csuse{@glscompstyle@altlist}%
10160 }%
```

Backward compatible altlisthypergroup style.

```
10161 \compatglossarystyle{altlisthypergroup}{%
10162   \csuse{@glscompstyle@altlist}%
10163 }%
```

Backward compatible listdotted style.

```
10164 \compatglossarystyle{listdotted}{%
10165   \renewcommand*{\glossaryentryfield}[5]{%
10166     \item[\makebox[\glslistdottedwidth][l]{%
10167       \glentryitem{##1}\glstarget{##1}{##2}%
10168       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
10169   \renewcommand*{\glossarysubentryfield}[6]{%
10170     \item[\makebox[\glslistdottedwidth][l]{%
10171       \glssubentryitem{##2}%
10172       \glstarget{##2}{##3}%
10173       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
10174 }%
```

Backward compatible sublistdotted style.

```
10175 \compatglossarystyle{sublistdotted}{%
10176   \csuse{@glscompstyle@listdotted}%
10177   \renewcommand*{\glossaryentryfield}[5]{%
10178     \item[\glentryitem{##1}\glstarget{##1}{##2}]}%
10179 }%
```

Backward compatible long style.

```
10180 \compatglossarystyle{long}{%
10181   \renewcommand*{\glossaryentryfield}[5]{%
10182     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10183   \renewcommand*{\glossarysubentryfield}[6]{%
10184     \glssubentryitem{##2}\glstarget{##2}{##3} & ##4\glspostdescription\space ##5\\}%
10185 }%
```

```

10184      &
10185      \glssubentryitem{##2}%
10186      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10187 }%

```

Backward compatible longborder style.

```

10188 \compatglossarystyle{longborder}{%
10189   \csuse{@glscmpstyle@long}%
10190 }%

```

Backward compatible longheader style.

```

10191 \compatglossarystyle{longheader}{%
10192   \csuse{@glscmpstyle@long}%
10193 }%

```

Backward compatible longheaderborder style.

```

10194 \compatglossarystyle{longheaderborder}{%
10195   \csuse{@glscmpstyle@long}%
10196 }%

```

Backward compatible long3col style.

```

10197 \compatglossarystyle{long3col}{%
10198   \renewcommand*{\glossaryentryfield}[5]{%
10199     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10200   \renewcommand*{\glossarysubentryfield}[6]{%
10201     &
10202     \glssubentryitem{##2}%
10203     \glstarget{##2}{\strut}##4 & ##6\\}%
10204 }%

```

Backward compatible long3colborder style.

```

10205 \compatglossarystyle{long3colborder}{%
10206   \csuse{@glscmpstyle@long3col}%
10207 }%

```

Backward compatible long3colheader style.

```

10208 \compatglossarystyle{long3colheader}{%
10209   \csuse{@glscmpstyle@long3col}%
10210 }%

```

Backward compatible long3colheaderborder style.

```

10211 \compatglossarystyle{long3colheaderborder}{%
10212   \csuse{@glscmpstyle@long3col}%
10213 }%

```

Backward compatible long4col style.

```

10214 \compatglossarystyle{long4col}{%
10215   \renewcommand*{\glossaryentryfield}[5]{%
10216     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10217   \renewcommand*{\glossarysubentryfield}[6]{%
10218     &
10219     \glssubentryitem{##2}%

```


10220 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
 10221 }%

Backward compatible long4colheader style.

10222 \compatglossarystyle{long4colheader}{%
 10223 \csuse{@glscompstyle@long4col}%
 10224 }%

Backward compatible long4colborder style.

10225 \compatglossarystyle{long4colborder}{%
 10226 \csuse{@glscompstyle@long4col}%
 10227 }%

Backward compatible long4colheaderborder style.

10228 \compatglossarystyle{long4colheaderborder}{%
 10229 \csuse{@glscompstyle@long4col}%
 10230 }%

Backward compatible altlong4col style.

10231 \compatglossarystyle{altlong4col}{%
 10232 \csuse{@glscompstyle@long4col}%
 10233 }%

Backward compatible altlong4colheader style.

10234 \compatglossarystyle{altlong4colheader}{%
 10235 \csuse{@glscompstyle@long4col}%
 10236 }%

Backward compatible altlong4colborder style.

10237 \compatglossarystyle{altlong4colborder}{%
 10238 \csuse{@glscompstyle@long4col}%
 10239 }%

Backward compatible altlong4colheaderborder style.

10240 \compatglossarystyle{altlong4colheaderborder}{%
 10241 \csuse{@glscompstyle@long4col}%
 10242 }%

Backward compatible long style.

10243 \compatglossarystyle{longragged}{%
 10244 \renewcommand*{\glossaryentryfield}[5]{%
 10245 \glstarget{##1}{\strut}##2 & ##3\glspostdescription\space ##5%
 10246 \tabularnewline}%
 10247 \renewcommand*{\glossarysubentryfield}[6]{%
 10248 &
 10249 \glssubentryitem{##2}%
 10250 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 10251 \tabularnewline}%
 10252 }%

Backward compatible longraggedborder style.

10253 \compatglossarystyle{longraggedborder}{%
 10254 \csuse{@glscompstyle@longragged}%
 10255 }%

Backward compatible longraggedheader style.

```
10256 \compatglossarystyle{longraggedheader}{%
10257   \csuse{@glscompstyle@longragged}%
10258 }%
```

Backward compatible longraggedheaderborder style.

```
10259 \compatglossarystyle{longraggedheaderborder}{%
10260   \csuse{@glscompstyle@longragged}%
10261 }%
```

Backward compatible longragged3col style.

```
10262 \compatglossarystyle{longragged3col}{%
10263   \renewcommand*{\glossaryentryfield}[5]{%
10264     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10265   \renewcommand*{\glossarysubentryfield}[6]{%
10266     &
10267     \glssubentryitem{##2}%
10268     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10269 }%
```

Backward compatible longragged3colborder style.

```
10270 \compatglossarystyle{longragged3colborder}{%
10271   \csuse{@glscompstyle@longragged3col}%
10272 }%
```

Backward compatible longragged3colheader style.

```
10273 \compatglossarystyle{longragged3colheader}{%
10274   \csuse{@glscompstyle@longragged3col}%
10275 }%
```

Backward compatible longragged3colheaderborder style.

```
10276 \compatglossarystyle{longragged3colheaderborder}{%
10277   \csuse{@glscompstyle@longragged3col}%
10278 }%
```

Backward compatible altlongragged4col style.

```
10279 \compatglossarystyle{altlongragged4col}{%
10280   \renewcommand*{\glossaryentryfield}[5]{%
10281     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10282   \renewcommand*{\glossarysubentryfield}[6]{%
10283     &
10284     \glssubentryitem{##2}%
10285     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10286 }%
```

Backward compatible altlongragged4colheader style.

```
10287 \compatglossarystyle{altlongragged4colheader}{%
10288   \csuse{@glscompstyle@altlong4col}%
10289 }%
```

Backward compatible altlongragged4colborder style.

```
10290 \compatglossarystyle{altlongragged4colborder}{%
```

```
10291 \csuse{@glscompstyle@altlong4col}%
10292 }%
```

Backward compatible altlongragged4colheaderborder style.

```
10293 \compatglossarystyle{altlongragged4colheaderborder}{%
10294 \csuse{@glscompstyle@altlong4col}%
10295 }%
```

Backward compatible index style.

```
10296 \compatglossarystyle{index}{%
10297 \renewcommand*{\glossaryentryfield}[5]{%
10298 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10299 \ifx\relax##4\relax
10300 \else
10301 \space{##4}%
10302 \fi
10303 \space ##3\glspostdescription \space ##5}%
10304 \renewcommand*{\glossarysubentryfield}[6]{%
10305 \ifcase##1\relax
10306 % level 0
10307 \item
10308 \or
10309 % level 1
10310 \subitem
10311 \glssubentryitem{##2}%
10312 \else
10313 % all other levels
10314 \subsubitem
10315 \fi
10316 \textbf{\glstarget{##2}{##3}}%
10317 \ifx\relax##5\relax
10318 \else
10319 \space{##5}%
10320 \fi
10321 \space##4\glspostdescription\space ##6}%
10322 }%
```

Backward compatible indexgroup style.

```
10323 \compatglossarystyle{indexgroup}{%
10324 \csuse{@glscompstyle@index}%
10325 }%
```

Backward compatible indexhypergroup style.

```
10326 \compatglossarystyle{indexhypergroup}{%
10327 \csuse{@glscompstyle@index}%
10328 }%
```

Backward compatible tree style.

```
10329 \compatglossarystyle{tree}{%
10330 \renewcommand{\glossaryentryfield}[5]{%
10331 \hangindent0pt\relax
```

```

10332 \parindent0pt\relax
10333 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10334 \ifx\relax##4\relax
10335 \else
10336 \space(##4)%
10337 \fi
10338 \space ##3\glspostdescription \space ##5\par}%
10339 \renewcommand{\glossarysubentryfield}[6]{%
10340 \hangindent##1\glstreeindent\relax
10341 \parindent##1\glstreeindent\relax
10342 \ifnum##1=1\relax
10343 \glssubentryitem{##2}%
10344 \fi
10345 \textbf{\glstarget{##2}{##3}}%
10346 \ifx\relax##5\relax
10347 \else
10348 \space(##5)%
10349 \fi
10350 \space##4\glspostdescription\space ##6\par}%
10351 }%

```

Backward compatible treegroup style.

```

10352 \compatglossarystyle{treegroup}{%
10353 \csuse{@glscmpstyle@tree}%
10354 }%

```

Backward compatible treehypergroup style.

```

10355 \compatglossarystyle{treehypergroup}{%
10356 \csuse{@glscmpstyle@tree}%
10357 }%

```

Backward compatible treenoname style.

```

10358 \compatglossarystyle{treenoname}{%
10359 \renewcommand{\glossaryentryfield}[5]{%
10360 \hangindent0pt\relax
10361 \parindent0pt\relax
10362 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10363 \ifx\relax##4\relax
10364 \else
10365 \space(##4)%
10366 \fi
10367 \space ##3\glspostdescription \space ##5\par}%
10368 \renewcommand{\glossarysubentryfield}[6]{%
10369 \hangindent##1\glstreeindent\relax
10370 \parindent##1\glstreeindent\relax
10371 \ifnum##1=1\relax
10372 \glssubentryitem{##2}%
10373 \fi
10374 \glstarget{##2}{\strut}%
10375 ##4\glspostdescription\space ##6\par}%
10376 }%

```

Backward compatible treenonamegroup style.

```
10377 \compatglossarystyle{treenonamegroup}{%
10378   \csuse{@glscompstyle@treenoname}%
10379 }%
```

Backward compatible treenonamehypergroup style.

```
10380 \compatglossarystyle{treenonamehypergroup}{%
10381   \csuse{@glscompstyle@treenoname}%
10382 }%
```

Backward compatible alttree style.

```
10383 \compatglossarystyle{almtree}{%
10384   \renewcommand{\glossaryentryfield}[5]{%
10385     \ifnum\@gls@prevlevel=0\relax
10386     \else
10387       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10388       \hangindent\glstreeindent
10389       \parindent\glstreeindent
10390     \fi
10391     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10392       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10393     \ifx\relax##4\relax
10394     \else
10395       (##4)\space
10396     \fi
10397     ##3\glspostdescription \space ##5\par
10398     \def\@gls@prevlevel{0}%
10399   }%
10400   \renewcommand{\glossarysubentryfield}[6]{%
10401     \ifnum##1=1\relax
10402       \glssubentryitem{##2}%
10403     \fi
10404     \ifnum\@gls@prevlevel=##1\relax
10405     \else
10406       \@ifundefined{@glswidestname\romannumeral##1}{%
10407         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10408         \settowidth{\gls@tmplen}{\textbf{%
10409           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10410       \ifnum\@gls@prevlevel<##1\relax
10411         \setlength\glstreeindent\gls@tmplen
10412         \addtolength\glstreeindent\parindent
10413         \parindent\glstreeindent
10414       \else
10415         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10416           \settowidth{\glstreeindent}{\textbf{%
10417             \@glswidestname\space}}{%
10418           \settowidth{\glstreeindent}{\textbf{%
10419             \csname @glswidestname\romannumeral\@gls@prevlevel
10420             \endcsname\space}}}%
10421           \addtolength\parindent{-\glstreeindent}%

```

```

10422         \setlength\glstreeindent\parindent
10423     \fi
10424 \fi
10425 \hangindent\glstreeindent
10426 \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
10427     \textbf{\glstarget{##2}{##3}}}%
10428 \ifx##5\relax\relax
10429 \else
10430     (##5)\space
10431 \fi
10432 ##4\glspostdescription\space ##6\par
10433 \def\@gls@prevlevel{##1}%
10434 }%
10435 }%

```

Backward compatible alttreegroup style.

```

10436 \compatglossarystyle{alttreegroup}{%
10437 \csuse{@glscompstyle@alttree}%
10438 }%

```

Backward compatible alttreehypergroup style.

```

10439 \compatglossarystyle{alttreehypergroup}{%
10440 \csuse{@glscompstyle@alttree}%
10441 }%

```

Backward compatible mcolindex style.

```

10442 \compatglossarystyle{mcolindex}{%
10443 \csuse{@glscompstyle@index}%
10444 }%

```

Backward compatible mcolindexgroup style.

```

10445 \compatglossarystyle{mcolindexgroup}{%
10446 \csuse{@glscompstyle@index}%
10447 }%

```

Backward compatible mcolindexhypergroup style.

```

10448 \compatglossarystyle{mcolindexhypergroup}{%
10449 \csuse{@glscompstyle@index}%
10450 }%

```

Backward compatible mcoltree style.

```

10451 \compatglossarystyle{mcoltree}{%
10452 \csuse{@glscompstyle@tree}%
10453 }%

```

Backward compatible mcoltreegroup style.

```

10454 \compatglossarystyle{mcolindextreegroup}{%
10455 \csuse{@glscompstyle@tree}%
10456 }%

```

Backward compatible mcoltreehypergroup style.

```

10457 \compatglossarystyle{mcolindextreehypergroup}{%

```

```

10458 \csuse{@glscompstyle@tree}%
10459 }%

    Backward compatible mcoltreenoname style.
10460 \compatglossarystyle{mcoltreenoname}{%
10461 \csuse{@glscompstyle@tree}%
10462 }%

    Backward compatible mcoltreenonamegroup style.
10463 \compatglossarystyle{mcoltreenonamegroup}{%
10464 \csuse{@glscompstyle@tree}%
10465 }%

    Backward compatible mcoltreenonamehypergroup style.
10466 \compatglossarystyle{mcoltreenonamehypergroup}{%
10467 \csuse{@glscompstyle@tree}%
10468 }%

    Backward compatible mcolalmtree style.
10469 \compatglossarystyle{mcolalmtree}{%
10470 \csuse{@glscompstyle@almtree}%
10471 }%

    Backward compatible mcolalmtreegroup style.
10472 \compatglossarystyle{mcolalmtreegroup}{%
10473 \csuse{@glscompstyle@almtree}%
10474 }%

    Backward compatible mcolalmtreehypergroup style.
10475 \compatglossarystyle{mcolalmtreehypergroup}{%
10476 \csuse{@glscompstyle@almtree}%
10477 }%

    Backward compatible superragged style.
10478 \compatglossarystyle{superragged}{%
10479 \renewcommand*{\glossaryentryfield}[5]{%
10480 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10481 \tabularnewline}%
10482 \renewcommand*{\glossarysubentryfield}[6]{%
10483 &
10484 \glssubentryitem{##2}%
10485 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10486 \tabularnewline}%
10487 }%

    Backward compatible superraggedborder style.
10488 \compatglossarystyle{superraggedborder}{%
10489 \csuse{@glscompstyle@superragged}%
10490 }%

    Backward compatible superraggedheader style.
10491 \compatglossarystyle{superraggedheader}{%
10492 \csuse{@glscompstyle@superragged}%
10493 }%

```

Backward compatible superraggedheaderborder style.

```
10494 \compatglossarystyle{superraggedheaderborder}{%
10495   \csuse{@glscompstyle@superragged}%
10496 }%
```

Backward compatible superragged3col style.

```
10497 \compatglossarystyle{superragged3col}{%
10498   \renewcommand*{\glossaryentryfield}[5]{%
10499     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10500   \renewcommand*{\glossarysubentryfield}[6]{%
10501     &
10502     \glssubentryitem{##2}%
10503     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10504 }%
```

Backward compatible superragged3colborder style.

```
10505 \compatglossarystyle{superragged3colborder}{%
10506   \csuse{@glscompstyle@superragged3col}%
10507 }%
```

Backward compatible superragged3colheader style.

```
10508 \compatglossarystyle{superragged3colheader}{%
10509   \csuse{@glscompstyle@superragged3col}%
10510 }%
```

Backward compatible superragged3colheaderborder style.

```
10511 \compatglossarystyle{superragged3colheaderborder}{%
10512   \csuse{@glscompstyle@superragged3col}%
10513 }%
```

Backward compatible altsuperragged4col style.

```
10514 \compatglossarystyle{altsuperragged4col}{%
10515   \renewcommand*{\glossaryentryfield}[5]{%
10516     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10517   \renewcommand*{\glossarysubentryfield}[6]{%
10518     &
10519     \glssubentryitem{##2}%
10520     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10521 }%
```

Backward compatible altsuperragged4colheader style.

```
10522 \compatglossarystyle{altsuperragged4colheader}{%
10523   \csuse{@glscompstyle@altsuperragged4col}%
10524 }%
```

Backward compatible altsuperragged4colborder style.

```
10525 \compatglossarystyle{altsuperragged4colborder}{%
10526   \csuse{@glscompstyle@altsuperragged4col}%
10527 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10528 \compatglossarystyle{altsuperragged4colheaderborder}{%
```



```

10529 \csuse{@glscompstyle@altsuperragged4col}%
10530 }%

```

Backward compatible super style.

```

10531 \compatglossarystyle{super}{%
10532   \renewcommand*{\glossaryentryfield}[5]{%
10533     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10534   \renewcommand*{\glossarysubentryfield}[6]{%
10535     &
10536     \glssubentryitem{##2}%
10537     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10538 }%

```

Backward compatible superborder style.

```

10539 \compatglossarystyle{superborder}{%
10540   \csuse{@glscompstyle@super}%
10541 }%

```

Backward compatible superheader style.

```

10542 \compatglossarystyle{superheader}{%
10543   \csuse{@glscompstyle@super}%
10544 }%

```

Backward compatible superheaderborder style.

```

10545 \compatglossarystyle{superheaderborder}{%
10546   \csuse{@glscompstyle@super}%
10547 }%

```

Backward compatible super3col style.

```

10548 \compatglossarystyle{super3col}{%
10549   \renewcommand*{\glossaryentryfield}[5]{%
10550     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10551   \renewcommand*{\glossarysubentryfield}[6]{%
10552     &
10553     \glssubentryitem{##2}%
10554     \glstarget{##2}{\strut}##4 & ##6\\}%
10555 }%

```

Backward compatible super3colborder style.

```

10556 \compatglossarystyle{super3colborder}{%
10557   \csuse{@glscompstyle@super3col}%
10558 }%

```

Backward compatible super3colheader style.

```

10559 \compatglossarystyle{super3colheader}{%
10560   \csuse{@glscompstyle@super3col}%
10561 }%

```

Backward compatible super3colheaderborder style.

```

10562 \compatglossarystyle{super3colheaderborder}{%
10563   \csuse{@glscompstyle@super3col}%
10564 }%

```

Backward compatible super4col style.

```
10565 \compatglossarystyle{super4col}{%
10566   \renewcommand*{\glossaryentryfield}[5]{%
10567     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10568   \renewcommand*{\glossarysubentryfield}[6]{%
10569     &
10570     \glssubentryitem{##2}%
10571     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10572 }%
```

Backward compatible super4colheader style.

```
10573 \compatglossarystyle{super4colheader}{%
10574   \csuse{@glscompstyle@super4col}%
10575 }%
```

Backward compatible super4colborder style.

```
10576 \compatglossarystyle{super4colborder}{%
10577   \csuse{@glscompstyle@super4col}%
10578 }%
```

Backward compatible super4colheaderborder style.

```
10579 \compatglossarystyle{super4colheaderborder}{%
10580   \csuse{@glscompstyle@super4col}%
10581 }%
```

Backward compatible altsuper4col style.

```
10582 \compatglossarystyle{altsuper4col}{%
10583   \csuse{@glscompstyle@super4col}%
10584 }%
```

Backward compatible altsuper4colheader style.

```
10585 \compatglossarystyle{altsuper4colheader}{%
10586   \csuse{@glscompstyle@super4col}%
10587 }%
```

Backward compatible altsuper4colborder style.

```
10588 \compatglossarystyle{altsuper4colborder}{%
10589   \csuse{@glscompstyle@super4col}%
10590 }%
```

Backward compatible altsuper4colheaderborder style.

```
10591 \compatglossarystyle{altsuper4colheaderborder}{%
10592   \csuse{@glscompstyle@super4col}%
10593 }%
```

5 Accessibility Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibility support in glossary entries. See the documentation for further details about accessibility support.

```
10594 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10595 \ProvidesPackage{glossaries-accsupp}[2020/02/13 v4.45 (NLCT)]
```

```
10596 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10597 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10598 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10599 \@ifpackageloaded{glossaries-extra}
```

```
10600 {%
```

If the accsupp option was used, \glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10601 \ifx\glsxtr@doaccsupp\empty
```

```
10602 \GlossariesWarning{The 'glossaries-accsupp'
```

```
10603 package has been loaded\MessageBreak
```

```
10604 after the 'glossaries-extra' package. This\MessageBreak
```

```
10605 can cause a failure to integrate both packages. \MessageBreak
```

```
10606 Either use the 'accsupp' option when you load\MessageBreak
```

```
10607 'glossaries-extra' or load 'glossaries-accsupp'\MessageBreak
```

```
10608 before loading 'glossaries-extra'}%
```

```
10609 \fi
```

```
10610 }
```

```
10611 {}
```

tibleglossentry Override style compatibility macros:

```
10612 \def\compatibleglossentry#1#2{%
```

```
10613 \toks@{#2}%
```

```
10614 \protected@edef\do@glossentry{%
```

```
10615 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10616 {\noexpand\glsnamefont
```

```
10617 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```

10618     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@desc\endcsname}%
10619     {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@symbol\endcsname}%
10620     {\the\toks@}%
10621   }%
10622   \@do@glossentry
10623 }

```

lesubglossentry

```

10624 \def\compatiblesubglossentry#1#2#3{%
10625   \toks@{#3}%
10626   \protected@edef\@do@subglossentry{%
10627     \noexpand\accsuppglossarysubentryfield{\number#1}%
10628     {#2}%
10629     {\noexpand\glsnamefont
10630      {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@name\endcsname}}%
10631     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@desc\endcsname}%
10632     {\expandafter\expandonce\csname glo@\glsdetoklabel{#2}@symbol\endcsname}%
10633     {\the\toks@}%
10634   }%
10635   \@do@subglossentry
10636 }

```

Required packages:

```

10637 \RequirePackage{glossaries}

```

@accsupp@engine There's currently only support for accsupp, but if you define \gls@accsupp@engine before loading glossaries-accsupp, you can prevent accsupp from being loaded. Redefining this command after glossaries-accsupp has loaded obviously won't do anything (and so is an internal command to deter casual use). If it is defined to something other than accsupp then \gls@accessibility will need to be defined to something appropriate.

```

10638 \providecommand{\gls@accsupp@engine}{accsupp}

```

s@accessibility `\gls@accessibility[<options>]{<PDF element>}{<value>}{<content>}`

```

10639 \providecommand{\gls@accessibility}[4]{#4}
10640 \ifdefstring\gls@accsupp@engine{accsupp}
10641 {
10642   \RequirePackage{accsupp}
10643   \renewcommand{\gls@accessibility}[4]{%
10644     \BeginAccSupp{#1,#2={#3}}#4\EndAccSupp}%
10645   }
10646 }
10647 {}

```

lsaccessibility `\glsaccessibility[<options>]{<PDF element>}{<value>}{<content>}`

User-level command that includes debug info if required.

```

10648 \newcommand{\glsaccessibility}[4] [] {%
10649   \@glsshowaccsupp{#1}{#2}{#3}%
10650   \gls@accessibility{#1}{#2}{#3}{#4}%
10651 }

```

5.1 Defining Replacement Text

The version 0.1 stored the replacement text in the symbol key. This has been changed to use the new keys defined here. Example of use:

```
\newglossaryentry{dr}{name=Dr,description={},access={Doctor}}
```

access The replacement text corresponding to the name key:

```

10652 \define@key{glossentry}{access}{%
10653   \def\@glo@access{#1}%
10654 }

```

textaccess The replacement text corresponding to the text key:

```

10655 \define@key{glossentry}{textaccess}{%
10656   \def\@glo@textaccess{#1}%
10657 }

```

firstaccess The replacement text corresponding to the first key:

```

10658 \define@key{glossentry}{firstaccess}{%
10659   \def\@glo@firstaccess{#1}%
10660 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10661 \define@key{glossentry}{pluralaccess}{%
10662   \def\@glo@pluralaccess{#1}%
10663 }

```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

10664 \define@key{glossentry}{firstpluralaccess}{%
10665   \def\@glo@firstpluralaccess{#1}%
10666 }

```

symbolaccess The replacement text corresponding to the symbol key:

```

10667 \define@key{glossentry}{symbolaccess}{%
10668   \def\@glo@symbolaccess{#1}%
10669 }

```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```

10670 \define@key{glossentry}{symbolpluralaccess}{%
10671   \def\@glo@symbolpluralaccess{#1}%
10672 }

```

descriptionaccess The replacement text corresponding to the description key:

```
10673 \define@key{glossentry}{descriptionaccess}{%
10674   \def\@glo@descaccess{#1}%
10675 }
```

ionpluralaccess The replacement text corresponding to the descriptionplural key:

```
10676 \define@key{glossentry}{descriptionpluralaccess}{%
10677   \def\@glo@descpluralaccess{#1}%
10678 }
```

shortaccess The replacement text corresponding to the short key:

```
10679 \define@key{glossentry}{shortaccess}{%
10680   \def\@glo@shortaccess{#1}%
10681 }
```

ortpluralaccess The replacement text corresponding to the shortplural key:

```
10682 \define@key{glossentry}{shortpluralaccess}{%
10683   \def\@glo@shortpluralaccess{#1}%
10684 }
```

longaccess The replacement text corresponding to the long key:

```
10685 \define@key{glossentry}{longaccess}{%
10686   \def\@glo@longaccess{#1}%
10687 }
```

ongpluralaccess The replacement text corresponding to the longplural key:

```
10688 \define@key{glossentry}{longpluralaccess}{%
10689   \def\@glo@longpluralaccess{#1}%
10690 }
```

There are now also keys that correspond to the user keys:

user1access The replacement text corresponding to the user1 key:

```
10691 \define@key{glossentry}{user1access}{%
10692   \def\@glo@useriaccess{#1}%
10693 }
```

user2access The replacement text corresponding to the user2 key:

```
10694 \define@key{glossentry}{user2access}{%
10695   \def\@glo@useriiaccess{#1}%
10696 }
```

user3access The replacement text corresponding to the user3 key:

```
10697 \define@key{glossentry}{user3access}{%
10698   \def\@glo@useriiiaccess{#1}%
10699 }
```

user4access The replacement text corresponding to the user4 key:

```
10700 \define@key{glossentry}{user4access}{%
10701   \def\@glo@userivaccess{#1}%
10702 }
```

user5access The replacement text corresponding to the user5 key:

```
10703 \define@key{glossentry}{user5access}{%
10704   \def\@glo@uservaccess{#1}%
10705 }
```

user6access The replacement text corresponding to the user6 key:

```
10706 \define@key{glossentry}{user6access}{%
10707   \def\@glo@userviaccess{#1}%
10708 }
```

For any custom keys, the replacement text would have to be explicitly put in the value, e.g.,
user1={\glsshortaccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```
10709 \appto\@gls@keymap{,%
10710   {access}{access},%
10711   {textaccess}{textaccess},%
10712   {firstaccess}{firstaccess},%
10713   {pluralaccess}{pluralaccess},%
10714   {firstpluralaccess}{firstpluralaccess},%
10715   {symbolaccess}{symbolaccess},%
10716   {symbolpluralaccess}{symbolpluralaccess},%
10717   {descaccess}{descaccess},%
10718   {descpluralaccess}{descpluralaccess},%
10719   {shortaccess}{shortaccess},%
10720   {shortpluralaccess}{shortpluralaccess},%
10721   {longaccess}{longaccess},%
10722   {longpluralaccess}{longpluralaccess},%
10723   {user1access}{useriaccess},%
10724   {user2access}{useriiaccess},%
10725   {user3access}{useriiiaccess},%
10726   {user4access}{userivaccess},%
10727   {user5access}{uservaccess},%
10728   {user6access}{userviaccess}%
10729 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
10730 \def\@gls@noaccess{\relax}
```

Previously, the access key was initialised to the value of the symbol key at the start for backwards compatibility. This causes a problem for situations where the replacement text is provided for symbol but not for name so this behaviour has been removed.

```
10731 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10732 \renewcommand*{\@newglossaryentryprehook}{%
```

```

10733 \@gls@oldnewglossaryentryprehook
10734 \def\@glo@access{\relax}%

```

Initialise the other keys:

```

10735 \def\@glo@textaccess{\@glo@access}%
10736 \def\@glo@firstaccess{\@glo@access}%
10737 \def\@glo@pluralaccess{\@glo@textaccess}%
10738 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10739 \def\@glo@symbolaccess{\relax}%
10740 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10741 \def\@glo@descaccess{\relax}%
10742 \def\@glo@descpluralaccess{\@glo@descaccess}%
10743 \def\@glo@shortaccess{\relax}%
10744 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10745 \def\@glo@longaccess{\relax}%
10746 \def\@glo@longpluralaccess{\@glo@longaccess}%
10747 \def\@glo@useriaccess{\relax}%
10748 \def\@glo@useriiaaccess{\relax}%
10749 \def\@glo@useriiiaaccess{\relax}%
10750 \def\@glo@userivaccess{\relax}%
10751 \def\@glo@uservaccess{\relax}%
10752 \def\@glo@userviiaaccess{\relax}%
10753 }

```

Add to the end hook:

```

10754 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10755 \renewcommand*{\@newglossaryentryposthook}{%
10756 \@gls@oldnewglossaryentryposthook

```

Store the access information:

```

10757 \expandafter
10758 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10759 \@glo@access}%
10760 \expandafter
10761 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10762 \@glo@textaccess}%
10763 \expandafter
10764 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10765 \@glo@firstaccess}%
10766 \expandafter
10767 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10768 \@glo@pluralaccess}%
10769 \expandafter
10770 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10771 \@glo@firstpluralaccess}%
10772 \expandafter
10773 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10774 \@glo@symbolaccess}%
10775 \expandafter
10776 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10777 \@glo@symbolpluralaccess}%

```



```

10778 \expandafter
10779   \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10780     \@glo@descaccess}%
10781 \expandafter
10782   \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10783     \@glo@descpluralaccess}%
10784 \expandafter
10785   \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10786     \@glo@shortaccess}%
10787 \expandafter
10788   \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10789     \@glo@shortpluralaccess}%
10790 \expandafter
10791   \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10792     \@glo@longaccess}%
10793 \expandafter
10794   \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10795     \@glo@longpluralaccess}%
10796 \expandafter
10797   \protected@xdef\csname glo@\@glo@label @useriaccess\endcsname{%
10798     \@glo@useriaccess}%
10799 \expandafter
10800   \protected@xdef\csname glo@\@glo@label @useriiaccess\endcsname{%
10801     \@glo@useriiaccess}%
10802 \expandafter
10803   \protected@xdef\csname glo@\@glo@label @useriiiaccess\endcsname{%
10804     \@glo@useriiiaccess}%
10805 \expandafter
10806   \protected@xdef\csname glo@\@glo@label @userivaccess\endcsname{%
10807     \@glo@userivaccess}%
10808 \expandafter
10809   \protected@xdef\csname glo@\@glo@label @uservaccess\endcsname{%
10810     \@glo@uservaccess}%
10811 \expandafter
10812   \protected@xdef\csname glo@\@glo@label @userviaccess\endcsname{%
10813     \@glo@userviaccess}%
10814 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10815 \newcommand*{\glsentryaccess}[1]{%
10816   \@gls@entry@field{#1}{access}%
10817 }

```

`entrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

10818 \newcommand*{\glsentrytextaccess}[1]{%
10819   \@gls@entry@field{#1}{textaccess}%

```

```

10820 }

entryfirstaccess  Get the value of the firstaccess key for the entry with the given label:
10821 \newcommand*{\glentryfirstaccess}[1]{%
10822   \@gls@entry@field{#1}{firstaccess}%
10823 }

entrypluralaccess  Get the value of the pluralaccess key for the entry with the given label:
10824 \newcommand*{\glentrypluralaccess}[1]{%
10825   \@gls@entry@field{#1}{pluralaccess}%
10826 }

firstpluralaccess  Get the value of the firstpluralaccess key for the entry with the given label:
10827 \newcommand*{\glentryfirstpluralaccess}[1]{%
10828   \@gls@entry@field{#1}{firstpluralaccess}%
10829 }

entrysymbolaccess  Get the value of the symbolaccess key for the entry with the given label:
10830 \newcommand*{\glentrysymbolaccess}[1]{%
10831   \@gls@entry@field{#1}{symbolaccess}%
10832 }

symbolpluralaccess  Get the value of the symbolpluralaccess key for the entry with the given label:
10833 \newcommand*{\glentrysymbolpluralaccess}[1]{%
10834   \@gls@entry@field{#1}{symbolpluralaccess}%
10835 }

entrydescaccess  Get the value of the descriptionaccess key for the entry with the given label:
10836 \newcommand*{\glentrydescaccess}[1]{%
10837   \@gls@entry@field{#1}{descaccess}%
10838 }

descpluralaccess  Get the value of the descriptionpluralaccess key for the entry with the given label:
10839 \newcommand*{\glentrydescpluralaccess}[1]{%
10840   \@gls@entry@field{#1}{descpluralaccess}%
10841 }

entryshortaccess  Get the value of the shortaccess key for the entry with the given label:
10842 \newcommand*{\glentryshortaccess}[1]{%
10843   \@gls@entry@field{#1}{shortaccess}%
10844 }

shortpluralaccess  Get the value of the shortpluralaccess key for the entry with the given label:
10845 \newcommand*{\glentryshortpluralaccess}[1]{%
10846   \@gls@entry@field{#1}{shortpluralaccess}%
10847 }

```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10848 \newcommand*{\glentrylongaccess}[1]{%
10849   \@gls@entry@field{#1}{longaccess}%
10850 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10851 \newcommand*{\glentrylongpluralaccess}[1]{%
10852   \@gls@entry@field{#1}{longpluralaccess}%
10853 }
```

entryuseriaccess Get the value of the user1access key for the entry with the given label:

```
10854 \newcommand*{\glentryuseriaccess}[1]{%
10855   \@gls@entry@field{#1}{useriaccess}%
10856 }
```

tryuseriiaccess Get the value of the user2access key for the entry with the given label:

```
10857 \newcommand*{\glentryuseriiaccess}[1]{%
10858   \@gls@entry@field{#1}{useriiaccess}%
10859 }
```

ryuseriiiaccess Get the value of the user3access key for the entry with the given label:

```
10860 \newcommand*{\glentryuseriiiaccess}[1]{%
10861   \@gls@entry@field{#1}{useriiiaccess}%
10862 }
```

tryuserivaccess Get the value of the user4access key for the entry with the given label:

```
10863 \newcommand*{\glentryuserivaccess}[1]{%
10864   \@gls@entry@field{#1}{userivaccess}%
10865 }
```

entryuservaccess Get the value of the user5access key for the entry with the given label:

```
10866 \newcommand*{\glentryuservaccess}[1]{%
10867   \@gls@entry@field{#1}{uservaccess}%
10868 }
```

tryuserviaccess Get the value of the user6access key for the entry with the given label:

```
10869 \newcommand*{\glentryuserviaccess}[1]{%
10870   \@gls@entry@field{#1}{userviaccess}%
10871 }
```

There are three types of replacement text:

Alt Description of some content that's non-textual (for example, an image). A word break is assumed after the content.

ActualText A character or sequence of characters that replaces textual content (for example, a dropped capital, a ligature or a symbol). No word break is assumed after the content.

E Expansion of an abbreviation to avoid ambiguity (for example, “St” could be short for “saint” or “street”).

Therefore, rather than having one command for all fields, it’s better to have a command dependent on the field type. For example, the short and shortpl keys would require E, the symbol key would require ActualText, and a field that contains an image would require Alt.

glsfieldaccsupp

`\glsfieldaccsupp{<replacement>}{<content>}{<field>}{<label>}`

Test if there’s a command called `\gls<field>accsupp`. If there is then use that otherwise use `\glsaccsupp`. The first argument should be the internal field label (not the key). The final argument is the entry label. If glossaries-extra has been loaded, this first checks for `\glsxtr<category><field>accsupp` and `\glsxtr<category>accsupp`.

```
10872 \newcommand{\glsfieldaccsupp}[4]{%
10873   \ifdef\glscategory
10874   {%
10875     \ifcsdef{glsxtr\glscategory{#4}#3accsupp}%
10876     {\csname glsxtr\glscategory{#4}#3accsupp\endcsname{#1}{#2}}%
10877     {%
10878       \ifcsdef{glsxtr\glscategory{#4}accsupp}%
10879       {\csname glsxtr\glscategory{#4}accsupp\endcsname{#1}{#2}}%
10880       {%
10881         \ifcsdef{gls#3accsupp}%
10882         {\csname gls#3accsupp\endcsname{#1}{#2}}%
10883         {\glsaccsupp{#1}{#2}}%
10884       }%
10885     }%
10886   }%
10887   {%
10888     \ifcsdef{gls#3accsupp}%
10889     {\csname gls#3accsupp\endcsname{#1}{#2}}%
10890     {\glsaccsupp{#1}{#2}}%
10891   }%
10892 }
```

glsfieldaccsupp

`\xglsfieldaccsupp{<replacement>}{<content>}{<field>}{<label>}`

As `\glsfieldaccsupp` but fully expand replacement text.

```
10893 \newcommand{\xglsfieldaccsupp}[1]{%
10894   \protected@edef\@gls@replacementtext{#1}%
10895   \expandafter\glsfieldaccsupp\expandafter{\@gls@replacementtext}%
10896 }
```

glsshortaccsupp

`\glsshortaccsupp{<replacement text>}{<text>}`

```
10897 \newcommand*{\glsshortaccsupp}[2]{\glsaccessibility{E}{#1}{#2}}
```

`\glsshortplaccsupp` `\glsshortplaccsupp{<replacement text>}{<text>}`

```
10898 \newcommand*{\glsshortplaccsupp}{\glsshortaccsupp}
```

`\glsaccsupp` `\glsaccsupp{<replacement text>}{<text>}`

```
10899 \newcommand*{\glsaccsupp}[2]{\glsaccessibility{ActualText}{#1}{#2}}
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
10900 \newcommand*{\xglsaccsupp}[2]{%
10901   \protected@edef\@gls@replacementtext{#1}%
10902   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10903 }
```

`@access@display` Deprecated. Use `\@gls@fieldaccess@display` instead.

```
10904 \newcommand*{\@gls@access@display}[2]{%
10905   \protected@edef\@glo@access{#2}%
10906   \ifx\@glo@access\@gls@noaccess
10907     #1%
10908   \else
10909     \xglsaccsupp{\@glo@access}{#1}%
10910   \fi
10911 }
```

`daccess@display` `\@gls@fieldaccess@display{<label>}{<field>}{<content>}{<replacement>}`

```
10912 \newcommand*{\@gls@fieldaccess@display}[4]{%
10913   \protected@edef\@glo@access{#4}%
10914   \ifdefequal\@glo@access\@gls@noaccess
10915     {#3}%
10916   {\expandafter\glsfieldaccsupp\expandafter{\@glo@access}{#3}{#2}{#1}}%
10917 }
```

`meaccessdisplay` Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10918 \newrobustcmd*{\glsnameaccessdisplay}[2]{%
10919   \ifcsundef{glo@glsdetoklabel{#2}@access}%
10920     {#1}%
10921   {\@gls@fieldaccess@display{#2}{name}{#1}{\glsentryaccess{#2}}}%
10922 }
```

`xtaccessdisplay` As above but for the textaccess replacement text.

```
10923 \newrobustcmd*{\glsxtaccessdisplay}[2]{%
10924   \ifcsundef{glo@glsdetoklabel{#2}@textaccess}%
10925     {#1}%
10926 }
```

```

10926 {\@gls@fieldaccess@display{#2}{text}{#1}{\glsentrytextaccess{#2}}}%
10927 }

alaccessdisplay As above but for the pluralaccess replacement text.
10928 \newrobustcmd*{\glspluralaccessdisplay}[2]{%
10929 \ifcsundef{glo@glstdetoklabel{#2}@pluralaccess}%
10930 {#1}%
10931 {\@gls@fieldaccess@display{#2}{plural}{#1}{\glsentrypluralaccess{#2}}}%
10932 }

staccessdisplay As above but for the firstaccess replacement text.
10933 \newrobustcmd*{\glsfirstaccessdisplay}[2]{%
10934 \ifcsundef{glo@glstdetoklabel{#2}@firstaccess}%
10935 {#1}%
10936 {\@gls@fieldaccess@display{#2}{first}{#1}{\glsentryfirstaccess{#2}}}%
10937 }

alaccessdisplay As above but for the firstpluralaccess replacement text.
10938 \newrobustcmd*{\glsfirstpluralaccessdisplay}[2]{%
10939 \ifcsundef{glo@glstdetoklabel{#2}@firstpluralaccess}%
10940 {#1}%
10941 {\@gls@fieldaccess@display{#2}{firstpl}{#1}{\glsentryfirstpluralaccess{#2}}}%
10942 }

olaccessdisplay As above but for the symbolaccess replacement text.
10943 \newrobustcmd*{\glsymbolaccessdisplay}[2]{%
10944 \ifcsundef{glo@glstdetoklabel{#2}@symbolaccess}%
10945 {#1}%
10946 {\@gls@fieldaccess@display{#2}{symbol}{#1}{\glsentrysymbolaccess{#2}}}%
10947 }

alaccessdisplay As above but for the symbolpluralaccess replacement text.
10948 \newrobustcmd*{\glsymbolpluralaccessdisplay}[2]{%
10949 \ifcsundef{glo@glstdetoklabel{#2}@symbolpluralaccess}%
10950 {#1}%
10951 {\@gls@fieldaccess@display{#2}{symbolplural}{#1}{\glsentrysymbolpluralaccess{#2}}}%
10952 }

onaccessdisplay As above but for the descriptionaccess replacement text.
10953 \newrobustcmd*{\glsdescriptionaccessdisplay}[2]{%
10954 \ifcsundef{glo@glstdetoklabel{#2}@descaccess}%
10955 {#1}%
10956 {\@gls@fieldaccess@display{#2}{desc}{#1}{\glsentrydescaccess{#2}}}%
10957 }

alaccessdisplay As above but for the descriptionpluralaccess replacement text.
10958 \newrobustcmd*{\glsdescriptionpluralaccessdisplay}[2]{%
10959 \ifcsundef{glo@glstdetoklabel{#2}@descpluralaccess}%

```

```

10960 {#1}%
10961 {\@gls@fieldaccess@display{#2}{descplural}{#1}{\glentrydescpluralaccess{#2}}}%
10962 }

rtaccessdisplay As above but for the shortaccess replacement text.
10963 \newrobustcmd*{\glsshortaccessdisplay}[2]{%
10964 \ifcsundef{glo@glsdetoklabel{#2}@shortaccess}%
10965 {#1}%
10966 {\@gls@fieldaccess@display{#2}{short}{#1}{\glentryshortaccess{#2}}}%
10967 }

alaccessdisplay As above but for the shortpluralaccess replacement text.
10968 \newrobustcmd*{\glsshortpluralaccessdisplay}[2]{%
10969 \ifcsundef{glo@glsdetoklabel{#2}@shortpluralaccess}%
10970 {#1}%
10971 {\@gls@fieldaccess@display{#2}{shortpl}{#1}{\glentryshortpluralaccess{#2}}}%
10972 }

ngaccessdisplay As above but for the longaccess replacement text.
10973 \newrobustcmd*{\glslongaccessdisplay}[2]{%
10974 \ifcsundef{glo@glsdetoklabel{#2}@longaccess}%
10975 {#1}%
10976 {\@gls@fieldaccess@display{#2}{long}{#1}{\glentrylongaccess{#2}}}%
10977 }

alaccessdisplay As above but for the longpluralaccess replacement text.
10978 \newrobustcmd*{\glslongpluralaccessdisplay}[2]{%
10979 \ifcsundef{glo@glsdetoklabel{#2}@longpluralaccess}%
10980 {#1}%
10981 {\@gls@fieldaccess@display{#2}{longpl}{#1}{\glentrylongpluralaccess{#2}}}%
10982 }

riaccessdisplay As above but for the user1access replacement text.
10983 \newrobustcmd*{\gluseriaccessdisplay}[2]{%
10984 \ifcsundef{glo@glsdetoklabel{#2}@useriaccess}%
10985 {#1}%
10986 {\@gls@fieldaccess@display{#2}{useri}{#1}{\glentryuseriaccess{#2}}}%
10987 }

iiaccessdisplay As above but for the user2access replacement text.
10988 \newrobustcmd*{\gluseriiaccessdisplay}[2]{%
10989 \ifcsundef{glo@glsdetoklabel{#2}@useriiaccess}%
10990 {#1}%
10991 {\@gls@fieldaccess@display{#2}{userii}{#1}{\glentryuseriiaccess{#2}}}%
10992 }

iiiaccessdisplay As above but for the user3access replacement text.
10993 \newrobustcmd*{\gluseriiiaccessdisplay}[2]{%

```

```

10994 \ifcsundef{glo@glsdetoklabel{#2}@useriiiaccess}%
10995 {#1}%
10996 {\@gls@fieldaccess@display{#2}{useriii}{#1}{\glsentryuseriiiaccess{#2}}}%
10997 }

```

ivaccessdisplay As above but for the user4access replacement text.

```

10998 \newrobustcmd*{\glsuserivaccessdisplay}[2]{%
10999 \ifcsundef{glo@glsdetoklabel{#2}@userivaccess}%
11000 {#1}%
11001 {\@gls@fieldaccess@display{#2}{useriv}{#1}{\glsentryuserivaccess{#2}}}%
11002 }

```

rvaccessdisplay As above but for the user5access replacement text.

```

11003 \newrobustcmd*{\glsuservaccessdisplay}[2]{%
11004 \ifcsundef{glo@glsdetoklabel{#2}@uservaccess}%
11005 {#1}%
11006 {\@gls@fieldaccess@display{#2}{userv}{#1}{\glsentryuservaccess{#2}}}%
11007 }

```

viaccessdisplay As above but for the user6access replacement text.

```

11008 \newrobustcmd*{\glsuserviaccessdisplay}[2]{%
11009 \ifcsundef{glo@glsdetoklabel{#2}@userviaccess}%
11010 {#1}%
11011 {\@gls@fieldaccess@display{#2}{uservi}{#1}{\glsentryuserviaccess{#2}}}%
11012 }

```

lsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

11013 \newrobustcmd*{\glsaccessdisplay}[3]{%
11014 \ifcsundef{gls#1accessdisplay}%
11015 {%
11016 \PackageError{glossaries-accsupp}{No accessibility support
11017 for key ‘#1’}{}%
11018 }%
11019 {%
11020 \csname gls#1accessdisplay\endcsname{#2}{#3}%
11021 }%
11022 }

```

default@entryfmt Redefine the default entry format to use accessibility information

```

11023 \renewcommand*{\@gls@default@entryfmt}[2]{%
11024 \ifdefempty{glscustomtext}
11025 {%
11026 \glsifplural
11027 {%
11028 \glscapscase
11029 {%

```


Don't adjust case

```
11030      \ifglsused\glslabel
11031      {%
```

Subsequent use

```
11032      #2{\glspluralaccessdisplay
11033          {\glsentryplural{\glslabel}}{\glslabel}}%
11034      {\glsdescriptionpluralaccessdisplay
11035          {\glsentrydescplural{\glslabel}}{\glslabel}}%
11036      {\glsymbolpluralaccessdisplay
11037          {\glsentrysymbolplural{\glslabel}}{\glslabel}}
11038      {\glsinsert}%
11039      }%
11040      {%
```

First use

```
11041      #1{\glsfirstpluralaccessdisplay
11042          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
11043      {\glsdescriptionpluralaccessdisplay
11044          {\glsentrydescplural{\glslabel}}{\glslabel}}%
11045      {\glsymbolpluralaccessdisplay
11046          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
11047      {\glsinsert}%
11048      }%
11049      }%
11050      {%
```

Make first letter upper case

```
11051      \ifglsused\glslabel
11052      {%
```

Subsequent use.

```
11053      #2{\glspluralaccessdisplay
11054          {\Glsentryplural{\glslabel}}{\glslabel}}%
11055      {\glsdescriptionpluralaccessdisplay
11056          {\Glsentrydescplural{\glslabel}}{\glslabel}}%
11057      {\glsymbolpluralaccessdisplay
11058          {\Glsentrysymbolplural{\glslabel}}{\glslabel}}%
11059      {\glsinsert}%
11060      }%
11061      {%
```

First use

```
11062      #1{\glsfirstpluralaccessdisplay
11063          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
11064      {\glsdescriptionpluralaccessdisplay
11065          {\Glsentrydescplural{\glslabel}}{\glslabel}}%
11066      {\glsymbolpluralaccessdisplay
11067          {\Glsentrysymbolplural{\glslabel}}{\glslabel}}%
11068      {\glsinsert}%
11069      }%
```

11070 }%
 11071 {%

Make all upper case

11072 \ifglused\glslabel
 11073 {%

Subsequent use

11074 \MakeUppercase{%
 11075 #2{\glsppluralaccessdisplay
 11076 {\glentryplural{\glslabel}}{\glslabel}}%
 11077 {\gldescriptionpluralaccessdisplay
 11078 {\glentrydescplural{\glslabel}}{\glslabel}}%
 11079 {\glssymbolpluralaccessdisplay
 11080 {\glentrysymbolplural{\glslabel}}{\glslabel}}%
 11081 {\glinsert}}%
 11082 }%
 11083 {%

First use

11084 \MakeUppercase{%
 11085 #1{\glfirstpluralaccessdisplay
 11086 {\glentryfirstplural{\glslabel}}{\glslabel}}%
 11087 {\gldescriptionpluralaccessdisplay
 11088 {\glentrydescplural{\glslabel}}{\glslabel}}%
 11089 {\glssymbolpluralaccessdisplay
 11090 {\glentrysymbolplural{\glslabel}}{\glslabel}}%
 11091 {\glinsert}}%
 11092 }%
 11093 }%
 11094 }%
 11095 {%

Singular form

11096 \glscapscase
 11097 {%

Don't adjust case

11098 \ifglused\glslabel
 11099 {%

Subsequent use

11100 #2{\glstextaccessdisplay
 11101 {\glentrytext{\glslabel}}{\glslabel}}%
 11102 {\gldescriptionaccessdisplay
 11103 {\glentrydesc{\glslabel}}{\glslabel}}%
 11104 {\glssymbolaccessdisplay
 11105 {\glentrysymbol{\glslabel}}{\glslabel}}%
 11106 {\glinsert}}%
 11107 }%
 11108 {%

First use

```

11109      #1{\glsfirstaccessdisplay
11110          {\glsentryfirst{\glslabel}}{\glslabel}}%
11111          {\glsdescriptionaccessdisplay
11112          {\glsentrydesc{\glslabel}}{\glslabel}}%
11113          {\glsymbolaccessdisplay
11114          {\glsentrysymbol{\glslabel}}{\glslabel}}%
11115          {\glsinsert}}%
11116      }%
11117  }%
11118  {%

```

Make first letter upper case

```

11119      \ifglsused\glslabel
11120      {%

```

Subsequent use

```

11121      #2{\glstextaccessdisplay
11122          {\Glsentrytext{\glslabel}}{\glslabel}}%
11123          {\glsdescriptionaccessdisplay
11124          {\glsentrydesc{\glslabel}}{\glslabel}}%
11125          {\glsymbolaccessdisplay
11126          {\glsentrysymbol{\glslabel}}{\glslabel}}%
11127          {\glsinsert}}%
11128      }%
11129      {%

```

First use

```

11130      #1{\glsfirstaccessdisplay
11131          {\Glsentryfirst{\glslabel}}{\glslabel}}%
11132          {\glsdescriptionaccessdisplay
11133          {\glsentrydesc{\glslabel}}{\glslabel}}%
11134          {\glsymbolaccessdisplay
11135          {\glsentrysymbol{\glslabel}}{\glslabel}}%
11136          {\glsinsert}}%
11137      }%
11138  }%
11139  {%

```

Make all upper case

```

11140      \ifglsused\glslabel
11141      {%

```

Subsequent use

```

11142      \MakeUppercase{%
11143      #2{\glstextaccessdisplay
11144          {\glsentrytext{\glslabel}}{\glslabel}}%
11145          {\glsdescriptionaccessdisplay
11146          {\glsentrydesc{\glslabel}}{\glslabel}}%
11147          {\glsymbolaccessdisplay
11148          {\glsentrysymbol{\glslabel}}{\glslabel}}%

```

```

11149         {\glsinsert}}}%
11150     }%
11151     {%

```

First use

```

11152     \MakeUppercase{%
11153         #1{\glsfirstaccessdisplay
11154             {\glsentryfirst{\glslabel}}{\glslabel}}}%
11155         {\glsdescriptionaccessdisplay
11156             {\glsentrydesc{\glslabel}}{\glslabel}}}%
11157         {\glsymbolaccessdisplay
11158             {\glsentrysymbol{\glslabel}}{\glslabel}}}%
11159         {\glsinsert}}}%
11160     }%
11161 }%
11162 }%
11163 }%
11164 {%

```

Custom text provided in \glsdisp

```

11165     \ifglsused{\glslabel}%
11166     {%

```

Subsequent use

```

11167         #2{\glscustomtext}%
11168         {\glsdescriptionaccessdisplay
11169             {\glsentrydesc{\glslabel}}{\glslabel}}}%
11170         {\glsymbolaccessdisplay
11171             {\glsentrysymbol{\glslabel}}{\glslabel}}}%
11172         {\glsinsert}}}%
11173     }%
11174     {%

```

First use

```

11175         #1{\glscustomtext}%
11176         {\glsdescriptionaccessdisplay
11177             {\glsentrydesc{\glslabel}}{\glslabel}}}%
11178         {\glsymbolaccessdisplay
11179             {\glsentrysymbol{\glslabel}}{\glslabel}}}%
11180         {\glsinsert}}}%
11181     }%
11182 }%
11183 }

```

\glsгенentryfmt Redefine to use accessibility information.

```

11184 \renewcommand*{\glsгенentryfmt}{%
11185     \ifdefempty\glscustomtext
11186     {%
11187         \glsifplural
11188         {%

```

Plural form

```
11189      \glscapscase
11190      {%
```

Don't adjust case

```
11191      \ifglused\glslabel
11192      {%
```

Subsequent use

```
11193      \glspluralaccessdisplay
11194      {\glentryplural{\glslabel}}{\glslabel}%
11195      \glsinsert
11196      }%
11197      {%
```

First use

```
11198      \glsfirstpluralaccessdisplay
11199      {\glentryfirstplural{\glslabel}}{\glslabel}%
11200      \glsinsert
11201      }%
11202      }%
11203      {%
```

Make first letter upper case

```
11204      \ifglused\glslabel
11205      {%
```

Subsequent use.

```
11206      \glspluralaccessdisplay
11207      {\Glsentryplural{\glslabel}}{\glslabel}%
11208      \glsinsert
11209      }%
11210      {%
```

First use

```
11211      \glsfirstpluralaccessdisplay
11212      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
11213      \glsinsert
11214      }%
11215      }%
11216      {%
```

Make all upper case

```
11217      \ifglused\glslabel
11218      {%
```

Subsequent use

```
11219      \glspluralaccessdisplay
11220      {\mfirstucMakeUppercase{\glentryplural{\glslabel}}}%
11221      {\glslabel}%
11222      \mfirstucMakeUppercase{\glsinsert}%
11223      }%
11224      {%
```

First use

```
11225      \glsfirstpluralaccessdisplay
11226      {\mfirstucMakeUppercase{\glsentryfirstplural{\glslabel}}}%
11227      {\glslabel}%
11228      \mfirstucMakeUppercase{\glsinsert}%
11229      }%
11230      }%
11231      }%
11232      {%
```

Singular form

```
11233      \glscapscase
11234      {%
```

Don't adjust case

```
11235      \ifglused\glslabel
11236      {%
```

Subsequent use

```
11237      \glstextaccessdisplay{\glsentrytext{\glslabel}}{\glslabel}%
11238      \glsinsert
11239      }%
11240      {%
```

First use

```
11241      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
11242      \glsinsert
11243      }%
11244      }%
11245      {%
```

Make first letter upper case

```
11246      \ifglused\glslabel
11247      {%
```

Subsequent use

```
11248      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
11249      \glsinsert
11250      }%
11251      {%
```

First use

```
11252      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
11253      \glsinsert
11254      }%
11255      }%
11256      {%
```

Make all upper case

```
11257      \ifglused\glslabel
11258      {%
```

Subsequent use

```

11259      \glstextaccessdisplay
11260      {\mfirstucMakeUppercase{\glentrytext{\glslabel}}}{\glslabel}%
11261      \mfirstucMakeUppercase{\glsinsert}%
11262      }%
11263      {%

```

First use

```

11264      \glsfirstaccessdisplay
11265      {\mfirstucMakeUppercase{\glentryfirst{\glslabel}}}{\glslabel}%
11266      \mfirstucMakeUppercase{\glsinsert}%
11267      }%
11268      }%
11269      }%
11270      }%
11271      {%

```

Custom text provided in `\glsdisp`. (The insert should be empty at this point.) The accessibility information, if required, will have to be explicitly included in the custom text.

```

11272      \glscustomtext\glsinsert
11273      }%
11274      }

```

`\glsgenacfmt` Redefine to include accessibility information.

```

11275 \renewcommand*{\glsgenacfmt}{%
11276   \ifdefempty\glscustomtext
11277   {%
11278     \ifglsused\glslabel
11279     {%

```

Subsequent use:

```

11280      \glsifplural
11281      {%

```

Subsequent plural form:

```

11282      \glscapscase
11283      {%

```

Subsequent plural form, don't adjust case:

```

11284      \acronymfont
11285      {\glsshortpluralaccessdisplay
11286       {\glentryshortpl{\glslabel}}{\glslabel}}%
11287      \glsinsert
11288      }%
11289      {%

```

Subsequent plural form, make first letter upper case:

```

11290      \acronymfont
11291      {\glsshortpluralaccessdisplay
11292       {\Glsentryshortpl{\glslabel}}{\glslabel}}%
11293      \glsinsert

```

11294 }%

11295 {%

Subsequent plural form, all caps:

11296 \mfirstucMakeUppercase

11297 {\acronymfont

11298 {\glsshortpluralaccessdisplay

11299 {\glentryshortpl{\glslabel}}{\glslabel}}%

11300 \glsinsert}%

11301 }%

11302 }%

11303 {%

Subsequent singular form

11304 \glscapscase

11305 {%

Subsequent singular form, don't adjust case:

11306 \acronymfont

11307 {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%

11308 \glsinsert

11309 }%

11310 {%

Subsequent singular form, make first letter upper case:

11311 \acronymfont

11312 {\glsshortaccessdisplay{\Glentryshort{\glslabel}}{\glslabel}}%

11313 \glsinsert

11314 }%

11315 {%

Subsequent singular form, all caps:

11316 \mfirstucMakeUppercase

11317 {\acronymfont{%

11318 \glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%

11319 \glsinsert}%

11320 }%

11321 }%

11322 }%

11323 {%

First use:

11324 \glsifplural

11325 {%

First use plural form:

11326 \glscapscase

11327 {%

First use plural form, don't adjust case:

11328 \genplacrformat{\glslabel}{\glsinsert}%

11329 }%

11330 {%

First use plural form, make first letter upper case:

```
11331      \Genplacrfullformat{\glslabel}{\glsinsert}%
11332      }%
11333      {%
```

First use plural form, all caps:

```
11334      \mfirstucMakeUppercase
11335      {\genplacrfullformat{\glslabel}{\glsinsert}}%
11336      }%
11337      }%
11338      {%
```

First use singular form

```
11339      \glscapscase
11340      {%
```

First use singular form, don't adjust case:

```
11341      \genacrfullformat{\glslabel}{\glsinsert}%
11342      }%
11343      {%
```

First use singular form, make first letter upper case:

```
11344      \Genacrfullformat{\glslabel}{\glsinsert}%
11345      }%
11346      {%
```

First use singular form, all caps:

```
11347      \mfirstucMakeUppercase
11348      {\genacrfullformat{\glslabel}{\glsinsert}}%
11349      }%
11350      }%
11351      }%
11352      }%
11353      {%
```

User supplied text. (The insert should be empty at this point.) The accessibility information, if required, will have to be explicitly included in the custom text.

```
11354      \glscustomtext
11355      }%
11356 }
```

`\genacrfullformat` Redefine to include accessibility information.

```
11357 \renewcommand*{\genacrfullformat}[2]{%
11358   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
11359   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}{#1}})%
11360 }
```

`\genacrfullformat` Redefine to include accessibility information.

```
11361 \renewcommand*{\Genacrfullformat}[2]{%
11362   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
11363   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}{#1}})%
11364 }
```

placrfullformat Redefine to include accessibility information.

```
11365 \renewcommand*{\genplacrfullformat}[2]{%
11366   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
11367   (\glsshortpluralaccessdisplay
11368     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
11369 }
```

placrfullformat Redefine to include accessibility information.

```
11370 \renewcommand*{\Genplacrfullformat}[2]{%
11371   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
11372   (\glsshortpluralaccessdisplay
11373     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1})%
11374 }
```

\@acrshort

```
11375 \def\@acrshort#1#2[#3]{%
11376   \glsdoifexists{#2}%
11377   {%
11378     \let\do@gls@link@checkfirsthyper\relax

11379     \let\glsifplural\@secondoftwo
11380     \let\glscapscase\@firstofthree
11381     \let\glsinsert\@empty
11382     \def\glscustomtext{%
11383       \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
11384     }%

    Call \@gls@link

11385     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11386   }%

11387   \glspostlinkhook
11388 }
```

\@Acrshort

```
11389 \def\@Acrshort#1#2[#3]{%
11390   \glsdoifexists{#2}%
11391   {%
11392     \let\do@gls@link@checkfirsthyper\relax

11393     \let\glsifplural\@secondoftwo
11394     \let\glscapscase\@secondofthree
11395     \let\glsinsert\@empty
11396     \def\glscustomtext{%
11397       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
11398     }%

    Call \@gls@link

11399     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11400   }%
```

```

11401 \glspostlinkhook
11402 }

```

\@ACRshort

```

11403 \def\@ACRshort#1#2[#3]{%
11404   \glsdoidexists{#2}%
11405   {%
11406     \let\do@gl@link@checkfirsthyper\relax

11407     \let\gl@sifplural\@secondoftwo
11408     \let\gl@scapscase\@thirdofthree
11409     \let\gl@insert\@empty
11410     \def\glscustomtext{%
11411       \acronymfont{\glsshortaccessdisplay
11412         {\MakeUppercase{\glentryshort{#2}}}{#2}}#3%
11413     }%

    Call\@gl@link

11414     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11415     }%

11416 \glspostlinkhook
11417 }

```

\@acrlong

```

11418 \def\@acrlong#1#2[#3]{%
11419   \glsdoidexists{#2}%
11420   {%
11421     \let\do@gl@link@checkfirsthyper\relax

11422     \let\gl@sifplural\@secondoftwo
11423     \let\gl@scapscase\@firstofthree
11424     \let\gl@insert\@empty
11425     \def\glscustomtext{%
11426       \acronymfont{\glslongaccessdisplay{\glentrylong{#2}}{#2}}#3%
11427     }%

    Call\@gl@link

11428     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11429     }%

11430 \glspostlinkhook
11431 }

```

\@Acrlong

```

11432 \def\@Acrlong#1#2[#3]{%
11433   \glsdoidexists{#2}%
11434   {%
11435     \let\do@gl@link@checkfirsthyper\relax

```

```

11436 \let\glsifplural\@secondoftwo
11437 \let\glschapscase\@firstofthree
11438 \let\glsinsert\@empty
11439 \def\glscustomtext{%
11440 \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
11441 }%

```

Call \@gls@link

```

11442 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11443 }%

11444 \glspostlinkhook
11445 }

```

\@ACRlong

```

11446 \def\@ACRlong#1#2[#3]{%
11447 \glsdoifexists{#2}%
11448 {%
11449 \let\do@gls@link@checkfirsthyper\relax

11450 \let\glsifplural\@secondoftwo
11451 \let\glschapscase\@firstofthree
11452 \let\glsinsert\@empty
11453 \def\glscustomtext{%
11454 \acronymfont{\glslongaccessdisplay{%
11455 \MakeUppercase{\Glsentrylong{#2}}}{#2}}#3}%
11456 }%

```

Call \@gls@link

```

11457 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11458 }%

11459 \glspostlinkhook
11460 }

```

\@glstext@

```

11461 \def\@glstext@#1#2[#3]{%
11462 \@gls@field@link{#1}{#2}{\glstextaccessdisplay{\Glsentrytext{#2}}{#2}}#3}%
11463 }

```

\@Glstext@

```

11464 \def\@Glstext@#1#2[#3]{%
11465 \@gls@field@link{#1}{#2}{\glstextaccessdisplay{\Glsentrytext{#2}}{#2}}#3}%
11466 }

```

\@GLStext@

```

11467 \def\@GLStext@#1#2[#3]{%
11468 \@gls@field@link{#1}{#2}%
11469 {\glstextaccessdisplay{\mfirstucMakeUppercase{\Glsentrytext{#2}}}{#2}%
11470 \mfirstucMakeUppercase{#3}}}%
11471 }

```

```

\@glsfirst@
11472 \def\@glsfirst@#1#2[#3]{%
11473   \@gls@field@link{#1}{#2}{\glsfirstaccessdisplay{\glentryfirst{#2}}{#2}#3}%
11474 }

\@Glsfirst@
11475 \def\@Glsfirst@#1#2[#3]{%
11476   \@gls@field@link{#1}{#2}{\glsfirstaccessdisplay{\Glsentryfirst{#2}}{#2}#3}%
11477 }

\@GLSfirst@
11478 \def\@GLSfirst@#1#2[#3]{%
11479   \@gls@field@link{#1}{#2}%
11480   {\glsfirstaccessdisplay{\mfirstucMakeUppercase{\glentryfirst{#2}}}{#2}%
11481     \mfirstucMakeUppercase{#3}}%
11482 }

\@glsplural@
11483 \def\@glsplural@#1#2[#3]{%
11484   \@gls@field@link{#1}{#2}{\glspluralaccessdisplay{\glentryplural{#2}}{#2}#3}%
11485 }

\@Glsplural@
11486 \def\@Glsplural@#1#2[#3]{%
11487   \@gls@field@link{#1}{#2}{\glspluralaccessdisplay{\Glsentryplural{#2}}{#2}#3}%
11488 }

\@GLSplural@
11489 \def\@GLSplural@#1#2[#3]{%
11490   \@gls@field@link{#1}{#2}%
11491   {\glspluralaccessdisplay{\mfirstucMakeUppercase{\glentryplural{#2}}}{#2}%
11492     \mfirstucMakeUppercase{#3}}%
11493 }

glsfirstplural@
11494 \def\@glsfirstplural@#1#2[#3]{%
11495   \@gls@field@link{#1}{#2}{\glsfirstpluralaccessdisplay{\glentryfirstplural{#2}}{#2}#3}%
11496 }

Glsfirstplural@
11497 \def\@glsfirstplural@#1#2[#3]{%
11498   \@gls@field@link{#1}{#2}{\glsfirstpluralaccessdisplay{\Glsentryfirstplural{#2}}{#2}#3}%
11499 }

GLSfirstplural@
11500 \def\@GLSfirstplural@#1#2[#3]{%
11501   \@gls@field@link{#1}{#2}%
11502   {\glsfirstpluralaccessdisplay{\mfirstucMakeUppercase{\glentryfirstplural{#2}}}{#2}%

```

```

11503     \mfirstucMakeUppercase{#3}}}%
11504 }

\@glsname@
11505 \def\@glsname@#1#2[#3]{%
11506   \@gls@field@link{#1}{#2}{\glsnameaccessdisplay{\glsentryname{#2}}{#2}#3}%
11507 }

\@Glsname@
11508 \def\@Glsname@#1#2[#3]{%
11509   \@gls@field@link{#1}{#2}{\glsnameaccessdisplay{\Glsentryname{#2}}{#2}#3}%
11510 }

\@GLSname@
11511 \def\@GLSname@#1#2[#3]{%
11512   \@gls@field@link{#1}{#2}%
11513   {\glsnameaccessdisplay{\mfirstucMakeUppercase{\glsentryname{#2}}}{#2}%
11514     \mfirstucMakeUppercase{#3}}}%
11515 }

\@glsdesc@
11516 \def\@glsdesc@#1#2[#3]{%
11517   \@gls@field@link{#1}{#2}{\glsdescriptionaccessdisplay{\glsentrydesc{#2}}{#2}#3}%
11518 }

\@Glsdesc@
11519 \def\@Glsdesc@#1#2[#3]{%
11520   \@gls@field@link{#1}{#2}{\glsdescriptionaccessdisplay{\Glsentrydesc{#2}}{#2}#3}%
11521 }

\@GLSdesc@
11522 \def\@GLSdesc@#1#2[#3]{%
11523   \@gls@field@link{#1}{#2}%
11524   {\glsdescriptionaccessdisplay{\mfirstucMakeUppercase{\glsentrydesc{#2}}}{#2}%
11525     \mfirstucMakeUppercase{#3}}}%
11526 }

@glsdescplural@
11527 \def\@glsdescplural@#1#2[#3]{%
11528   \@gls@field@link{#1}{#2}{\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}#3}%
11529 }

@Glsdescplural@
11530 \def\@Glsdescplural@#1#2[#3]{%
11531   \@gls@field@link{#1}{#2}{\glsdescriptionpluralaccessdisplay{\Glsentrydescplural{#2}}{#2}#3}%
11532 }

```

@GLSdescplural@

```
11533 \def\@GLSdescplural@#1#2[#3]{%
11534   \@gls@field@link{#1}{#2}%
11535   {\glsdescriptionpluralaccessdisplay{\mfirstucMakeUppercase{\glsentrydescplural{#2}}}{#2}%
11536   \mfirstucMakeUppercase{#3}}%
11537 }
```

\@glssymbol@

```
11538 \def\@glssymbol@#1#2[#3]{%
11539   \@gls@field@link{#1}{#2}{\glssymbolaccessdisplay{\glsentrysymbol{#2}}{#2}#3}%
11540 }
```

\@GLssymbol@

```
11541 \def\@GLssymbol@#1#2[#3]{%
11542   \@gls@field@link{#1}{#2}{\glssymbolaccessdisplay{\Glsentrysymbol{#2}}{#2}#3}%
11543 }
```

\@GLSsymbol@

```
11544 \def\@GLSsymbol@#1#2[#3]{%
11545   \@gls@field@link{#1}{#2}%
11546   {\glssymbolaccessdisplay{\mfirstucMakeUppercase{\glsentrysymbol{#2}}}{#2}%
11547   \mfirstucMakeUppercase{#3}}%
11548 }
```

lssymbolplural@

```
11549 \def\@glssymbolplural@#1#2[#3]{%
11550   \@gls@field@link{#1}{#2}{\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}#3}%
11551 }
```

lssymbolplural@

```
11552 \def\@Glssymbolplural@#1#2[#3]{%
11553   \@gls@field@link{#1}{#2}{\glssymbolpluralaccessdisplay{\Glsentrysymbolplural{#2}}{#2}#3}%
11554 }
```

LSSymbolplural@

```
11555 \def\@GLSsymbolplural@#1#2[#3]{%
11556   \@gls@field@link{#1}{#2}%
11557   {\glssymbolpluralaccessdisplay{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}}}{#2}%
11558   \mfirstucMakeUppercase{#3}}%
11559 }
```

\@glsuseri@

```
11560 \def\@glsuseri@#1#2[#3]{%
11561   \@gls@field@link{#1}{#2}{\glsuseriaccessdisplay{\glsentryuseri{#2}}{#2}#3}%
11562 }
```

\@Glsuseri@

```
11563 \def \@Glsuser@i#1#2[#3]{%
11564   \@gls@field@link{#1}{#2}{\glsuseriaccessdisplay{\Glsentryuseri{#2}}{#2}#3}%
11565 }
```

\@GLSuseri@

```
11566 \def \@GLSuseri@#1#2[#3]{%
11567   \@gls@field@link{#1}{#2}%
11568   {\glsuseriaccessdisplay{\mfirstucMakeUppercase{\glsentryuseri{#2}}}{#2}%
11569     \mfirstucMakeUppercase{#3}}%
11570 }
```

\@glsuserii@

```
11571 \def \@glsuserii@#1#2[#3]{%
11572   \@gls@field@link{#1}{#2}{\glsuseriiaccessdisplay{\glsentryuserii{#2}}{#2}#3}%
11573 }
```

\@Glsuserii@

```
11574 \def \@Glsuser@i#1#2[#3]{%
11575   \@gls@field@link{#1}{#2}{\glsuseriiaccessdisplay{\Glsentryuserii{#2}}{#2}#3}%
11576 }
```

\@GLSuserii@

```
11577 \def \@GLSuserii@#1#2[#3]{%
11578   \@gls@field@link{#1}{#2}%
11579   {\glsuseriiaccessdisplay{\mfirstucMakeUppercase{\glsentryuserii{#2}}}{#2}%
11580     \mfirstucMakeUppercase{#3}}%
11581 }
```

\@glsuseriii@

```
11582 \def \@glsuseriii@#1#2[#3]{%
11583   \@gls@field@link{#1}{#2}{\glsuseriiiaccessdisplay{\glsentryuseriii{#2}}{#2}#3}%
11584 }
```

\@Glsuseriii@

```
11585 \def \@Glsuser@i#1#2[#3]{%
11586   \@gls@field@link{#1}{#2}{\glsuseriiiaccessdisplay{\Glsentryuseriii{#2}}{#2}#3}%
11587 }
```

\@GLSuseriii@

```
11588 \def \@GLSuseriii@#1#2[#3]{%
11589   \@gls@field@link{#1}{#2}%
11590   {\glsuseriiiaccessdisplay{\mfirstucMakeUppercase{\glsentryuseriii{#2}}}{#2}%
11591     \mfirstucMakeUppercase{#3}}%
11592 }
```



```

\@glsuseriv@
11593 \def\@glsuseriv@#1#2[#3]{%
11594   \@gls@field@link{#1}{#2}{\glsuserivaccessdisplay{\glsentryuseriv{#2}}{#2}#3}%
11595 }

\@Glsuseriv@
11596 \def\@Glsuser@i#1#2[#3]{%
11597   \@gls@field@link{#1}{#2}{\glsuserivaccessdisplay{\Glsentryuseriv{#2}}{#2}#3}%
11598 }

\@GLSuseriv@
11599 \def\@GLSuseriv@#1#2[#3]{%
11600   \@gls@field@link{#1}{#2}%
11601   {\glsuserivaccessdisplay{\mfirstucMakeUppercase{\glsentryuseriv{#2}}{#2}%
11602     \mfirstucMakeUppercase{#3}}}%
11603 }

\@glsuserv@
11604 \def\@glsuserv@#1#2[#3]{%
11605   \@gls@field@link{#1}{#2}{\glsuservaccessdisplay{\glsentryuserv{#2}}{#2}#3}%
11606 }

\@Glsuserv@
11607 \def\@Glsuser@i#1#2[#3]{%
11608   \@gls@field@link{#1}{#2}{\glsuservaccessdisplay{\Glsentryuserv{#2}}{#2}#3}%
11609 }

\@GLSuserv@
11610 \def\@GLSuserv@#1#2[#3]{%
11611   \@gls@field@link{#1}{#2}%
11612   {\glsuservaccessdisplay{\mfirstucMakeUppercase{\glsentryuserv{#2}}{#2}%
11613     \mfirstucMakeUppercase{#3}}}%
11614 }

\@glsuservi@
11615 \def\@glsuservi@#1#2[#3]{%
11616   \@gls@field@link{#1}{#2}{\glsuserviaccessdisplay{\glsentryuservi{#2}}{#2}#3}%
11617 }

\@Glsuservi@
11618 \def\@Glsuser@i#1#2[#3]{%
11619   \@gls@field@link{#1}{#2}{\glsuserviaccessdisplay{\Glsentryuservi{#2}}{#2}#3}%
11620 }

\@GLSuservi@
11621 \def\@GLSuservi@#1#2[#3]{%
11622   \@gls@field@link{#1}{#2}%
11623   {\glsuserviaccessdisplay{\mfirstucMakeUppercase{\glsentryuservi{#2}}{#2}%
11624     \mfirstucMakeUppercase{#3}}}%
11625 }

```

5.3 Displaying the Glossary

We need to redefine the way the glossary entries are formatted to include the accessibility support. The predefined glossary styles use `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol`, but we need to provide compatibility with earlier versions in case users have defined their own styles using `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

Now redefine `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol` etc so they use the accessibility stuff.

```
11626 \renewcommand*{\glossentryname}[1]{%
11627   \glsdoifexists{#1}%
11628   {%
11629     \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
11630   }%
11631 }

11632 \renewcommand*{\glossentryname}[1]{%
11633   \glsdoifexists{#1}%
11634   {%
11635     \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
11636   }%
11637 }

11638 \renewcommand*{\glossentrydesc}[1]{%
11639   \glsdoifexists{#1}%
11640   {%
11641     \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
11642   }%
11643 }

11644 \renewcommand*{\Glossentrydesc}[1]{%
11645   \glsdoifexists{#1}%
11646   {%
11647     \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
11648   }%
11649 }

11650 \renewcommand*{\glossentrysymbol}[1]{%
11651   \glsdoifexists{#1}%
11652   {%
11653     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
11654   }%
11655 }

11656 \renewcommand*{\Glossentrysymbol}[1]{%
11657   \glsdoifexists{#1}%
11658   {%
11659     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
11660   }%
11661 }
```

ssaryentryfield

```

11662 \newcommand*{\accsuppglossaryentryfield}[5]{%
11663   \glossaryentryfield{#1}%
11664   {\glslnameaccessdisplay{#2}{#1}}%
11665   {\glsdescriptionaccessdisplay{#3}{#1}}%
11666   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
11667 }

```

rysubentryfield

```

11668 \newcommand*{\accsuppglossarysubentryfield}[6]{%
11669   \glossarysubentryfield{#1}{#2}%
11670   {\glslnameaccessdisplay{#3}{#2}}%
11671   {\glsdescriptionaccessdisplay{#4}{#2}}%
11672   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
11673 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

11674 \renewacronymstyle{long-short}%
11675 {%

```

Check for long form in case this is a mixed glossary.

```

11676   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11677 }%
11678 {%
11679   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11680   \renewcommand*{\genacrfullformat}[2]{%
11681     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
11682     (\glsshortaccessdisplay
11683       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11684   }%
11685   \renewcommand*{\Genacrfullformat}[2]{%
11686     \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11687     (\glsshortaccessdisplay
11688       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11689   }%
11690   \renewcommand*{\genplacrfullformat}[2]{%
11691     \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11692     (\glsshortpluralaccessdisplay
11693       {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
11694   }%
11695   \renewcommand*{\Genplacrfullformat}[2]{%
11696     \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11697     (\glsshortpluralaccessdisplay
11698       {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
11699   }%
11700   \renewcommand*{\acronymentry}[1]{%

```

```

11701 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11702 \renewcommand*{\acronymsort}[2]{##1}%
11703 \renewcommand*{\acronymfont}[1]{##1}%
11704 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11705 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11706 }

```

short-long *<short>* (*<long>*) acronym style.

```

11707 \renewacronymstyle{short-long}%
11708 {%

```

Check for long form in case this is a mixed glossary.

```

11709 \ifglshaslong{\glslabel}{\glsгенacfmt}{\glsгенentryfmt}%
11710 }%
11711 {%
11712 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11713 \renewcommand*{\genacrfullformat}[2]{%
11714 \glsshortaccessdisplay
11715 {\protect\firstacronymfont{\glentryshort{##1}}}{##1}##2\space
11716 (\glslongaccessdisplay{\glentrylong{##1}}{##1})%
11717 }%
11718 \renewcommand*{\Genacrfullformat}[2]{%
11719 \glsshortaccessdisplay
11720 {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
11721 (\glslongaccessdisplay{\glentrylong{##1}}{##1})%
11722 }%
11723 \renewcommand*{\genplacrfullformat}[2]{%
11724 \glsshortpluralaccessdisplay
11725 {\protect\firstacronymfont{\glentryshortpl{##1}}}{##1}##2\space
11726 (\glslongpluralaccessdisplay
11727 {\glentrylongpl{##1}}{##1})%
11728 }%
11729 \renewcommand*{\Genplacrfullformat}[2]{%
11730 \glsshortpluralaccessdisplay
11731 {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space
11732 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1})%
11733 }%
11734 \renewcommand*{\acronymentry}[1]{%
11735 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}%
11736 \renewcommand*{\acronymsort}[2]{##1}%
11737 \renewcommand*{\acronymfont}[1]{##1}%
11738 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11739 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11740 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

11741 \renewacronymstyle{long-short-desc}%
11742 {%

```

```

11743 \GlsUseAcrEntryDispStyle{long-short}%
11744 }%
11745 {%
11746 \GlsUseAcrStyleDefs{long-short}%
11747 \renewcommand*{\GenericAcronymFields}{}%
11748 \renewcommand*{\acronymsort}[2]{##2}%
11749 \renewcommand*{\acronymentry}[1]{%
11750 \glslongaccessdisplay{\gl Sentrylong{##1}}{##1}\space
11751 (\glsshortaccessdisplay{\acronymfont{\gl Sentryshort{##1}}}{##1))}%
11752 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11753 \renewacronymstyle{long-sc-short-desc}%
11754 {%
11755 \GlsUseAcrEntryDispStyle{long-sc-short}%
11756 }%
11757 {%
11758 \GlsUseAcrStyleDefs{long-sc-short}%
11759 \renewcommand*{\GenericAcronymFields}{}%
11760 \renewcommand*{\acronymsort}[2]{##2}%
11761 \renewcommand*{\acronymentry}[1]{%
11762 \glslongaccessdisplay{\gl Sentrylong{##1}}{##1}\space
11763 (\glsshortaccessdisplay{\acronymfont{\gl Sentryshort{##1}}}{##1))}%
11764 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11765 \renewacronymstyle{long-sm-short-desc}%
11766 {%
11767 \GlsUseAcrEntryDispStyle{long-sm-short}%
11768 }%
11769 {%
11770 \GlsUseAcrStyleDefs{long-sm-short}%
11771 \renewcommand*{\GenericAcronymFields}{}%
11772 \renewcommand*{\acronymsort}[2]{##2}%
11773 \renewcommand*{\acronymentry}[1]{%
11774 \glslongaccessdisplay{\gl Sentrylong{##1}}{##1}\space
11775 (\glsshortaccessdisplay{\acronymfont{\gl Sentryshort{##1}}}{##1))}%
11776 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

11777 \renewacronymstyle{short-long-desc}%
11778 {%
11779 \GlsUseAcrEntryDispStyle{short-long}%
11780 }%
11781 {%
11782 \GlsUseAcrStyleDefs{short-long}%

```

```

11783 \renewcommand*{\GenericAcronymFields}{}%
11784 \renewcommand*{\acronymsort}[2]{##2}%
11785 \renewcommand*{\acronymentry}[1]{%
11786     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11787     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}})%
11788 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11789 \renewacronymstyle{sc-short-long-desc}%
11790 {%
11791     \GlsUseAcrEntryDispStyle{sc-short-long}%
11792 }%
11793 {%
11794     \GlsUseAcrStyleDefs{sc-short-long}%
11795     \renewcommand*{\GenericAcronymFields}{}%
11796     \renewcommand*{\acronymsort}[2]{##2}%
11797     \renewcommand*{\acronymentry}[1]{%
11798         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11799         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}})%
11800 }

```

short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11801 \renewacronymstyle{sm-short-long-desc}%
11802 {%
11803     \GlsUseAcrEntryDispStyle{sm-short-long}%
11804 }%
11805 {%
11806     \GlsUseAcrStyleDefs{sm-short-long}%
11807     \renewcommand*{\GenericAcronymFields}{}%
11808     \renewcommand*{\acronymsort}[2]{##2}%
11809     \renewcommand*{\acronymentry}[1]{%
11810         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11811         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}})%
11812 }

```

dua *<long>* only acronym style.

```

11813 \renewacronymstyle{dua}%
11814 {%

```

Check for long form in case this is a mixed glossary.

```

11815 \ifdefempty\glscustomtext
11816 {%
11817     \ifglshaslong{\glslabel}%
11818     {%
11819         \glsifplural
11820         {%

```

Plural form:

```
11821      \glscapscase
11822      {%
```

Plural form, don't adjust case:

```
11823      \glslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%
11824      \glsinsert
11825      }%
11826      {%
```

Plural form, make first letter upper case:

```
11827      \glslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%
11828      \glsinsert
11829      }%
11830      {%
```

Plural form, all caps:

```
11831      \glslongpluralaccessdisplay
11832      {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%
11833      \mfirstucMakeUppercase{\glsinsert}%
11834      }%
11835      }%
11836      {%
```

Singular form

```
11837      \glscapscase
11838      {%
```

Singular form, don't adjust case:

```
11839      \glslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert
11840      }%
11841      {%
```

Subsequent singular form, make first letter upper case:

```
11842      \glslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert
11843      }%
11844      {%
```

Subsequent singular form, all caps:

```
11845      \glslongaccessdisplay
11846      {\mfirstucMakeUppercase
11847      {\glentrylong{\glslabel}\glsinsert}}{\glslabel}%
11848      \mfirstucMakeUppercase{\glsinsert}%
11849      }%
11850      }%
11851      }%
11852      {%
```

Not an acronym:

```
11853      \glsgenentryfmt
11854      }%
11855      }%
```

```

11856 {\glscustomtext\glsinsert}%
11857 }%
11858 {%
11859 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11860 \renewcommand*{\acrfullfmt}[3]{%
11861   \glslink[##1]{##2}{%
11862     \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11863     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
11864 \renewcommand*{\Acrfullfmt}[3]{%
11865   \glslink[##1]{##2}{%
11866     \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11867     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
11868 \renewcommand*{\ACRfullfmt}[3]{%
11869   \glslink[##1]{##2}{%
11870     \glslongaccessdisplay
11871     {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11872     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
11873 \renewcommand*{\acrfullplfmt}[3]{%
11874   \glslink[##1]{##2}{%
11875     \glslongpluralaccessdisplay
11876     {\glsentrylongpl{##2}}{##2}##3\space
11877     (\glsshortpluralaccessdisplay
11878     {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
11879 \renewcommand*{\Acrfullplfmt}[3]{%
11880   \glslink[##1]{##2}{%
11881     \glslongpluralaccessdisplay
11882     {\Glsentrylongpl{##2}}{##2}##3\space
11883     (\glsshortpluralaccessdisplay
11884     {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
11885 \renewcommand*{\ACRfullplfmt}[3]{%
11886   \glslink[##1]{##2}{%
11887     \glslongpluralaccessdisplay
11888     {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11889     (\glsshortpluralaccessdisplay
11890     {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
11891 \renewcommand*{\glsentryfull}[1]{%
11892   \glslongaccessdisplay{\glsentrylong{##1}}\space
11893   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11894 }%
11895 \renewcommand*{\Glsentryfull}[1]{%
11896   \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11897   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11898 }%
11899 \renewcommand*{\glsentryfullpl}[1]{%
11900   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11901   (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11902 }%
11903 \renewcommand*{\Glsentryfullpl}[1]{%
11904   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space

```



```

11905 (\glsshortpluralaccessdisplay{\acronymfont{\glentryshortpl{##1}}}{##1})%
11906 }%
11907 \renewcommand*{\acronymentry}[1]{%
11908 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}%
11909 \renewcommand*{\acronymsort}[2]{##1}%
11910 \renewcommand*{\acronymfont}[1]{##1}%
11911 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11912 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11913 \renewacronymstyle{dua-desc}%
11914 {%
11915 \GlsUseAcrEntryDispStyle{dua}%
11916 }%
11917 {%
11918 \GlsUseAcrStyleDefs{dua}%
11919 \renewcommand*{\GenericAcronymFields}{}%
11920 \renewcommand*{\acronymentry}[1]{%
11921 \glslongaccessdisplay{\acronymfont{\glentrylong{##1}}}{##1}}%
11922 \renewcommand*{\acronymsort}[2]{##2}%
11923 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11924 \renewacronymstyle{footnote}%
11925 {%
11926 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
11927 }%
11928 {%
11929 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11930 \glshyperfirstfalse
11931 \renewcommand*{\genacrfullformat}[2]{%
11932 \glsshortaccessdisplay
11933 {\protect\firstacronymfont{\glentryshort{##1}}}{##1}##2%
11934 \protect\footnote{\glslongaccessdisplay{\glentrylong{##1}}}{##1}}%
11935 }%
11936 \renewcommand*{\Genacrfullformat}[2]{%
11937 \glsshortaccessdisplay
11938 {\firstacronymfont{\Glsentryshort{##1}}}{##1}##2%
11939 \protect\footnote{\glslongaccessdisplay{\glentrylong{##1}}}{##1}}%
11940 }%
11941 \renewcommand*{\genplacrfullformat}[2]{%
11942 \glsshortpluralaccessdisplay
11943 {\protect\firstacronymfont{\glentryshortpl{##1}}}{##1}##2%
11944 \protect\footnote{\glslongpluralaccessdisplay{\glentrylongpl{##1}}}{##1}}%
11945 }%
11946 \renewcommand*{\Genplacrfullformat}[2]{%

```

```

11947 \glsshortpluralaccessdisplay
11948 {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2%
11949 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}}{##1}}%
11950 }%
11951 \renewcommand*{\acronymentry}[1]{%
11952 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11953 \renewcommand*{\acronymsort}[2]{##1}%
11954 \renewcommand*{\acronymfont}[1]{##1}%
11955 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11956 \renewcommand*{\acrfullfmt}[3]{%
11957 \glslink[##1]{##2}{%
11958 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}##3\space
11959 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11960 \renewcommand*{\Acrfullfmt}[3]{%
11961 \glslink[##1]{##2}{%
11962 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}}{##2}##3\space
11963 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11964 \renewcommand*{\ACRfullfmt}[3]{%
11965 \glslink[##1]{##2}{%
11966 \glsshortaccessdisplay
11967 {\mfirstucMakeUppercase
11968 {\acronymfont{\glsentryshort{##2}}}{##2}##3\space
11969 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11970 \renewcommand*{\acrfullplfmt}[3]{%
11971 \glslink[##1]{##2}{%
11972 \glsshortpluralaccessdisplay
11973 {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
11974 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11975 \renewcommand*{\ACRfullplfmt}[3]{%
11976 \glslink[##1]{##2}{%
11977 \glsshortpluralaccessdisplay
11978 {\acronymfont{\Glsentryshortpl{##2}}}{##2}##3\space
11979 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11980 \renewcommand*{\ACRfullplfmt}[3]{%
11981 \glslink[##1]{##2}{%
11982 \glsshortpluralaccessdisplay
11983 {\mfirstucMakeUppercase
11984 {\acronymfont{\glsentryshortpl{##2}}}{##2}##3\space
11985 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%

```

Similarly for \glsentryfull etc:

```

11986 \renewcommand*{\glsentryfull}[1]{%
11987 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}\space
11988 (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
11989 \renewcommand*{\Glsentryfull}[1]{%
11990 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
11991 (\glslongaccessdisplay{\glsentrylong{##1}}{##1})}%
11992 \renewcommand*{\glsentryfullpl}[1]{%

```

```

11993 \glsshortpluralaccessdisplay
11994 {\acronymfont{\glentryshortpl{##1}}{##1}\space
11995 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11996 \renewcommand*\Glsentryfullpl[1]{%
11997 \glsshortpluralaccessdisplay
11998 {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
11999 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
12000 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

12001 \renewacronymstyle{footnote-sc}%
12002 {%
12003 \GlsUseAcrEntryDisplayStyle{footnote}%
12004 }%
12005 {%
12006 \GlsUseAcrStyleDefs{footnote}%
12007 \renewcommand{\acronymentry}[1]{%
12008 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
12009 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
12010 \renewcommand*\acrpluralsuffix{\glstextup{\glspluralsuffix}}%
12011 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

12012 \renewacronymstyle{footnote-sm}%
12013 {%
12014 \GlsUseAcrEntryDisplayStyle{footnote}%
12015 }%
12016 {%
12017 \GlsUseAcrStyleDefs{footnote}%
12018 \renewcommand{\acronymentry}[1]{%
12019 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
12020 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
12021 \renewcommand*\acrpluralsuffix{\glspluralsuffix}%
12022 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

12023 \renewacronymstyle{footnote-desc}%
12024 {%
12025 \GlsUseAcrEntryDisplayStyle{footnote}%
12026 }%
12027 {%
12028 \GlsUseAcrStyleDefs{footnote}%
12029 \renewcommand*\GenericAcronymFields{}%
12030 \renewcommand*\acronymsort[2]{##2}%
12031 \renewcommand*\acronymentry[1]{%
12032 \glslongaccessdisplay{\glentrylong{##1}}{##1}\space
12033 (\glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}}%
12034 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```
12035 \renewacronymstyle{footnote-sc-desc}%
12036 {%
12037   \GlsUseAcrEntryDispStyle{footnote-sc}%
12038 }%
12039 {%
12040   \GlsUseAcrStyleDefs{footnote-sc}%
12041   \renewcommand*{\GenericAcronymFields}{}%
12042   \renewcommand*{\acronymsort}[2]{##2}%
12043   \renewcommand*{\acronymentry}[1]{%
12044     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
12045     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1)}}%
12046 }
```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```
12047 \renewacronymstyle{footnote-sm-desc}%
12048 {%
12049   \GlsUseAcrEntryDispStyle{footnote-sm}%
12050 }%
12051 {%
12052   \GlsUseAcrStyleDefs{footnote-sm}%
12053   \renewcommand*{\GenericAcronymFields}{}%
12054   \renewcommand*{\acronymsort}[2]{##2}%
12055   \renewcommand*{\acronymentry}[1]{%
12056     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
12057     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1)}}%
12058 }
```

aultshortaccess \glsdefaultshortaccess{<long>}{<short>}

Default shortaccess value.

```
12059 \newcommand*{\glsdefaultshortaccess}[2]{#1}
```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```
12060 \renewcommand*{\newacronymhook}{%
12061   \edef\@gls@keylist{%
12062     shortaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok}},%
12063     shortpluralaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok}},%
12064     \the\glskeylisttok}%
12065   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
12066 }
```

ltNewAcronymDef Modify default style to use access text:

```
12067 \renewcommand*{\DefaultNewAcronymDef}{%
```

```

12068 \edef\@do@newglossaryentry{%
12069   \noexpand\newglossaryentry{\the\glslabeltok}%
12070   {%
12071     type=\acronymtype,%
12072     name={\the\glsshorttok},%
12073     description={\the\glslongtok},%
12074     descriptionaccess=\relax,
12075     text={\the\glsshorttok},%
12076     access={\noexpand\@glo@textaccess},%
12077     sort={\the\glsshorttok},%
12078     short={\the\glsshorttok},%
12079     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12080     shortaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok}},%
12081     long={\the\glslongtok},%
12082     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12083     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12084     first={\noexpand\glslongaccessdisplay
12085       {\the\glslongtok}{\the\glslabeltok}\space
12086       (\noexpand\glsshortaccessdisplay
12087         {\the\glsshorttok}{\the\glslabeltok})},%
12088     plural={\the\glsshorttok\acrpluralsuffix},%
12089     firstplural={\noexpand\glslongpluralaccessdisplay
12090       {\noexpand\@glo@longpl}{\the\glslabeltok}\space
12091       (\noexpand\glsshortpluralaccessdisplay
12092         {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
12093     firstaccess=\relax,
12094     firstpluralaccess=\relax,
12095     textaccess={\noexpand\@glo@shortaccess},%
12096     \the\glskeylisttok
12097   }%
12098 }%
12099 \let\@org@gls@assign@firstpl\gls@assign@firstpl
12100 \let\@org@gls@assign@plural\gls@assign@plural
12101 \let\@org@gls@assign@descplural\gls@assign@descplural
12102 \def\gls@assign@firstpl##1##2{%
12103   \@gls@expand@field{##1}{firstpl}{##2}%
12104 }%
12105 \def\gls@assign@plural##1##2{%
12106   \@gls@expand@field{##1}{plural}{##2}%
12107 }%
12108 \def\gls@assign@descplural##1##2{%
12109   \@gls@expand@field{##1}{descplural}{##2}%
12110 }%
12111 \@do@newglossaryentry
12112 \let\gls@assign@firstpl\@org@gls@assign@firstpl
12113 \let\gls@assign@plural\@org@gls@assign@plural
12114 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
12115 }

```

teNewAcronymDef

```
12116 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
12117   \edef\@do@newglossaryentry{%
12118     \noexpand\newglossaryentry{\the\glslabeltok}%
12119     {%
12120       type=\acronymtype,%
12121       name={\noexpand\acronymfont{\the\glsshorttok}},%
12122       sort={\the\glsshorttok},%
12123       text={\the\glsshorttok},%
12124       short={\the\glsshorttok},%
12125       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12126       shortaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glsshorttok}},%
12127       long={\the\glslongtok},%
12128       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12129       access={\noexpand\@glo@textaccess},%
12130       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12131       symbol={\the\glslongtok},%
12132       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12133       firstpluralaccess=\relax,
12134       textaccess={\noexpand\@glo@shortaccess},%
12135       \the\glskeylisttok
12136     }%
12137   }%
12138   \let\@org@gls@assign@firstpl\gls@assign@firstpl
12139   \let\@org@gls@assign@plural\gls@assign@plural
12140   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
12141   \def\gls@assign@firstpl##1##2{%
12142     \@@gls@expand@field{##1}{firstpl}{##2}%
12143   }%
12144   \def\gls@assign@plural##1##2{%
12145     \@@gls@expand@field{##1}{plural}{##2}%
12146   }%
12147   \def\gls@assign@symbolplural##1##2{%
12148     \@@gls@expand@field{##1}{symbolplural}{##2}%
12149   }%
12150   \@do@newglossaryentry
12151   \let\gls@assign@plural\@org@gls@assign@plural
12152   \let\gls@assign@firstpl\@org@gls@assign@firstpl
12153   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
12154 }
```

onNewAcronymDef

```
12155 \renewcommand*{\DescriptionNewAcronymDef}{%
12156   \edef\@do@newglossaryentry{%
12157     \noexpand\newglossaryentry{\the\glslabeltok}%
12158     {%
12159       type=\acronymtype,%
12160       name={\noexpand
12161         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
```

```

12162     access={\noexpand\@glo@textaccess},%
12163     sort={\the\glssshorttok},%
12164     short={\the\glssshorttok},%
12165     shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
12166     shortaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glssshorttok}},%
12167     long={\the\glslongtok},%
12168     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12169     first={\the\glslongtok},%
12170     firstaccess=\relax,
12171     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12172     text={\the\glssshorttok},%
12173     textaccess={\the\glslongtok},%
12174     plural={\the\glssshorttok\noexpand\acrpluralsuffix},%
12175     symbol={\noexpand\@glo@text},%
12176     symbolaccess={\noexpand\@glo@textaccess},%
12177     symbolplural={\noexpand\@glo@plural},%
12178     firstpluralaccess=\relax,
12179     textaccess={\noexpand\@glo@shortaccess},%
12180     \the\glskeylisttok}%
12181 }%
12182 \let\@org@gls@assign@firstpl\gls@assign@firstpl
12183 \let\@org@gls@assign@plural\gls@assign@plural
12184 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
12185 \def\gls@assign@firstpl##1##2{%
12186   \@gls@expand@field{##1}{firstpl}{##2}%
12187 }%
12188 \def\gls@assign@plural##1##2{%
12189   \@gls@expand@field{##1}{plural}{##2}%
12190 }%
12191 \def\gls@assign@symbolplural##1##2{%
12192   \@gls@expand@field{##1}{symbolplural}{##2}%
12193 }%
12194 \do@newglossaryentry
12195 \let\gls@assign@firstpl\@org@gls@assign@firstpl
12196 \let\gls@assign@plural\@org@gls@assign@plural
12197 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
12198 }

```

teNewAcronymDef

```

12199 \renewcommand*{\FootnoteNewAcronymDef}{%
12200   \edef\@do@newglossaryentry{%
12201     \noexpand\newglossaryentry{\the\glslabeltok}%
12202     {%
12203       type=\acronymtype,%
12204       name={\noexpand\acronymfont{\the\glssshorttok}},%
12205       sort={\the\glssshorttok},%
12206       text={\the\glssshorttok},%
12207       textaccess={\the\glslongtok},%
12208       access={\noexpand\@glo@textaccess},%

```

```

12209 plural={\the\glssshorttok\noexpand\acrpluralsuffix},%
12210 short={\the\glssshorttok},%
12211 shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
12212 long={\the\glslongtok},%
12213 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12214 description={\the\glslongtok},%
12215 descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12216 \the\glskeylisttok
12217 }%
12218 }%
12219 \let\@org@gls@assign@plural\gls@assign@plural
12220 \let\@org@gls@assign@firstpl\gls@assign@firstpl
12221 \let\@org@gls@assign@descplural\gls@assign@descplural
12222 \def\gls@assign@firstpl##1##2{%
12223   \@gls@expand@field{##1}{firstpl}{##2}%
12224 }%
12225 \def\gls@assign@plural##1##2{%
12226   \@gls@expand@field{##1}{plural}{##2}%
12227 }%
12228 \def\gls@assign@descplural##1##2{%
12229   \@gls@expand@field{##1}{descplural}{##2}%
12230 }%
12231 \do@newglossaryentry
12232 \let\gls@assign@plural\@org@gls@assign@plural
12233 \let\gls@assign@firstpl\@org@gls@assign@firstpl
12234 \let\gls@assign@descplural\@org@gls@assign@descplural
12235 }

```

11NewAcronymDef

```

12236 \renewcommand*{\SmallNewAcronymDef}{%
12237   \edef\@do@newglossaryentry{%
12238     \noexpand\newglossaryentry{\the\glslabeltok}%
12239     {%
12240       type=\acronymtype,%
12241       name={\noexpand\acronymfont{\the\glssshorttok}},%
12242       access={\noexpand\@glo@symbolaccess},%
12243       sort={\the\glssshorttok},%
12244       short={\the\glssshorttok},%
12245       shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
12246       shortaccess={\glsdefaultshortaccess{\the\glslongtok}{\the\glssshorttok}},%
12247       long={\the\glslongtok},%
12248       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12249       text={\noexpand\@glo@short},%
12250       textaccess={\noexpand\@glo@shortaccess},%
12251       plural={\noexpand\@glo@shortpl},%
12252       first={\the\glslongtok},%
12253       firstaccess=\relax,%
12254       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
12255       description={\noexpand\@glo@first},%

```



```

12256      descriptionplural={\noexpand\@glo@firstplural},%
12257      symbol={\the\glsshorttok},%
12258      symbolaccess={\the\glslongtok},%
12259      symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
12260      \the\glskeylisttok
12261    }%
12262  }%
12263  \let\@org@gls@assign@firstpl\gls@assign@firstpl
12264  \let\@org@gls@assign@plural\gls@assign@plural
12265  \let\@org@gls@assign@descplural\gls@assign@descplural
12266  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
12267  \def\gls@assign@firstpl##1##2{%
12268    \@@gls@expand@field{##1}{firstpl}{##2}%
12269  }%
12270  \def\gls@assign@plural##1##2{%
12271    \@@gls@expand@field{##1}{plural}{##2}%
12272  }%
12273  \def\gls@assign@descplural##1##2{%
12274    \@@gls@expand@field{##1}{descplural}{##2}%
12275  }%
12276  \def\gls@assign@symbolplural##1##2{%
12277    \@@gls@expand@field{##1}{symbolplural}{##2}%
12278  }%
12279  \do@newglossaryentry
12280  \let\gls@assign@firstpl\@org@gls@assign@firstpl
12281  \let\gls@assign@plural\@org@gls@assign@plural
12282  \let\gls@assign@descplural\@org@gls@assign@descplural
12283  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
12284 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
12285 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
12286 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
12287 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
12288 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

5.5 Debugging Commands

owglonameaccess

```
12289 \newcommand*{\showglonameaccess}[1]{%
```

```

12290 \expandafter\show\csname glo@\glsdetoklabel{#1}@access\endcsname
12291 }

owglotextaccess

12292 \newcommand*{\showglotextaccess}[1]{%
12293 \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
12294 }

glopluralaccess

12295 \newcommand*{\showglopluralaccess}[1]{%
12296 \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
12297 }

wglofirstaccess

12298 \newcommand*{\showglofirstaccess}[1]{%
12299 \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
12300 }

rstpluralaccess

12301 \newcommand*{\showglofirstpluralaccess}[1]{%
12302 \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
12303 }

glosymbolaccess

12304 \newcommand*{\showglosymbolaccess}[1]{%
12305 \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
12306 }

bolpluralaccess

12307 \newcommand*{\showglosymbolpluralaccess}[1]{%
12308 \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
12309 }

owglodescaccess

12310 \newcommand*{\showglodescaccess}[1]{%
12311 \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
12312 }

escpluralaccess

12313 \newcommand*{\showglodescpluralaccess}[1]{%
12314 \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
12315 }

wgloshortaccess

12316 \newcommand*{\showgloshortaccess}[1]{%
12317 \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
12318 }

```

ortpluralaccess

```
12319 \newcommand*{\showgloshortpluralaccess}[1]{%  
12320   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
12321 }
```

owglolongaccess

```
12322 \newcommand*{\showglolongaccess}[1]{%  
12323   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
12324 }
```

ongpluralaccess

```
12325 \newcommand*{\showglolongpluralaccess}[1]{%  
12326   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
12327 }
```

6 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex. Language support has now been split off into independent language modules.

```
12328 \NeedsTeXFormat{LaTeX2e}
12329 \ProvidesPackage{glossaries-babel}[2020/02/13 v4.45 (NLCT)]
```

Load tracklang to obtain language settings.

```
12330 \RequirePackage{tracklang}
12331 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
12332 \AnyTrackedLanguages
12333 {%
12334   \ForEachTrackedDialect{\this@dialect}{%
12335     \IfTrackedLanguageFileExists{\this@dialect}%
12336       {glossaries-}% prefix
12337       {.ldf}%
12338       {%
12339         \RequireGlossariesLang{\CurrentTrackedTag}%
12340       }%
12341       {%
12342         \PackageWarningNoLine{glossaries}%
12343           {No language module detected for ‘\this@dialect’.\MessageBreak
12344             Language modules need to be installed separately.\MessageBreak
12345             Please check on CTAN for a bundle called\MessageBreak
12346             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
12347       }%
12348     }%
12349   }%
12350 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
12351 \NeedsTeXFormat{LaTeX2e}
12352 \ProvidesPackage{glossaries-polyglossia}[2020/02/13 v4.45 (NLCT)]
```

Load tracklang to obtain language settings.

```
12353 \RequirePackage{tracklang}
12354 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
12355 \AnyTrackedLanguages
```

```

12356 {%
12357     \ForEachTrackedDialect{\this@dialect}{%
12358         \IfTrackedLanguageFileExists{\this@dialect}%
12359         {glossaries-}% prefix
12360         {.ldf}%
12361         {%
12362             \RequireGlossariesLang{\CurrentTrackedTag}%
12363         }%
12364         {%
12365             \PackageWarningNoLine{glossaries}%
12366             {No language module detected for ‘\this@dialect’.\MessageBreak
12367             Language modules need to be installed separately.\MessageBreak
12368             Please check on CTAN for a bundle called\MessageBreak
12369             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
12370         }%
12371     }%
12372 }%
12373 {}%

```

Glossary

`makeindex` An indexing application [10](#), [14](#), [30](#), [33](#), [186](#)

`xindy` An flexible indexing application with multilingual support written in Perl [10](#), [14](#), [30](#), [33](#), [186](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 8
General: Added range facility in format key 119	
\writeist: Added spaces after \delimN and \delimR in ist file 166	
1.04 (2007-08-03)	1.12 (2008-03-08)
General: Added \glstextformat 103	\@GLSpl: now uses
1.05 (2007-08-10)	\glentrydescplural and
\glossarysection: added \@mkboth to \glossarysection 46	\glentrysymbolplural instead of
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 87	\glentrydesc and
1.07 (2007-09-13)	\glentrysymbol 133
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 117	\@Glspl@: now uses
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 163	\glentrydescplural and
1.08 (2007-10-13)	\glentrysymbolplural instead of
General: Added babel support 40	\glentrydesc and
listgroup: changed listgroup style to use \glsgetgrouptitle 280	\glentrysymbol 131
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 281	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 127
1.1 (2008-02-22)	descriptionplural: new 69
\@glossarysection: numbered sections and auto label added 47	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 87
\@gls@tmpb: changed \toksdef to \newtoks 121	descriptionplural support added 87
\@gls@toc: numberline added 49	symbolplural support added 87
\@p@glossarysection: numbered sections and auto label added 48	\Glsentrydescplural: New 156
General: amsgen now loaded (\new@ifnextchar needed) 4	\glentrydescplural: New 156
translate: translate option added 28	\Glsentrysymbolplural: New 157
\setglossarysection: new 47	\glentrysymbolplural: New 157
numberedsection: numberedsection package option added 9	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 247
	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 253
	symbolplural: new 70

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	134–142
\ACRfullpl: new	228
\Acrfullpl: new	227
\acrfullpl: new	227
\acrpluralsuffix: New	225
\gls@defglossaryentry: Changed default first value	87
Changed default firstplural value	87
Removed restriction on only using \newglossaryentry in the preamble	92
\newacronym: Removed restriction on only using \newacronym in the preamble	225
1.14 (2008-06-17)	
\@gls@hypergroup: new	275
General: added nonumberlist key to \printglossary	212
added numberedsection key to \printglossary	211
\firstacronymfont: new	228
\glsautoprefix: new	8
\glsnavhyperlink: changed \edef to \protected@edef	274
\glsnavhypertarget: added write to aux file	274
\glsnavigation: changed to only use labels for groups that are present ..	276
1.15 (2008-08-15)	
\@gls@link: added \glslabel	117
\gls@defglossaryentry: check for \@glo@first in description	91
check for \@glo@text in symbol	91
\gls@hypergroup: new	275
\glsnavhypertarget: added check if rerun required	274
\glssettoctitle: new	39
\printglossary: changed the way the TOC title is set	196
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	130
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	133
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	130
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	132
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	129
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	134
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	131
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page	127
\gls@defglossaryentry: Changed def to let	87
1.17 (2008-12-26)	
\@do@esc@wrglossary: new	190
\@do@seeglossary: new	194
\@glo@storeentry: new	93
\@gls@glossary: changed definition to use \index instead of \@index	186
\@glsdefaultplural: new	74
\@glsdefaultsort: new	74
\@glshypernumber: new	222
\@glsnoname: new	74
\@glsnonextpages: new	212
General: added xindy support	30
parent: new	71
see: new	71
\gls@defglossaryentry: added nonumberlist key	87
added parent key	87
added see key	87
Stored main part of entry format when entry is defined	92
\gls@suffixF: new	44
\gls@suffixFF: new	44
\gls@wrglossary: modified to allow for xindy support	187

\glshyperlink: new	163	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	247
\glshypernumber: modified to allow material to be attached to location	222	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	253
\glshnavhyperlink: replaced \hyperlink to \@glslink	274	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	256
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	274	2.01 (2009 May 30) \@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	118
\glsssee: new	195	\forall glossaries: replaced \ifthenelse with \ifx	58
\glssseeformat: new	195	\forall glossentries: replaced \ifthenelse with \ifx	58
\glssSetSuffixF: new	44	\glsgdefmain: new	17
\glssSetSuffixFF: new	44	\glsgdescwidth: changed \linewidth to \hsize	282, 304
\ifglsxindy: new	30	\glsglistdottedwidth: changed \linewidth to \hsize	282
\istfilename: added xindy support ...	43	\glsgpagelistwidth: changed \linewidth to \hsize	282, 304
\newglossarystyle: made \newglossarystyle long	221	nomain: added nomain package option	17
\nopostdesc: new	42	\writeist: removed item_02 - no such makeindex key	171
nonumberlist: new	72	2.02 (2007-07-13) \@printglossary: suppressed warning globally rather than locally	199
\printglossary: added check to determine if \printglossary is already defined	196	2.02 (2009-07-13) \glossarysection: changed \@mkboth to \glossarymark	46
added print language to aux file	196	\glsglossarymark: New	46
order: order package option added	30	2.03 (2009-09-23) \@GLS@: Added check for hyperfirst	130
\writeist: added xindy support	166	\@GLSp1: Added check for hyperfirst	133
1.18 (2009-01-14) \@glsl@loadlist: new	11	\@GLS@: Added check for hyperfirst	130
\@glsl@loadlong: new	10	\@GLSp1@: Added check for hyperfirst	132
\@glsl@loadsuper: new	11	\@glsl@: Added check for hyperfirst	129
\@glsl@loadtree: new	11	\@glsl@@link: new	116
\glsgdefglossaryentry: Changed default value of sort to \@glsgdefaultsort	87	\@glsl@link: added \leavevmode	117
moved sort sanitization to \newglossaryentry	91	Moved entry existence check to avoid duplicate code	117
\glstarget: new	215	\@glsgdisp: Added check for hyperfirst	134
\oldacronym: new	224	\@glsgspl@: Added check for hyperfirst	131
nolist: new	11	\glsglossarymark: Added check to see if it's already defined	46
nolong: new	10	hyperfirst: new	29
sort: moved sanitization to \newglossaryentry	69		
nostyles: new	11		
nosuper: new	11		
notree: new	11		
1.19 (2009-03-02) \glsclearpage: new	48		
\glsgdisp: new	133		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	251		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	130
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms	133
\@GLs@: Changed test to check if glossary type has been identified as a list of acronyms	130
\@GLspl@: Changed test to check if glossary type has been identified as a list of acronyms	132
\@glossaryentryfield: new	93
\@glossarysubentryfield: new	93
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	129
\@glsacronymlists: new	18
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	134
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	131
\@newglossaryentryposthook: new	92
\@newglossaryentryprehook: new	92
acronymlists: new	20
\DeclareAcronymList: new	19
\DefineAcronymSynonyms: new	241
\gls@defglossaryentry: added user1-6 keys	87
\glsadd: fixed bug that ignored counter	163
\Glsentryuseri: new	159
\Glsentryuseri: new	159
\Glsentryuserii: new	159
\Glsentryuserii: new	159
\Glsentryuseriii: new	159
\Glsentryuseriii: new	159
\Glsentryuseriv: new	160
\Glsentryuseriv: new	159
\Glsentryuserv: new	160
\Glsentryuserv: new	160
\Glsentryuservi: new	160
\Glsentryuservi: new	160
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	66
\SetAcronymLists: new	20
\SetDefaultAcronymDisplayStyle: new	243
\SetDefaultAcronymStyle: new	244
\SetDescriptionAcronymDisplayStyle: new	249
\SetDescriptionDUAAcronymDisplayStyle: new	247
\SetDescriptionFootnoteAcronymDisplayStyle: new	245
\SetDUADisplayStyle: new	256
\SetFootnoteAcronymDisplayStyle: new	251
\SetSmallAcronymDisplayStyle: new	254
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	133
Removed spurious brace. Patch provided by Sergiu Dotenco	134
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	171
2.06 (2010-06-14)	
\altnewglossary: new	67
\CustomAcronymFields: new	259
\CustomNewAcronymDef: new	259
\SetCustomDisplayStyle: new	259
\SetCustomStyle: new	259
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	163
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites	176
\gls@wrglossary: modified to take into account savewrites	187
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument	119
3.0 (2011-04-02)	
\@do@esc@wrglossary: added check for hyper location prefix	192
modified to use new format	190
\@@glossarysec: replaced \@ifundefined with \ifcsundef	8
\@do@seeglossary: Sanitize and escape cross-referencing information	194
\@gls@counterwithin: new	13

\@gls@ifinlist: new	50	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter	163
\@gls@saveentrycounter	118	\GlsAddXdyCounters: new	50
added \@gls@setsort	118	\glentrycounterlabel: new	214
\@gls@saveentrycounter: new	118	\glentryitem: new	214
\@gls@setupsort@def: new	15	\Glentrylong: new	161
\@gls@setupsort@standard: new	14	\glentrylong: new	160
\@gls@setupsort@use: new	15	\Glentrylongpl: new	161
\@gls@xdy@locationlist: new	53	\glentrylongpl: new	161
\@glslink: replaced \@ifundefined		\Glentryshort: new	160
with \ifcsundef	127	\glentryshort: new	160
\@glsnextpages: new	212	\Glentryshortpl: new	160
\@print@glossary: replaced		\glentryshortpl: new	160
\@ifundefined with \ifcsundef ..	199	\glsgetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	218
\currentglossary	198	\gls glossarymark: replaced	
added \glsnextpages	198	\@ifundefined with \ifcsundef ..	46
make toctitle default to title	198	\glshyperlink: changed default from	
\@xdyattributelist: new	49	\glentryname to \glentrytext ..	163
General: added prefix to hyperlink	223	\glshypernumber: replaced	
etoolbox now loaded	4	\@ifundefined with \ifcsundef ..	222
replaced \@ifundefined with		\glsnumberformat: replaced	
\ifcsundef	38, 41, 114, 210	\@ifundefined with \ifcsundef ..	45
\acrfootnote: new	245	\glsrefentry: new	214
\ACRfull: added starred version	227	\glsresetsubentrycounter: new ...	213
\Acrfull: added starred version	226	\glsseeitem: hyperlink uses	
\acrfull: added starred version	225	\glsseeitemformat instead of	
\ACRfullpl: added starred version ...	228	\glentryname	196
\Acrfullpl: added starred version ...	227	\glsseeitemformat: new	196
\acrfullpl: added starred version ...	227	\glssortnumberfmt: new	14
\acrlinkfootnote: new	245	\glsstepentry: new	213
\acrno linkfootnote: new	245	\glsstepsubentry: new	213
savewrites: new	33	\glssubentrycounterlabel: new ...	214
see: added \@glo@seeautonumberlist ..	71	\glssubentryitem: new	214
seeautonumberlist: new	10	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef	214
\@ifundefined with \ifcsundef ..	46	short: new	73
\glossarystyle: replaced		shortplural: new	73
\@ifundefined with \ifcsundef ..	220	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	59
\@ifundefined with \ifcsundef ..	31	\ifglentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	59
\@gls@defsort	91	\istfile: deprecated	185
added short and long keys	88	glossaryentry: new	12
replaced \@ifundefined with		glossarysubentry: new	13
\ifcsundef	88	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	48	\newrobustcmd	76

<code>\newglossarystyle</code> : replaced		<code>\showglouseriii</code> : new	262
<code>\@ifundefined</code> with <code>\ifcsundef</code>	221	<code>\showglouseriv</code> : new	262
<code>\ns@newglossary</code> : added		<code>\showglouserv</code> : new	263
<code>\@gls@defsortcount</code>	67	<code>\showglouservi</code> : new	263
replaced <code>\@ifundefined</code> with		<code>subentrycounter</code> : new	13
<code>\ifcsundef</code>	66	<code>\writeist</code> : added xindy-only macro	
<code>entrycounter</code> : new	12	definitions to glossary open tag	168
<code>\oldacronym</code> : replaced <code>\@ifundefined</code>		modified to support new format	166
with <code>\ifcsundef</code>	224	3.01 (2011-04-12)	
<code>compatible-2.07</code> : <code>compatible-2.07</code>		<code>\@glswritefiles</code> : added check for	
option added	34	empty glossaries	185
<code>long</code> : new	73	General: made robust	130
<code>longplural</code> : new	73	<code>\ACRfull</code> : made robust	226
<code>nonumberlist</code> : now boolean	72	<code>\Acrfull</code> : made robust	226
<code>sort</code> : new	13	<code>\acrfull</code> : made robust	225
<code>counter</code> : replaced <code>\@ifundefined</code> with		<code>\acrfullformat</code> : removed	
<code>\ifcsundef</code>	70	<code>\acronymfont</code> as it should already be	
<code>counterwithin</code> : new	12	set in the second argument.	226
<code>\printglossary</code> : replaced		<code>\ACRfullpl</code> : made robust	228
<code>\@ifundefined</code> with <code>\ifcsundef</code>	196	<code>\Acrfullpl</code> : made robust	227
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>		<code>\acrfullpl</code> : made robust	227
expanded options link options	245	<code>\ACRlong</code> : made robust	151
<code>\setentrycounter</code> : added optional		<code>\Acrlong</code> : made robust	151
argument	219	<code>\acrlong</code> : made robust	150
<code>\showacronymlists</code> : new	265	<code>\ACRlongpl</code> : made robust	153
<code>\showglocounter</code> : new	262	<code>\Acrlongpl</code> : made robust	153
<code>\showgloDESC</code> : new	263	<code>\acrlongpl</code> : made robust	152
<code>\showgloDESCplural</code> : new	263	<code>\ACRshort</code> : made robust	148
<code>\showglofirst</code> : new	261	<code>\Acrshort</code> : made robust	147
<code>\showglofirstpl</code> : new	261	<code>\acrshort</code> : made robust	146
<code>\showgloflag</code> : new	264	<code>\ACRshortpl</code> : made robust	149
<code>\showgloindex</code> : new	264	<code>\Acrshortpl</code> : made robust	149
<code>\showglolevel</code> : new	261	<code>\acrshortpl</code> : made robust	148
<code>\showglongame</code> : new	263	<code>\Gls</code> : made robust	129
<code>\showgloparent</code> : new	261	<code>\glsadd</code> : made robust	163
<code>\showgloplural</code> : new	261	<code>\glsaddall</code> : made robust	164
<code>\showglosort</code> : new	263	<code>\GLSdesc</code> : made robust	139
<code>\showglossaries</code> : new	265	<code>\Glsdesc</code> : made robust	139
<code>\showglossarycounter</code> : new	266	<code>\glsdesc</code> : made robust	138
<code>\showglossaryentries</code> : new	266	<code>\GLSdescplural</code> : made robust	140
<code>\showglossaryin</code> : new	265	<code>\Glsdescplural</code> : made robust	139
<code>\showglossaryout</code> : new	265	<code>\glsdescplural</code> : made robust	139
<code>\showglossarytitle</code> : new	265	<code>\glsfirst</code> : made robust	135
<code>\showglosymbol</code> : new	264	<code>\GLSfirstplural</code> : made robust	137
<code>\showglosymbolplural</code> : new	264	<code>\Glsfirstplural</code> : made robust	137
<code>\showglotext</code> : new	261	<code>\glsfirstplural</code> : made robust	137
<code>\showglotype</code> : new	262	<code>\glslink</code> : made robust	116
<code>\showglouserI</code> : new	262	<code>\GLSname</code> : made robust	138
<code>\showglouserII</code> : new	262	<code>\Glsname</code> : made robust	138

\glsname: made robust	138	\@printglossary: add a way to fetch	
\GLSpl: made robust	132	current entry label	199
\Glspl: made robust	132	savenumberlist: new	10
\glspl: made robust	131	ucmark: new	12
\GLSplural: made robust	136	\gls@defglossaryentry: added	
\GLSsymbol: made robust	141	numberlist element	91
\Glsymbol: made robust	140	\gls@save@numberlist: new	196
\glssymbol: made robust	140	\gls@wrglossary: added check for	
\GLSsymbolplural: made robust	141	glossary file defined	187
\Glsymbolplural: made robust	141	\glsdisplaynumberlist: new	161
\glssymbolplural: made robust	141	\glsentrycounter: set default value ..	118
\Glstext: made robust	135	\Glsentryfull: fixed bug (replaced	
\glstext: made robust	134	\glsentryshortpl with	
\GLSuseri: made robust	142	\glsentryshort)	161
\Glsuseri: made robust	142	\glsentryfullpl: fixed bug (replaced	
\glsuseri: made robust	142	\glsentryshort with	
\GLSuserii: made robust	143	\glsentryshortpl)	161
\Glsuserii: made robust	143	\glsentrynumberlist: new	161
\glsuserii: made robust	142	\glsmoveentry: new	93
\GLSuseriii: made robust	144	\glsresetsubentrycounter: new ...	213
\Glsuseriii: made robust	144	\ifglshaschildren: new	61
\glsuseriii: made robust	143	\ifglshasparent: new	62
\GLSuseriv: made robust	145	\makeglossaries: added list parser ..	180
\Glsuseriv: made robust	144	indexonlyfirst: new	29
\glsuseriv: made robust	144	\renewglossarystyle: new	221
\GLSuserv: made robust	145	\showglossaryentries: fixed misspelt	
\Glsuserv: made robust	145	command	266
\glsuserv: made robust	145	\SmallNewAcronymDef: fixed broken	
\GLSuservi: made robust	146	short and long plural	254
\Glsuservi: made robust	146	3.03 (2012/09/21)	
\glsuservi: made robust	146	\@gls@sanitizesort: new	23
3.02 (2012-05-19)		\@gls@setupsort@standard: used	
\glsnumlistlastsep: new	162	\@gls@sanitizesort	14
\glsnumlistsep: new	162	\@printglossary: allow title to override	
3.02 (2012-05-21)		default toctitle	197
\@do@wrglossary: changed		General: allow title to set toctitle	210
\@glslocref to		\glsinlinedescformat: new	278
\theglsentrycounter	193	\glsinlineemptydescformat: new ..	278
\@do@wrglossary: changed		\glsinlinenameformat: new	278
\@do@wr@glossary to test for		\glsinlinepostchild: new	278
indexonlyfirst option; put old		\glsinlinesubdescformat: new	278
\@do@wr@glossary code into		\glsinlinesubnameformat: new	278
\@do@wrglossary	188	\glspostinline: replaced "." with	
\@gls@missingnumberlist: new	74	\glspostdescription	278
\@glswritefiles: added check for		list: added check for glsnogroupskip ..	280
existence of token in case		altlongragged4col: added check for	
\makeglossaries has been		glsnogroupskip	297
omitted	185	altsuperragged4col: added check for	
		glsnogroupskip	316

alttree: added check for		\gls@disablepagerefexpansion: new	188
glsnogroupskip	325	\gls@numberpage: new	188
index: added check for glsnogroupskip	319	\gls@protected@pagefmts: new	188
nogroupskip: new	12	\gls@romanpage: new	188
long: added check for glsnogroupskip	283	\glsdefmain: added check for doc	
long3col: added check for		package	17
glsnogroupskip	285	\glsorg@endtheglossary: new	5
long4col: added check for		\glsorg@theglossary: new	5
glsnogroupskip	286	\PrintChanges: new	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip	294	\@do@esc@wrglossary: add Roman	
longragged3col: added check for		case. Fixed bugs in the else	
glsnogroupskip	295	statements	191
nopostdot: new	12	\@gls@link: added check for	
tree: added check for glsnogroupskip	320	“nohypertypes”	117
treenoname: added check for		mcolalttree: replaced ‘2’ with	
glsnogroupskip	322	\glsmcols	303
super: added check for glsnogroupskip	305	mcolindex: replaced ‘2’ with \glsmcols	299
super3col: added check for		mcolindexspannav: replaced ‘2’ with	
glsnogroupskip	307	\glsmcols	300
super4col: added check for		mcoltree: replaced ‘2’ with \glsmcols	300
glsnogroupskip	309	mcoltreenoname: replaced ‘2’ with	
superragged: added check for		\glsmcols	302
glsnogroupskip	312	mcoltreespannav: replaced ‘2’ with	
superragged3col: added check for		\glsmcols	301
glsnogroupskip	314	\gls@protected@pagefmts: added	
3.04 (2012-11-11)		Roman to list	188
altlist: replaced \newline with		\gls@Romanpage: new	189
paragraph break	280	\glsgetgrouplabel: fixed bug (typo in	
3.04 (2012-11-18)		\equal)	219
\@do@wrglossary: changed		\nopostdesc: made robust	42
\theglscopycounter back to		3.05 (2013/04/21)	
\@glslocref	193	\@gls@nohyperlist: new	20
\@do@esc@wrglossary: modified to		\GlsDeclareNoHyperList: new	20
compensate for possible incorrect		nohypertypes: new	20
page number	191	3.06 (2013/06/17)	
\@gls@escbsdq: unsanitize		\@xdy@main@language: Changed back to	
\gls@numberpage, \gls@alphpage,		using \language	30
\gls@Alphpage and		\findrootlanguage: Obsoleted	56
\gls@romanpage	120	3.07 (2013-07-05)	
\@print@glossary: Moved aux write to		\@gls@link: fixed bug that failed to find	
end of document to prevent		entry in list	117
unwanted whatsit occurring here. . .	199	\glossarypreamble: modified to work	
General: Added check for doc package	4	with \setglossarypreamble	45
added datatool-base as a required		\gls@docclearpage: added check for	
package	4	openright	48
added local key	114	\glspostdescription: Added	
\gls@Alphpage: new	188	spacefactor code	11
\gls@alphpage: new	188		

\GlsSetXdyCodePage: Added check for fontspec	57	\glssseelist: made robust	195
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	249	\ifglshasdesc: new	62
\setglossarypreamble: new	45	\ifglshassymbol: new	62
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	297
list: updated list style to use \glossentry and \subglossentry	279	alttree: updated to use \glossentry and \subglossentry	324
listdotted: updated listdotted style to use \glossentry and \subglossentry	281	index: added paragraph break at end of environment	318
altlist: updated altlist style to use \glossentry and \subglossentry	280	updated to use \glossentry and \subglossentry	318
inline: updated inline style to use \glossentry and \subglossentry	277	long: updated to use \glossentry and \subglossentry	283
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	294
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	94	longragged3col: updated to use \glossentry and \subglossentry	295
updated for \glossentry	94	tree: updated to use \glossentry and \subglossentry	320
\@glossaryentryfield: switched to \glossentry	93	\setglossarystyle: new	220
\@glossarysubentryfield: switched to \subglossentry	93	\setglossentrycompatibility: new	217
General: added nogroupskip key to \printglossary	211	superragged: updated to use \glossentry and \subglossentry	312
removed definition of \@glossaryentryfield	378	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield	378	\@gls@assign@symbolplural@field: new	23
\compatibleglossentry: new	215	\@gls@default@value: new	70
\compatiblesubglossentry: new ...	216	\Glsentrydesc: made robust	156
\glossaryentryfield: deprecated ...	217	\Glsentrydescplural: made robust ..	156
\Glossentrydesc: new	216	\Glsentryfirst: made robust	158
\glossentrydesc: new	216	\Glsentryfirstplural: made robust .	158
\Glossentryname: new	216	\Glsentryfull: made robust	161
\glossentryname: new	215	\Glsentryfullpl: made robust	161
\Glossentrysymbol: new	216	\Glsentrylong: made robust	161
\glossentrysymbol: new	216	\Glsentrylongpl: made robust	161
\gls@assign@desc@field: new	22	\Glsentryname: made robust	155
\gls@assign@descplural@field: new	22	\Glsentryplural: made robust	157
\gls@assign@field: new	76	\Glsentryshort: made robust	160
\gls@ifnotmeasuring: new	95	\Glsentryshortpl: made robust	160
\glsaddallunused: new	164	\Glsentrysymbol: made robust	157
\glsexpandfields: new	76	\Glsentrysymbolplural: made robust	157
\glsnoexpandfields: new	76	\Glsentrytext: made robust	157
\glsssee: made robust	195	\Glsentryuseri: made robust	159
\glssseeformat: made robust	195	\Glsentryuserii: made robust	159
\glssseeitem: made robust	196	\Glsentryuseriii: made robust	159
		\Glsentryuseriv: made robust	160
		\Glsentryuserv: made robust	160
		\Glsentryuservi: made robust	160

\glstextup: new	225	\@Gls@: add \glsifplural,	
\ifglshassymbol: changed test to check		\glscapscase, \glscustomtext and	
for \@gls@default@symbol	62	\glsinsert	130
3.10a (2013-09-28)		change to using \glentryfmt style	
\gls@assign@type@field: new	22	commands	130
3.10a (2013-10-13)		removed \makefirstuc (now dealt	
\@gls@keymap: new	78	with in \glentryfmt)	130
\@gls@provide@newglossary: new ...	65	\@Glspl@: add \glsifplural,	
\@gls@writedef: new	78	\glscapscase, \glscustomtext and	
\@glsdefaultplural: Obsolete	74	\glsinsert	132
\@glsnodesc: new	74	change to using \glentryfmt style	
\@print@glossary: Added		commands	132
providecommand code to aux file ..	200	removed \makefirstuc (now dealt	
\gls@defglossaryentry: Changed to		with in \glentryfmt)	132
using \@gls@default@value	87	\@acrlong: added \glslabel,	
new	86	\glsifplural, \glscapscase,	
\gls.writedefhook: new	86	\glsinsert and \glscustomtext	371
\makeglossaries: Added		\@acrshort: added \glslabel,	
providecommand code to aux file ..	179	\glsifplural, \glscapscase,	
\new@glossaryentry: new	77	\glsinsert and \glscustomtext	370
\ns@newglossary: added		\@gls@: add \glslabel, \glsifplural,	
\@gls@provide@newglossary	66	\glscapscase, \glscustomtext and	
3.11a (2013-10-15)		\glsinsert	129
\@ACRlong: added \glslabel,		change to using \glentryfmt style	
\glsifplural, \glscapscase,		commands	129
\glsinsert and \glscustomtext	372	\@gls@noexpand@fields: Fixed bug	
\@ACRshort: added \glslabel,		expand replaced with noexpand	75
\glsifplural, \glscapscase,		\@glsdisp: add \glslabel,	
\glsinsert and \glscustomtext	371	\glsifplural, \glscapscase,	
\@Acrlong: added \glslabel,		\glscustomtext and \glsinsert	133
\glsifplural, \glscapscase,		change to using \glentryfmt style	
\glsinsert and \glscustomtext	372	commands	134
\@Acrshort: added \glslabel,		\@glspl@: add \glslabel,	
\glsifplural, \glscapscase,		\glsifplural, \glscapscase,	
\glsinsert and \glscustomtext	370	\glscustomtext and \glsinsert	131
\@GLS@: add \glslabel, \glsifplural,		change to using \glentryfmt style	
\glscapscase, \glscustomtext and		commands	131
\glsinsert	130	General: added \glslabel,	
change to using \glentryfmt style		\glsifplural, \glscapscase,	
commands	130	\glsinsert and	
removed \MakeUppercase (now		\glscustomtext	147–153
moved to \glentryfmt)	130	changed to just use	
\@GLSpl: add \glslabel,		\Glsentrydescplural	140
\glsifplural, \glscapscase,		changed to just use	
\glscustomtext and \glsinsert	133	\glentrydescplural	139, 140
change to using \glentryfmt style		changed to just use \Glsentrydesc .	139
commands	133	changed to just use \glentrydesc .	139
removed \MakeUppercase as now		changed to just use	
dealt with in \glentryfmt	133	\Glsentryfirstplural	137

changed to just use		
\glentryfirstplural	137	
changed to just use \Glentryfirst	136	
changed to just use		
\glentryfirst	135, 136	
changed to just use \Glentryname	138	
changed to just use \glentryname	138	
changed to just use \Glentryplural	136	
changed to just use		
\glentryplural	136, 137	
changed to just use		
\Glentrysymbolplural	141	
changed to just use		
\glentrysymbolplural	141, 142	
changed to just use \Glentrysymbol	140	
changed to just use		
\glentrysymbol	140, 141	
Changed to just use \Glentrytext	135	
changed to just use \glentrytext	134	
changed to just use		
\Glentryuseriii	144	
changed to just use		
\glentryuseriii	143, 144	
changed to just use \Glentryuserii	143	
changed to just use \glentryuserii	143	
changed to just use \Glentryuseriv	144	
changed to just use		
\glentryuseriv	144, 145	
changed to just use \Glentryuseri	142	
changed to just use \glentryuseri	142	
changed to just use \Glentryuservi	146	
changed to just use \glentryuservi	146	
changed to just use \Glentryuserv	145	
changed to just use		
\glentryuserv	145, 146	
Now requires textcase	4	
acronymlists: replaced		
\@addtoacronymlists with		
\DeclareAcronymList	20	
\defgldisplay: obsoleted	113	
\defgldisplayfirst: obsoleted	113	
\defglentryfmt: new	65	
\forglentries: replaced \ifx with		
\ifdefempty	58	
\gls@assign@desc: new	86	
\gls@defglossaryentry: Fixed default		
counter if none supplied	90	
\gls@doentryfmt: new	65	
\gldisplay: obsoleted	112	
\gldisplayfirst: obsoleted	112	
\glgenentryfmt: new	107	
\glsgetgrouptitle: Added check in		
case non-Latin alphabet in use	218	
\glsglossarymark: replaced		
\MakeUppercase with		
\mfirstucMakeUppercase	46	
\glsnavigation: switched to using		
\@gls@getgrouptitle	276	
\ifglshasdesc: replaced \ifdefempty		
with \ifcempty	62	
\ifglshaslong: new	62	
\ifglshasshort: new	63	
\ifglshassymbol: replaced		
\ifdefempty with \ifcempty	62	
\ifglused: replaced \ifthenelse with		
\ifbool	60	
\longnewglossaryentry: new	86	
\ns@newglossary: replaced		
\gldisplay and		
\gldisplayfirst with		
\glentryfmt	66	
compatible-3.07: cnew	33	
\SetCustomDisplayStyle: updated to		
use \defglentryfmt	259	
\SetDefaultAcronymDisplayStyle:		
changed to use \defglentryfmt	243	
\SetDescriptionAcronymDisplayStyle:		
updated to use \defglentryfmt	249	
\SetDescriptionDUAAcronymDisplayStyle:		
updated to use \defglentryfmt	247	
\SetDescriptionFootnoteAcronymDisplayStyle:		
updated to use \defglentryfmt	245	
\SetDUADisplayStyle: updated to use		
\defglentryfmt	256	
\SetFootnoteAcronymDisplayStyle:		
updated to use \defglentryfmt	251	
\SetSmallAcronymDisplayStyle:		
updated to use \defglentryfmt	254	
\setupglossaries: new	37	
\showglolong: new	264	
\showgloshort: new	264	
numbers: new	36	
symbols: new	35	
3.12a (2013-10-16)		
\gls@defglossaryentry: added		
\glslabel	87	
\glsaddkey: new	80	

3.13a (2013-11-05)

\@gls@assign@symbol@field: changed to use \glssetnoexpandfield	23
\@gls@assign@symbolplural@field: changed to use \glssetnoexpandfield	23
\@gls@link: removed \relax	118
\@gls@notranslatorhook: new	27
\@gls@setupsort@standard: moved \@gls@santizesort to \glsprestandardsort	14
ucmark: added check for memoir	12
see: added \gls@checkseeallowed	71
\glossarysection: changed \glossarymark to \gls glossarymark	46
\glossarystyle: fixed bug caused by using \ifdef instead of \ifcsdef	220
\gls@assign@desc@field: changed to use \glssetnoexpandfield	22
\gls@assign@descplural@field: changed to use \glssetnoexpandfield	22
\gls@assign@name@field: changed to use \glssetnoexpandfield	22
\gls@assign@type@field: changed to use \glssetexpandfield	22
\gls@checkseeallowed: new	71
\glsaddallunused: set default to \@glo@types	164
\Glsentryfull: changed to use \acrfullformat	161
\glsentryfull: changed to use \acrfullformat	161
\Glsentryfullpl: changed to use \acrfullformat	161
\glsentryfullpl: changed to use \acrfullformat	161
\gls glossarymark: renamed \glossarymark to \gls glossarymark to avoid conflict with memoir	46
\glsprestandardsort: new	14
\glssetexpandfield: new	22
\glssetnoexpandfield: new	22
altsuper4colheader: switched to \tabularnewline	310
altsuper4colheaderborder: switched to \tabularnewline	311

long: switched to \tabularnewline	283
long3col: switched to \tabularnewline	284
long3colheader: switched to \tabularnewline	285
long3colheaderborder: switched to \tabularnewline	285
long4col: switched to \tabularnewline	286
long4colheader: switched to \tabularnewline	286
longheader: switched to \tabularnewline	284
longheaderborder: switched to \tabularnewline	284
\SetFootnoteAcronymDisplayStyle: fixed missing argument bug	252
super: switched to \tabularnewline	305
super3col: switched to \tabularnewline	307
super3colheader: switched to \tabularnewline	307
super4col: switched to \tabularnewline	308
super4colheader: switched to \tabularnewline	309
super4colheaderborder: switched to \tabularnewline	309
superheader: switched to \tabularnewline	306
superheaderborder: switched to \tabularnewline	306

3.14a (2013-11-12)

\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles	185
General: new	267
acronyms: new	18
\gls@def glossaryentry: added check for existence of default glossary	88
set the default for firstplural to be the value of plural	90
xindy gloss: new	31
\longprovide glossaryentry: new	86
compatible-2.07: added check for 2.07 before setting 3.07 compatibility	34
notranslate: new	27

\provideglossaryentry:new	77	index:new	36
4.0 (2013-11-14)		\newacronymstyle:new	231
\gls@defglossaryentry: added check		long-sc-short:new	234
for first key	90	long-sc-short-desc:new	235
super: fixed typo in \subglossentry		long-short:new	231
(\glossentrydesc)	305	long-short-desc:new	234
4.01 (2013-11-16)		long-sm-short:new	234
General: fixed non-value options so that		long-sm-short-desc:new	235
they can be passed to document class .	9	long-sp-short-desc:new	235
\CustomAcronymFields: inserted		footnote:new	239
missing comma	259	footnote-desc:new	241
4.02 (2013-12-05)		footnote-sc:new	240
\@acrfull: now using \acrfullfmt ..	225	footnote-sc-desc:new	241
\@gls@indexdef:new	37	footnote-sm:new	240
\@gls@numbersdef:new	36	footnote-sm-desc:new	241
\@gls@symbolsdef:new	36	\setacronymstyle:new	230
General: Removed \acronymfont .	150–154	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new	227	Moved check for empty custom text to	
\ACRfullfmt:new	226	prevent unwanted parenthetical	
\acrfullfmt:new	226	material	249
\ACRfullplfmt:new	228	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new	228	Moved check for empty custom text to	
\acrfullplfmt:new	227	prevent unwanted parenthetical	
\acronymentry:new	230	material	245
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here	26	Moved check for empty custom text to	
sc-short-long:new	234	prevent unwanted parenthetical	
sc-short-long-desc:new	236	material	251
\Genacrfullformat:new	112	\SetGenericNewAcronym:new	229
\genacrfullformat:new	112	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new	230	Moved check for empty custom text to	
\Genplacrfullformat:new	112	prevent unwanted parenthetical	
\genplacrfullformat:new	112	material	254
\Glsentryfull: bug fix: added missing		dua:new	236
\acronymfont	161	dua-desc:new	238
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont	161	option	9
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont	161	\makeglossaries: made preamble only	181
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont	161	General: changed default to \@empty	
\glsngenacfmt:new	110	instead of \relax	33
\GlsUseAcrEntryDisplayStyle:new ...	231	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new	231	\@do@esc@wrglossary: added	
short-long:new	233	\glsdetoklabel	192
short-long-desc:new	236	\@do@noesc@wrglossary: added	
xindynoglsnumbers:new	31	\glsdetoklabel	190
sm-short-long:new	234	\@ACRlong: removed \glslabel	
sm-short-long-desc:new	236	(defined in \@gls@link)	372

\@ACRshort: removed \glslabel (defined in \@gls@link)	371	\Genplacrfullformat: redefined to use accessibility information	370
\@Acrlong: removed \glslabel (defined in \@gls@link)	372	\genplacrfullformat: redefined to use accessibility information	370
\@Acrshort: removed \glslabel (defined in \@gls@link)	370	\glossentryname: added \glsdetoklabel	215
\@GLS@: removed \glslabel (defined in \@gls@link)	130	\gls@defglossaryentry: added \glsdetoklabel	86
\@GLSpl: removed \glslabel (defined in \@gls@link)	133	replaced #1 with \@gls@label	88
\@Gls@: removed \glslabel (defined in \@gls@link)	130	replaced \ifthenelse with \ifdefequal	89
\@Gls@entry@field: new	154	\glsadd: added \glsdetoklabel	163
\@Glspl@: removed \glslabel (defined in \@gls@link)	132	\glsaddkey: switched to using \@gls@field@link	81
\@acrlong: removed \glslabel (defined in \@gls@link)	371	\glsdetoklabel: new	59
\@acrshort: removed \glslabel (defined in \@gls@link)	370	\glsdisplaynumberlist: added \glsdetoklabel	162
\@gls@: removed \glslabel (defined in \@gls@link)	129	\glsdoifexistsorwarn: new	60
\@gls@access@display: new	357	\glsentryaccess: switched to using \@gls@entry@field	353
\@gls@entry@field: new	154	\glsentrydescaccess: switched to using \@gls@entry@field	354
\@gls@fetchfield: new	79	\glsentrydescpluralaccess: switched to using \@gls@entry@field	354
\@gls@field@link: new	134	\glsentryfirstaccess: switched to using \@gls@entry@field	354
\@gls@link: added \glsdetoklabel .	117	\glsentryfirstplural: added \glsdetoklabel	158
moved \@gls@link@opts and \@gls@link@label to \@gls@link	117	\glsentrylongaccess: switched to using \@gls@entry@field	355
\@gls@writedef: added \glsdetoklabel	78	\glsentrylongpluralaccess: switched to using \@gls@entry@field	355
\@glsdisp: removed \glslabel (defined in \@gls@link)	133	\glsentrypluralaccess: switched to using \@gls@entry@field	354
\@Glspl@: removed \glslabel (defined in \@gls@link)	131	\glsentryshortaccess: switched to using \@gls@entry@field	354
\@printglossary: added \glsdetoklabel	199	\glsentryshortpluralaccess: switched to using \@gls@entry@field	354
General: removed \glslabel (defined in \@gls@link)	147	\glsentrysymbolaccess: switched to using \@gls@entry@field	354
sc-short-long-desc: redefined to use accessibility information	382	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field	354
\compatibleglossentry: added \glsdetoklabel	347	\glsentrytextaccess: switched to using \@gls@entry@field	353
\compatiblesubglossentry: added \glsdetoklabel	348	\glsngenacfmt: redefined to use accessibility information	367
\Genacrfullformat: redefined to use accessibility information	369		
\genacrfullformat: redefined to use accessibility information	369		

\glsgenentryfmt: redefined to use accessibility information	364	sm-short-long-desc: redefined to use accessibility information	382
\glshyperlink: added \glsdetoklabel	163	long-sc-short-desc: redefined to use accessibility information	381
\glslocalreset: added \glsdetoklabel	95	long-short: redefined to use accessibility information	379
\glslocalunset: added \glsdetoklabel	96	long-short-desc: redefined to use accessibility information	380
\glsmoveentry: added \glsdetoklabel	93	long-sm-short-desc: redefined to use accessibility information	381
replaced \ifthenelse with \ifdefequal	93	footnote: redefined to use accessibility information	385
\glsrefentry: added \glsdetoklabel	214	footnote-desc: redefined to use accessibility information	387
\glsreset: added \glsdetoklabel ...	95	footnote-sc: redefined to use accessibility information	387
\glsseelist: added \expandafter commands	195	footnote-sc-desc: redefined to use accessibility information	388
\glsstepentry: added \glsdetoklabel	213	footnote-sm: redefined to use accessibility information	387
\glsstepsubentry: added \glsdetoklabel	213	footnote-sm-desc: redefined to use accessibility information	388
\glsunset: added \glsdetoklabel ...	96	\renewacronymstyle: new	231
short-long: commented spurious EOL	233	\showglocounter: added \glsdetoklabel	262
redefined to use accessibility information	380	\showglodesc: added \glsdetoklabel	263
short-long-desc: redefined to use accessibility information	381	\showglodescaccess: added \glsdetoklabel	394
\ifglshdescsuppressed: added \glsdetoklabel	62	\showglodescplural: added \glsdetoklabel	263
fixed typo	62	\showglodescpluralaccess: added \glsdetoklabel	394
\ifglshentryexists: added \glsdetoklabel	59	\showglofirst: added \glsdetoklabel	261
\ifglshaschildren: added \glsdetoklabel	61	\showglofirstaccess: added \glsdetoklabel	394
\ifglshasdesc: added \glsdetoklabel	62	\showglofirstpl: added \glsdetoklabel	261
\ifglshasfield: new	63	\showglofirstpluralaccess: added \glsdetoklabel	394
\ifglshaslong: added \glsdetoklabel	62	\showgloflag: added \glsdetoklabel	264
\ifglshasparent: added \glsdetoklabel	62	\showgloindex: added \glsdetoklabel	264
\ifglshasshort: added \glsdetoklabel	63	\showglolevel: added \glsdetoklabel	261
\ifglshassymbol: added \glsdetoklabel	62	\showglolong: added \glsdetoklabel	264
replaced \ifcempty with \ifdefempty and replaced \ifx with \ifdefequal	62	\showglolongaccess: added \glsdetoklabel	395
\ifglshused: added \glsdetoklabel ..	60		

\showglolongpluralaccess: added		redefined to use accessibility	
\glsdetoklabel	395	information	385
\showglongname: added \glsdetoklabel	263	4.04 (2014-03-04)	
\showglongnameaccess: added		\@gls@getcounterprefix: added	
\glsdetoklabel	393	warning if no prefix can be formed .	194
\showgloparent: added		4.04 (2014-03-06)	
\glsdetoklabel	261	\@@gls@noidx@nosanitizesort: new .	23
\showgloplural: added		\@@gls@noidx@sanitizesort: new ...	23
\glsdetoklabel	261	\@@gls@nosanitizesort: new	23
\showglopluralaccess: added		\@@gls@sanitizesort: new	23
\glsdetoklabel	394	\@glo@addchildren: new	201
\showgloshort: added		\@glo@do@sortentries: new	202
\glsdetoklabel	264	\@glo@grabfirst: new	207
\showgloshortaccess: added		\@glo@sortedinsert: new	202
\glsdetoklabel	394	\@glo@sortentries: new	201
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new	203
\glsdetoklabel	395	\@glo@sorthandler@letter: new ...	203
\showglosort: added \glsdetoklabel	263	\@glo@sorthandler@nocase: new ...	203
\showglosymbol: added		\@glo@sorthandler@word: new	203
\glsdetoklabel	264	\@glo@sortmacro@case: new	204
\showglosymbolaccess: added		\@glo@sortmacro@def: new	205
\glsdetoklabel	394	\@glo@sortmacro@def@do: new	205
\showglosymbolplural: added		\@glo@sortmacro@letter: new	204
\glsdetoklabel	264	\@glo@sortmacro@nocase: new	205
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ...	204
\glsdetoklabel	394	\@glo@sortmacro@use: new	206
\showglotext: added \glsdetoklabel	261	\@glo@sortmacro@word: new	203
\showglotextaccess: added		\@gls@noidx@do: new	207
\glsdetoklabel	394	\@gls@noidx@getgrouptitle: new ..	219
\showglotype: added \glsdetoklabel	262	\@gls@noref@warn: new	185
\showglouseri: added		\@gls@reference: new	209
\glsdetoklabel	262	\@gls@warnonglossdefined: new	21
\showglouserii: added		\@gls@warnontheGLOSSdefined: new .	21
\glsdetoklabel	262	\@no@makeglossaries: new	185
\showglouseriii: added		\@print@glossary: new	199
\glsdetoklabel	262	\@print@noidx@glossary: new	206
\showglouseriv: added		\@printgloss@setsort: new	197
\glsdetoklabel	262	\@printglossary: new	197
\showglouserv: added		General: added sort key to printgloss	
\glsdetoklabel	263	group	212
\showglouservi: added		\compatibleglossentry: changed	
\glsdetoklabel	263	\newcommand to \def as is may or	
dua: fixed bug in \acrfullfmt	238	may not be defined	347
fixed bug in \Acrfullplfmt	238	\compatiblesubglossentry: changed	
fixed bug in \acrfullplfmt	238	\newcommand to \def as is may or	
redefined to use accessibility		may not be defined	348
information	382	\defglsdisplayfirst: fixed unwanted	
dua-desc: commented spurious EOL ..	239	space	113
		\glo@grabfirst: new	207

\gls@defglossaryentry: replaced \ifx with \ifdefvoid	92	4.08 (2014-07-30)	\@ACRlong: added \do@gls@link@checkfirsthyper	372
\glsnoidxdisplayloc: new	209		\@ACRshort: added \do@gls@link@checkfirsthyper	371
\glsnoidxdisplayloclisthandler: new	209		\@Acrlong: added \do@gls@link@checkfirsthyper	371
\glsnoidxloclist: new	209		\@Acrshort: added \do@gls@link@checkfirsthyper	370
\glsnoidxloclisthandler: new	209		\@GLS@: moved \glsifhyper	130
\glsnoidxstripaccents: new	24		moved check for first use to \@gls@link	130
alttree: moved hangindent and parindent assignments outside level test	324		\@GLSpl: moved \glsifhyper	133
\makeglossaries: Moved definition of \glswrite to \makeglossaries ..	179		moved check for first use to \@gls@link	133
\makenoidxglossaries: new	181		\@Gls@: moved \glsifhyper	130
\printglossary: changed to use new \@printglossary	196		moved check for first use to \@gls@link	130
\printnoidxglossaries: new	197		\@Glspl@: moved \glsifhyper	132
\printnoidxglossary: new	197		moved check for first use to \@gls@link	132
\showgloclist: new	265		\@acrlong: added \do@gls@link@checkfirsthyper	371
\warn@noprintglossary: Activate warning in \makeglossaries	196		\@acrshort: added \do@gls@link@checkfirsthyper	370
\writeist: checked for definition of \glswrite	166, 171		\@closegls: new	176
4.06 (2014-03-12)			\@gls@: moved \glsifhyper	129
\@GLS@: added \glsifhyper	130		moved check for first use to \@gls@link	129
\@GLSpl: added \glsifhyper	133		\@gls@automake: new	176
\@Gls@: added \glsifhyper	130		\@gls@doautomake: new	33
\@Glspl@: added \glsifhyper	132		\@gls@field@link: added assignment of \do@gls@link@checkfirsthyper	134
\@gls@: added \glsifhyper	129		\@gls@forbidtexext: new	65
\@gls@numbersdef: added hook to set toc title	36		\@gls@hyp@opt: new	115
\@gls@symbolsdef: added hook to set toc title	36		\@gls@link: removed redundancy	117
\@glsdisp: added \glsifhyper	134		renamed \gls@type to \glstype ...	117
\@glspl@: added \glsifhyper	131		\@gls@link@checkfirsthyper: new .	116
General: added \glsifhyper	147–154		\@glsdisp: moved \glsifhyper	134
acronym: added hook to set toc title	18		moved check for first use to \@gls@link	134
acronyms: added hook to set toc title ...	18		\@glspl@: moved \glsifhyper	131
\glsdefmain: added hook to set toc title	17		moved check for first use to \@gls@link	131
4.07 (2014-04-04)			\@ignored@glossaries: new	68
\@glossarysection: added optional argument when using unstarred version	47		General: added entrycounter option to printgloss family	211
\@gls@noidx@do: added \global in case it's used in a tabular-like style	207			
\Acrfullplfmt: fixed no case change bug	228			
\glsletentryfield: new	154			

added nopostdot option to printgloss family	211	removed \@sglstext	134
added subentrycounter option to printgloss family	211	removed \@sGLSuseriii	144
explicitly initialise hyper key	114	removed \@sGlsuseriii	144
moved \glsifhyper	147–154	removed \@sglsuseriii	143
removed \@sACRlongpl	153	removed \@sGLSuserii	143
removed \@sAcrlongpl	153	removed \@sGlsuserii	143
removed \@sacrlongpl	152	removed \@sGLSuseriv	145
removed \@sACRlong	151	removed \@sGlsuseriv	144
removed \@sAcrlong	151	removed \@sglsuseriv	144
removed \@sacrlong	150	removed \@sGLSuseri	142
removed \@sACRshortpl	149	removed \@sGlsuseri	142
removed \@sAcrshortpl	149	removed \@sglsuseri	142
removed \@sacrshortpl	148	removed \@sGLSservi	146
removed \@sACRshort	148	removed \@sGlsservi	146
removed \@sAcrshort	147	removed \@sglsuseri	146
removed \@sacrshort	147	removed \@sGLSserv	145
removed \@sgls@link	116	removed \@sGlsuseri	145
removed \@sGLSdescplural	140	removed \@sGls	130
removed \@sGlsdescplural	140	removed \@sGls	129
removed \@sglsdescplural	139	removed \@sgls	128
removed \@sGLSdesc	139	removed \@thirdofthree (defined in kernel)	128
removed \@sGlsdesc	139	removed sPGLS	272
removed \@sglsdesc	138	removed sPgls	271
removed \@sglsdisp	133	removed spgls	269
removed \@sGLSfirstplural	137	removed sPGLSpl	273
removed \@sGlsfirstplural	137	removed sPglspl	271
removed \@sglsfirstplural	137	removed spglspl	270
removed \@sGLSfirst	136	\ACRfull: removed \s@ACRfull	227
removed \@sGlsfirst	135	switched to using \@gls@hyp@opt ..	226
removed \@sGlsfirst	135	\Acrfull: removed \s@Acrfull	226
removed \@sGLSname	138	switched to using \@gls@hyp@opt ..	226
removed \@sGlsname	138	\acrfull: removed \@sacrfull	225
removed \@sglsname	138	switched to using \@gls@hyp@opt ..	225
removed \@sGLSplural	137	\ACRfullpl: removed \s@ACRfullpl ..	228
removed \@sGlsplural	136	switched to using \@gls@hyp@opt ..	228
removed \@sglsplural	136	\Acrfullpl: removed \s@Acrfullpl ..	227
removed \@sGLSpl	132	switched to using \@gls@hyp@opt ..	227
removed \@sGlspl	132	\acrfullpl: removed \s@acrfullpl ..	227
removed \@sglspl	131	switched to using \@gls@hyp@opt ..	227
removed \@sGLSsymbolplural	141	\ACRlong: switched to using \@gls@hyp@opt	151
removed \@sGlsymbolplural	141	\Acrlong: switched to using \@gls@hyp@opt	151
removed \@sglsymbolplural	141	\acrlong: switched to using \@gls@hyp@opt	150
removed \@sGLSsymbol	140		
removed \@sglsymbol	140		
removed \@sGLStext	135		
removed \@sGlstext	135		

\ACRlongpl: switched to using \@gls@hyp@opt	153	\glsenablehyper: added \KV@glslink@hypertrue to definition	128
\Acrlongpl: switched to using \@gls@hyp@opt	153	\GLSfirst: switched to using \@gls@hyp@opt	136
\acrlongpl: switched to using \@gls@hyp@opt	152	\Glsfirst: switched to using \@gls@hyp@opt	135
\ACRshort: switched to using \@gls@hyp@opt	148	\glsfirst: switched to using \@gls@hyp@opt	135
\Acrshort: switched to using \@gls@hyp@opt	147	\GLSfirstplural: switched to using \@gls@hyp@opt	137
\acrshort: switched to using \@gls@hyp@opt	146	\Glsfirstplural: switched to using \@gls@hyp@opt	137
\ACRshorttpl: switched to using \@gls@hyp@opt	149	\glsfirstplural: switched to using \@gls@hyp@opt	137
\Acrshorttpl: switched to using \@gls@hyp@opt	149	\glsifhyper: deprecated	115
\acrshorttpl: switched to using \@gls@hyp@opt	148	\glslink: switched to using \@gls@hyp@opt	116
\forallacronyms: new	58	\glslinkcheckfirsthyperhook: new	117
\GLS: switched to using \@gls@hyp@opt	130	\glslinkvar: new	115
\Gls: switched to using \@gls@hyp@opt	129	\GLSname: switched to using \@gls@hyp@opt	138
\gls: switched to using \@gls@hyp@opt	128	\Glsname: switched to using \@gls@hyp@opt	138
\gls@defglossaryentry: added check for ignored glossary	88	\glsname: switched to using \@gls@hyp@opt	138
\gls@istfilebase: new	43	\GLSpl: switched to using \@gls@hyp@opt	132
\glsaddkey: removed \@sGLS@user@<key>	82	\Glspl: switched to using \@gls@hyp@opt	132
removed \@sGls@user@<key>	82	\glspl: switched to using \@gls@hyp@opt	131
removed \@sgls@user@<key>	81	\GLSplural: switched to using \@gls@hyp@opt	136
switched to using \@gls@hyp@opt	81, 82	\Glsplural: switched to using \@gls@hyp@opt	136
\GLSdesc: switched to using \@gls@hyp@opt	139	\glsplural: switched to using \@gls@hyp@opt	136
\Glsdesc: switched to using \@gls@hyp@opt	139	\glsspace: new	226
\glsdesc: switched to using \@gls@hyp@opt	138	\GLSsymbol: switched to using \@gls@hyp@opt	141
\GLSdescplural: switched to using \@gls@hyp@opt	140	\Glsymbol: switched to using \@gls@hyp@opt	140
\Glsdescplural: switched to using \@gls@hyp@opt	139	\glsymbol: switched to using \@gls@hyp@opt	140
\glsdescplural: switched to using \@gls@hyp@opt	139	\GLSsymbolplural: switched to using \@gls@hyp@opt	141
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	128	\Glsymbolplural: switched to using \@gls@hyp@opt	141
\glsdisp: switched to using \@gls@hyp@opt	133		
\glsdohyperlink: new	127		
\glsdohypertarget: new	127		

\glssymbolplural: switched to using \@gls@hyp@opt	141	\newglossary: added starred version ..	66
\GLStext: switched to using \@gls@hyp@opt	135	\newignoredglossary: new	68
\Glstext: switched to using \@gls@hyp@opt	135	\ns@newglossary: added \@glotype@<name>@log	66
\glstext: switched to using \@gls@hyp@opt	134	new	66
\glstreenamefmt: new	317	\p@gls@hyp@opt: new	115
\GLSuseri: switched to using \@gls@hyp@opt	142	\PGLS: changed to use \@gls@hyp@opt	272
\Glsuseri: switched to using \@gls@hyp@opt	142	\Pgls: changed to use \@gls@hyp@opt	271
\glsuseri: switched to using \@gls@hyp@opt	142	\pgls: changed to use \@gls@hyp@opt	269
\GLSuserii: switched to using \@gls@hyp@opt	143	\PGLSpl: changed to use \@gls@hyp@opt	273
\Glsuserii: switched to using \@gls@hyp@opt	143	\Pglspl: changed to use \@gls@hyp@opt	271
\GLSuseriii: switched to using \@gls@hyp@opt	144	\pglspl: changed to use \@gls@hyp@opt	270
\Glsuseriii: switched to using \@gls@hyp@opt	144	\s@gls@hyp@opt: new	115
\glsuseriii: switched to using \@gls@hyp@opt	143	\s@newglossary: new	66
\GLSuseriv: switched to using \@gls@hyp@opt	145	automake: new	33
\Glsuseriv: switched to using \@gls@hyp@opt	144	4.09 (2014-08-12)	
\glsuseriv: switched to using \@gls@hyp@opt	144	\glsaddkey: fixed bug in user commands	81
\GLSuseriv: switched to using \@gls@hyp@opt	145	4.10 (2014-08-27)	
\Glsuseriv: switched to using \@gls@hyp@opt	144	\@Gls@acentryname: new	155
\GLSuserv: switched to using \@gls@hyp@opt	145	\@Gls@entryname: new	155
\Glsuserv: switched to using \@gls@hyp@opt	145	\@gls@glossary: Renamed \@glossary to \@gls@glossary	186
\GLSuservi: switched to using \@gls@hyp@opt	146	\glspercentchar: new	165
\Glsuservi: switched to using \@gls@hyp@opt	146	\glstildechar: new	165
\ifignoredglossary: new	68	alttree: moved space after symbol	324, 325
altlongragged4col: fixed bug that displayed description instead of symbol	297	4.11 (2014-09-01)	
		\@@do@esc@wrglossary: added hook ..	192
		sanitize: none option	26
		\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	187
		\glsaddprotectedpagefmt: new	189
		\glsbackslash: new	165
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute	52
		\@gls@adjustmode: new	164
		\@gls@notranslatorhook: removed ...	27
		\@gls@toc: added \protect to \numberline	49
		\@gls@usetranslator: new	27
		\glsacrpluralsuffix: new	40
		\glsadd: added check for vertical mode	163
		\glsaddallunused: replaced @gobble with glsignore	164
		\glsifusedtranslatordict: new	27
		\glsignore: new	164

\glsupacrpluralsuffix:new	40	4.16 (2015-06-18)	
\ProvidesGlossariesLang:new	40	\glsaddstoragekey:new	80
\RequireGlossariesLang:new	40	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	372
\indexspace:new	279, 299, 317	\@ACRshort: added \glspostlinkhook	371
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	372
\@glslocalreset:new	97	\@Acrshort: added \glspostlinkhook	371
\@glslocalunset:new	96	\@GLS@: added \glspostlinkhook ...	131
\@glsreset:new	97	\@GLSpl: added \glspostlinkhook ..	133
\@glsunset:new	96	\@Gls@: added \glspostlinkhook ...	130
\@newglossaryentry@defcounters:		\@Glspl@: added \glspostlinkhook ..	132
new	98	\@acrlong: added \glspostlinkhook	371
\cGls:new	101	\@acrshort: added \glspostlinkhook	370
\cGls@:new	101	\@gls@: added \glspostlinkhook ...	129
\cGlspl@:new	102	\@gls@@link: added	
\cgls:new	101	\glspostlinkhook	116
\cgls@:new	101	\@gls@field@link: added	
\cglspl:new	102	\glspostlinkhook	134
\cglspl@:new	102	\@gls@link: moved definition of	
\gls@entry@count:new	101	\glsifhyperon outside of this	
\gls@increment@currcount:new ..	100	macro	118
\gls@local@increment@currcount:		\@glsdisp: added \glspostlinkhook	134
new	100	\@glspl@: added \glspostlinkhook ..	131
\gls@write@entrycounts:new	100	General: added \glspostlinkhook	147–154
\glslocalreset:new	97	\glsacspace:new	233
\glslocalunset:new	96	\glsadd: changed \@do@wrglossary to	
\glsreset:new	97	\@do@wrglossary	163
\glsunset:new	96	\glsfielddef:new	84
\@newglossaryentry@defcounters:		\glsfieldedef:new	83
new	92	\glsfieldfetch:new	84
\cGls:new	101	\glsfieldgdef:new	83
\cgls:new	101	\glsfieldxdef:new	83
\cGlsformat:new	102	\glsifhyperon: moved definition of	
\cglsformat:new	101	\glsifhyperon	117
\cGlspl:new	102	\glslinkpostsetkeys:new	117
\cglspl:new	102	\glspostlinkhook:new	116
\cGlsplformat:new	103	\glswriteentry:new	188
\cglsplformat:new	102	\ifglsfieldcseq:new	85
\gls@defdocnewglossaryentry:new ..	77	\ifglsfielddefeq:new	85
\glsenableentrycount:new	98	\ifglsfieldeq:new	85
\glslocalreset: switched to		long-sp-short:new	232
\@glslocalreset	95	\showglofield:new	265
\glslocalunset: switched to		4.18 (2015-09-09)	
\@glslocalunset	96	General: split mfirstuc into separate	
\glsreset: switched to \@glsreset ...	95	bundle	4
\glsunset: switched to \@glsunset ...	96	4.19 (2015-10-31)	
4.15 (2015-03-16)		\glsstreenamibox:new	323
General: bug fix replaced \@glo@type		4.19 (2015-11-22)	
with \glstype	153	\@gls@link@nocheckfirsthyper:new	134

\@gls@preglossaryhook: new	197	\glsfindwidesttoplevelname: new .	323
\@printglossary: added		\glslistgroupheaderfmt: new	279
\@gls@preglossaryhook	199	\glslistnavigationitem: new	279
\do@glsglisablehyperinlist: new ..	117	\glstreegroupheaderfmt: new	317
\doifglossarynoexistsordo: new ...	61	\glstreenavigationfmt: new	317
\gls@gobbleopt: new	65	\ifglswrallowprimitivemods: new .	190
\glsglsoifexistsordo: new	60	list: fixed missing space before	
4.20 (2015-11-30)		description	279
\@gls@link: added		long: fixed typo in \glossentrydesc .	283
\@gls@setdefault@glslink@opts	117	super4col: fixed bug in \glossentry .	308
added \glsglsonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed	118	\glsglscurrentfieldvalue: new	64
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new	117	\glsglscurrentfieldvalue	63, 64
\glsgl@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	71	nogroupskip changed	297
\glsglsonohyperlink: new	127	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	316
\@printglossary: warn if no style has		long: check for nogroupskip changed ..	283
been set	197	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	289
assignment	147–153	long3col: check for nogroupskip	
\glsglstyle: set default style if not		changed	285
already set	220	long3col-booktabs: check for	
\glsglLTpenaltycheck: new	292	nogroupskip changed	290
\glsglpatchLToutput: new	292	long4col: check for nogroupskip	
\glsglspenaltygroupskip: new	292	changed	286
altlong4col-booktabs: new	290	long4col-booktabs: check for	
altlongragged4col-booktabs: new .	291	nogroupskip changed	290
long-booktabs: new	289	longragged: check for nogroupskip	
long3col-booktabs: new	289	changed	294
long4col-booktabs: new	290	longragged3col: check for nogroupskip	
longragged-booktabs: new	291	changed	295
longragged3col-booktabs: new	291	super: check for nogroupskip changed .	305
\setglsglstyle: set default style if		super3col: check for nogroupskip	
not already set	220	changed	307
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@esc@wrglossary: added check		changed	309
for \@arabic	191	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	312
modifications and added arabic case	191	superragged3col: check for	
mcolalttreespannav: new	304	nogroupskip changed	314
mcolindexspannav: new	300	4.24 (2016-05-27)	
mcoltreenonamespannav: new	302	\@glsgl@extramakeindexopts: new ...	175
mcoltreespannav: new	301	\@glsgl@glossary: added check for debug	
\glsgl@arabicpage: new	188	mode	186
\glsgl@protected@pagefmts: added		\@glsgl@see@noindex: new	7
arabic to list	188	debug: new	5
\glsglentrytitlecase: new	158	seenoindex: new	7

\glsnomakeindexwarning: new	49	\gls@set@xr@key: new	71
\GlsSetQuote: new	172	\gls@xr@key: new	71
\GlsSetWriteIstHook: new	172	\GlsAddXdyLocation: bug fix: changed	
4.25 (2016-06-09)		#1 to #2	55
\@gls@enablesavenonumberlist: new	72	\glsnoidxstripaccents: added \a ...	24
\@gls@initnonumberlist: new	72	added \TH, \dh and \DH	25
\@gls@savenonumberlist: new	72	4.31 (2017-08-10)	
4.25 (??)		nolist: added check for “list” style	11
General: changed		4.31 (2017-09-10)	
\DeclareRobustCommand to		style: changed \renewcommand to \def .	9
\newrobustcmd and changed		4.32 (2017-08-24)	
\@ifundefined to \ifcsundef ...	360	\@glsnavhypertarget: new	274
4.26 (2016-10-12)		\@glsshowtarget: new	7
\@glossary@default@style: added		\glsshowtarget: new	6
check for classicthesis	9	4.33 (2017-09-20)	
mcolindex: replaced \@idxitem with		\@do@esc@wrglossary: added	
\glstreeitem	299	\gls@the and \gls@number	191
mcolindexspannav: replaced \@idxitem		renamed from	
with \glstreeitem	300	\@do@esc@wrglossary	190
\glstreechildpredesc: new	318	\@do@noesc@wrglossary: new	189
\glstreeitem: new	317	\@do@wrglossary: changed to check	
\glstreepredesc: new	318	for esclocations	189
\glstreesubitem: new	318	\@gls@missinglang@warn: new	21
\glstreesubsubitem: new	318	\GlsSetXdyFirstLetterAfterDigits:	
4.28 (2017-01-07)		added starred version	165
\glspatchtabularx: new	95	\GlsSetXdyNumberGroupOrder: new .	166
4.29 (2017-01-19)		esclocations: new	10
\@gls@noidx@do: current letter group		4.34 (2017-11-03)	
assignment made global	208	mcolalttreespannav: removed spurious	
\@print@noidx@glossary: moved		space	304
definition of		\glsshowtarget: modified to check for	
\@gls@currentlettergroup outside		math mode and inner	6
of the glossary environment	206	4.35 (2017-11-14)	
General: added check for		\glsadd: added \@gls@setsort (in case	
\@glsxtr@doaccsupp	347	of sort=use)	163
\glsnavhyperlinkname: new	274	4.36 (2018-03-07)	
4.30 (2017-06-11)		\@gls@glossary: removed \index ...	187
\@glo@autosee: new	92	4.37 (2018-04-07)	
\@glo@autoseehook: new	92	\gls@begindocdefs: new	77
\@glo@check@sortallowed: new	14	4.38 (2018-05-10)	
\@gls@noidx@do: letter group		\@gls@define@glossaryentrycounter:	
assignment made global	208	added check for existence of	
\@gls@setupsort@def: added check for		glossaryentry counter	12
register	15	new	12
\@gls@setupsort@none: new	16	\@gls@define@glossarysubentrycounter:	
\@xdycrossrefhook: new	54	new	13
\@xdylocationclassorder: bug fix:		prepended \currentglossary. to	
changed \edef to \def	55	\theHglossarysubentry and	
\glosortentrieswarning: new	21	removed spurious eol space	13

\glsaccsupp: added braces around actual text argument	357	numberedsection: changed \val and \nr to \gls@numberedsection@val and \gls@numberedsection@nr	9
\glsentrycounterlabel: bug fix: move conditional inside command	214	4.42 (2019-01-06)	
\GlsEntryCounterLabelPrefix: new	211	\@gls@@automake@immediate: new ..	179
\glsentryitem: bug fix: move conditional inside command	214	\@gls@automake@immediate: new ...	177
\glsrefentry: bug fix: move conditional inside command	214	\gls@automake@nr: new	33
\glsresetsubentrycounter: bug fix: move conditional inside command .	213	\glsfieldedef: changed from \edef to \protected@csedef	83
\glsstepentry: bug fix: move conditional inside command	213	\glsfieldxdef: changed from \edef to \protected@csxdef	83
\glsstepsubentry: bug fix: move conditional inside command	213	\ifglsautomake: now defined explicitly instead of through boolean key	32
\glssubentrycounterlabel: bug fix: move conditional inside command .	214	noglossaryindex: new	36
\glssubentryitem: bug fix: move conditional inside command	214	automake: switch from boolean to choice	33
\showglongnameaccess: bug fix: corrected field (was showing text access field)	393	4.42 (??)	
4.40 (2018-06-01)		altlong4col-booktabs: removed superfluous \glspatchLToutput .	290
\istfile: changed \def to \providecommand	185	4.43 (2019-09-28)	
\makenoidxglossaries: false	181	\glsnoidxstripaccents: add check for LaTeX version 2019/10/01	25
4.41 (2018-07-23)		4.44 (2019-12-06)	
\@gls@override@glossary: new	34	\@gls@prefix@record@hook: new	269
General: changed \val and \nr to \gls@numberedsection@val and \gls@numberedsection@nr	211	4.45 (2020-02-13)	
debug: changed \val and \nr to \gls@debug@val and \gls@debug@nr	5	\@do@write@glslabels: new	32
seenoindex: changed \val and \nr to \gls@seenoindex@val and \gls@seenoindex@nr	7	\@glsshows@target: new	7
kernelglossredefs: new	35	\@GLSdesc@: added accessibility support	374
\glossary: added warning	35	\@GLSdescplural@: added accessibility support	375
\gls@original@glossary: new	34	\@GLSfirst@: added accessibility support	373
\gls@original@makeglossary: new ..	34	\@GLSfirstplural@: added accessibility support	373
\makeglossaries: removed redefinition of \makeglossary	180	\@GLSname@: added accessibility support	374
\makeglossary: added warning	34	\@GLSplural@: added accessibility support	373
nonumberlist: changed \val and \nr to \gls@nonumberlist@val and \gls@nonumberlist@nr	72	\@GLSsymbol@: added accessibility support	375
translate: changed \val and \nr to \gls@translate@val and \gls@translate@nr	28	\@GLSsymbolplural@: added accessibility support	375
		\@GLStext@: added accessibility support	372
		\@GLSuseri@: added accessibility support	376
		\@GLSuserii@: added accessibility support	376
		\@GLSuseriii@: added accessibility support	376
		\@GLSuseriv@: added accessibility support	377

\@GLSuser@: added accessibility support	377	\@glssymbolplural@: added accessibility support	375
\@GLSuseri@: added accessibility support	377	\@glstext@: added accessibility support	372
\@Gls@acentryname: added check for \glsshortaccessdisplay	156	\@glseri@: added accessibility support	375
\@Glsdesc@: added accessibility support	374	\@glserii@: added accessibility support	376
\@Glsdescplural@: added accessibility support	374	\@glseriii@: added accessibility support	376
\@Glsfirst@: added accessibility support	373	\@glseriv@: added accessibility support	377
\@Glsfirstplural@: added accessibility support	373	\@glserv@: added accessibility support	377
\@Glsname@: added accessibility support	374	\@glservi@: added accessibility support	377
\@Glsplural@: added accessibility support	373	General: removed backward compatibility use of symbol key	351
\@Glsymbol@: added accessibility support	375	acronyms: changed \renewcommand to \def	18
\@Glsymbolplural@: added accessibility support	375	debug: showaccsupp	5
\@glstext@: added accessibility support	372	restoremakeloss: new	32
\@glseri@: added accessibility support	376	\@gl@accessibility: new	348
\@glserii@: added accessibility support	376	\@gl@accsupp@engine: new	348
\@glseriii@: added accessibility support	376	\@gl@accessibility: new	348
\@glseriv@: added accessibility support	377	\@gl@defaultshortaccess: new	388
\@glserv@: added accessibility support	377	\@gl@descriptionaccessdisplay: added check for existence	358
\@glservi@: added accessibility support	377	\@gl@descriptionpluralaccessdisplay: added check for existence	358
\@do@write@glslabels: new	32	\@gl@entrydescpluralaccess: corrected field reference	354
\@domakeglossaries: new	31	\@gl@entryfirstpluralaccess: switched to using \@gl@entry@field	354
\@gl@fieldaccess@display: new	357	\@gl@entryparent: new	159
\@gl@desc@: added accessibility support	374	\@gl@entryprefix: added \@gl@detoklabel	268
\@gl@descplural@: added accessibility support	374	\@gl@entryprefix: added \@gl@detoklabel	268
\@gl@first@: added accessibility support	373	\@gl@entryprefixfirst: added \@gl@detoklabel	268
\@gl@firstplural@: added accessibility support	373	\@gl@entryprefixfirst: added \@gl@detoklabel	268
\@gl@name@: added accessibility support	374	\@gl@entryprefixfirstplural: added \@gl@detoklabel	268
\@gl@plural@: added accessibility support	373	\@gl@entryprefixfirstplural: added \@gl@detoklabel	268
\@gl@showaccsupp: new	7	\@gl@entryprefixplural: added \@gl@detoklabel	268
\@gl@symbol@: added accessibility support	375		

\glentryprefixplural: added		\glsshowtargetouter: new	6
\glsdetoklabel	268	\glsshowtargetsymbol: new	6
\glentrytitlecase: added existence		\glssymbolaccessdisplay: added	
check	158	check for existence	358
\glentryuseriaccess: new	355	\glssymbolpluralaccessdisplay:	
\glentryuseriiaccess: new	355	added check for existence	358
\glentryuseriiiaccess: new	355	\glstextaccessdisplay: added check	
\glentryuserivaccess: new	355	for existence	357
\glentryuservaccess: new	355	\gluseriaccessdisplay: new	359
\glentryuserviaccess: new	355	\gluseriiaccessdisplay: new	359
\glfieldaccsupp: new	356	\gluseriiiaccessdisplay: new	359
\glfirstaccessdisplay: added check		\gluserivaccessdisplay: new	360
for existence	358	\gluservaccessdisplay: new	360
\glfirstpluralaccessdisplay:		\gluserviaccessdisplay: new	360
added check for existence	358	\ifglshaschildren: made robust	61
\glslongaccessdisplay: added check		\ifglshasfield: made robust	63
for existence	359	\ifglshaslong: made robust	62
\glslongpluralaccessdisplay: added		\ifglshasprefix: added	
check for existence	359	\glsdetoklabel	269
\glnameaccessdisplay: added check		\ifglshasprefixfirst: added	
for existence	357	\glsdetoklabel	269
\glsppluralaccessdisplay: added		\ifglshasprefixfirstplural: added	
check for existence	358	\glsdetoklabel	269
\glsprefixsep: new	269	\ifglshasprefixplural: added	
\glsee: switched to \newrobustcmd	195	\glsdetoklabel	269
\glseeformat: switched to		\ifglshasshort: made robust	63
\newrobustcmd	195	\ifglshassymbol: made robust	62
\glseeitem: switched to		disablemakegloss: new	31
\newrobustcmd	196	\makeglossaries: let \@makeglossary	
\glseeelist: switched to		to \@gobble instead of \relax	180
\newrobustcmd	195	writelabels: new	32
\glsshortaccessdisplay: added check		user1access: new	350
for existence	359	user2access: new	350
\glsshortaccsupp: new	356	user3access: new	350
\glsshortplaccsupp: new	357	user4access: new	351
\glsshortpluralaccessdisplay:		user5access: new	351
added check for existence	359	user6access: new	351
\glsshowaccsupp: new	7	\xglfieldaccsupp: new	356
\glsshowtargetfont: new	6		

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	
\!	123, 124
\"	24, 121–124, 126
\#	169
\%	165, 171, 331, 332
\&	40, 162
\'	24
\.	11, 24
\=	24
\?	121–123, 174
\@	78
\@@delimN	222
\@do@wrglossary	182, 190, 192
\@do@esc@wrglossary	189
\@do@noesc@wrglossary	189
\@do@wrglossary	163, 188
\@do@write@glslabels	32
\@glo@assign@sortkey	182
\@glo@list	58
\@glo@sort	24
\@glo@type	197
\@glossarysec	8, 47, 48
\@glossaryseclabel	9, 47, 48, 211
\@glossarysecstar	9, 47, 48, 211
\@gls@checkactual	125
\@gls@checkbar	124
\@gls@checkescactual	123
\@gls@checkescbar	123
\@gls@checkesclevel	124
\@gls@checkescquote	122, 174, 175
\@gls@checklevel	125
\@gls@checkquote	121, 122, 173
\@gls@default@entryfmt	104, 113
\@gls@expand@field	22, 75, 76, 80, 81, 244, 246, 248, 250, 252, 255, 257, 389–393
\@gls@extramakeindexopts	172, 179
\@gls@fixbraces	194
\@@gls@noexpand@field	22, 74, 75
\@@gls@noidx@no@sanitizesort	23, 24
\@@gls@noidx@nosanitizesort	184
\@@gls@nosanitizesort	23, 184
\@@gls@sanitizesort	23, 184
\@@gls@xdycheckbackslash	126, 127
\@@gls@xdycheckquote	126
\@@gls@localreset	97, 99
\@@gls@localunset	96, 99
\@@gls@reset	97, 99
\@@gls@showtarget	6, 7
\@@gls@unset	96, 99
\@newglossaryentry@defcounters	98
\@this@glo@	59
\@ACRfull	227
\@ACRfullpl	228
\@ACRlong	151, 227
\@ACRlongpl	153, 228
\@ACRshort	148, 227
\@ACRshortpl	149, 150, 228
\@Acrfull	226
\@Acrfullpl	227
\@Acrlong	151, 226
\@Acrlongpl	153, 228
\@Acrshort	147
\@Acrshortpl	149
\@Alph	188, 191, 192
\@GLS	130
\@GLS@	130, 273
\@GLSdesc	139
\@GLSdesc@	139
\@GLSdescplural	140
\@GLSdescplural@	140
\@GLSfirst	136
\@GLSfirst@	136
\@GLSfirstplural	137
\@GLSfirstplural@	137

\@GLSname	138	\@Glstext	135
\@GLSname@	138	\@Glstext@	135
\@GLSpl	132	\@Glsuser@i	376, 377
\@GLSpl@	132, 133, 273	\@Glsuseri	142
\@GLSplural	136, 137	\@Glsuseri@	142
\@GLSplural@	137	\@Glsuserii	143
\@GLSsymbol	141	\@Glsuserii@	143
\@GLSsymbol@	141	\@Glsuseriii	144
\@GLSsymbolplural	141	\@Glsuseriii@	144
\@GLSsymbolplural@	141, 142	\@Glsuseriv	144
\@GLStext	135	\@Glsuseriv@	144
\@GLStext@	135	\@Glsuserv	145
\@GLSuseri	142	\@Glsuserv@	145
\@GLSuseri@	142	\@Glsuservi	146
\@GLSuserii	143	\@Glsuservi@	146
\@GLSuserii@	143	\@Mi	292
\@GLSuseriii	144	\@PGLS	272
\@GLSuseriii@	144	\@PGLS@	272
\@GLSuseriv	145	\@PGLSpl	273
\@GLSuseriv@	145	\@PGLSpl@	273
\@GLSuserv	145	\@Pgls	271
\@GLSuserv@	145, 146	\@Pgls@	271
\@GLSuservi	146	\@Pglspl	271
\@GLSuservi@	146	\@Pglspl@	271
\@Gls	129	\@Roman	189, 191, 192
\@Gls@	99, 101, 129, 271	\@acrfull	225
\@Gls@acrentryname	229	\@acrfullpl	227
\@Gls@entry@field	81, 155–161	\@acrlong	150, 226
\@Gls@entryname	155, 229	\@acrlongpl	152, 227
\@GlsSetXdyFirstLetterAfterDigits	165	\@acrshort	147, 226
\@GlsSetXdyNumberGroupOrder	166	\@acrshortpl	148, 227, 228
\@Glsdesc	139	\@addtoacronymlists	18, 19
\@Glsdesc@	139	\@afterheading	280, 335
\@Glsdescplural	139, 140	\@alph	188, 191, 192
\@Glsdescplural@	140	\@arabic	188, 191, 192
\@Glsfirst	135	\@auxout	65, 66, 100, 179, 182, 185, 196, 200, 275
\@Glsfirst@	135, 136	\@backslashchar	120, 126, 127
\@Glsfirstplural	137	\@bsphack	187
\@Glsfirstplural@	137	\@cGls	101
\@Glsname	138	\@cGls@	99, 101
\@Glsname@	138	\@cGlspl	102
\@Glspl	132	\@cGlspl@	100, 102
\@Glspl@	100, 102, 132, 272	\@cclv	292, 293
\@Glsplural	136	\@cgls	101
\@Glsplural@	136	\@cgls@	99, 101
\@Glsymbol	140	\@cglspl	102
\@Glsymbol@	140	\@cglspl@	99, 102
\@Glsymbolplural	141	\@chapter	38
\@Glsymbolplural@	141	\@classoptionslist	37

<code>\@closegls</code>	176, 177	<code>\@glo@addchildren</code>	201, 205
<code>\@colht</code>	292	<code>\@glo@assign@sortkey</code>	181, 182, 212
<code>\@colroom</code>	292, 293	<code>\@glo@autosee</code>	92
<code>\@currentlabelname</code>	9, 211	<code>\@glo@autoseehook</code>	92
<code>\@curroptions</code>	37	<code>\@glo@check@mkidxrangechar</code>	119, 120, 193, 328, 329
<code>\@declaredoptions</code>	37	<code>\@glo@check@sortallowed</code> ..	14–16, 181, 184
<code>\@delimN</code>	222	<code>\@glo@childlist</code>	201
<code>\@delimR</code>	222	<code>\@glo@counter</code>	71, 87, 90
<code>\@disable@onlypremakeg</code>	180	<code>\@glo@counterprefix</code>	185, 190, 192–194, 219, 220, 223
<code>\@disable@premakecs</code>	39	<code>\@glo@default@sorttype</code> ..	13, 182, 204, 205
<code>\@disabled@gl@saddxdycounters</code>	51	<code>\@glo@defaultcounter</code>	90
<code>\@do@addcounter</code>	50	<code>\@glo@desc</code>	69, 86–88, 91
<code>\@do@auxoutstuff</code>	200	<code>\@glo@descaccess</code>	350, 352, 353
<code>\@do@glossentry</code>	215, 347, 348	<code>\@glo@descplural</code>	69, 86, 87
<code>\@do@gls@getcounterprefix</code>	190, 192	<code>\@glo@descpluralaccess</code>	350, 352, 353
<code>\@do@gls@islistofacronyms</code>	19	<code>\@glo@do@sortentries</code>	201
<code>\@do@glssee</code>	92	<code>\@glo@entry</code>	164
<code>\@do@ifinlist</code>	50	<code>\@glo@entryprefix</code>	267
<code>\@do@newglossaryentry</code> ..	229, 243, 244, 246–248, 250, 252, 255, 257, 259, 389–393	<code>\@glo@entryprefixfirst</code>	267
<code>\@do@seeglossary</code>	182, 195	<code>\@glo@entryprefixfirstplural</code> ..	267, 268
<code>\@do@subglossentry</code>	217, 348	<code>\@glo@entryprefixplural</code>	267
<code>\@do@wrglossary</code>	118	<code>\@glo@esclabel</code>	93, 94
<code>\@do@write@glslabels</code>	32	<code>\@glo@etext</code>	104–106
<code>\@do@writeaux@info</code>	196	<code>\@glo@first</code>	70, 87, 90, 91, 255, 392
<code>\@domakeglossaries</code>	31, 32, 179, 181	<code>\@glo@firstaccess</code>	349, 352
<code>\@ehc</code>	292	<code>\@glo@firstplural</code>	70, 87, 90, 393
<code>\@empty</code>	12, 16, 19, 33, 36, 37, 39, 50, 51, 54, 57, 58, 88, 89, 94, 118, 119, 129–133, 147–153, 167, 170, 172, 176–178, 186, 187, 193, 194, 219, 244, 246, 248–251, 253, 254, 256, 257, 259, 329, 331, 333, 370–372	<code>\@glo@firstpluralaccess</code>	349, 352
<code>\@end@fixbraces</code>	194, 195	<code>\@glo@grabfirst</code>	207
<code>\@endfortrue</code>	29, 61, 79, 275	<code>\@glo@label</code>	74, 80, 81, 83–85, 87–92, 98, 162, 163, 267, 268, 323, 352, 353
<code>\@esphack</code>	188	<code>\@glo@list</code>	92
<code>\@expandtwoargs</code>	37	<code>\@glo@long</code>	62, 73, 88, 90
<code>\@firstofone</code>	24, 32	<code>\@glo@longaccess</code>	350, 352, 353
<code>\@firstofthree</code>	115, 128, 129, 131, 133, 147, 149, 150, 152, 370–372	<code>\@glo@longpl</code>	73, 88, 90, 244, 246, 248, 250, 252, 255, 257, 389
<code>\@firstoftwo</code>	27, 28, 78, 79, 115, 131–133, 149, 150, 152, 153	<code>\@glo@longpluralaccess</code>	350, 352, 353
<code>\@for</code>	28, 37, 39, 50, 51, 58, 78, 79, 120, 167, 169, 179–181, 188, 195, 201, 231, 244, 247, 249, 251, 253, 256, 258, 260, 275, 276, 329	<code>\@glo@name</code>	14, 69, 74, 87, 89–91
<code>\@glo@desc</code>	91	<code>\@glo@no@assign@sortkey</code>	181
<code>\@glo@symbol</code>	91	<code>\@glo@nonumberlist</code>	72
<code>\@glo@access</code>	349, 352, 357	<code>\@glo@numfmt</code>	193, 329
		<code>\@glo@parent</code>	16, 71, 87, 89, 90, 94, 202
		<code>\@glo@plural</code>	70, 87, 89, 90, 391
		<code>\@glo@pluralaccess</code>	349, 352
		<code>\@glo@prefix</code>	10, 72, 87, 94, 119, 120, 193, 328, 329
		<code>\@glo@range</code>	193, 328, 329
		<code>\@glo@see</code>	71, 87, 92

\@glo@seeautonumberlist	10, 71	\@glo@name	91
\@glo@short	63, 73, 88, 90, 392	\@glossary@default@style	9, 11, 197, 220, 260
\@glo@shortaccess ..	350, 352, 353, 389–392	\@glossaryentryfield	94
\@glo@shortpl	73, 88, 90, 243,	\@glossarysection	46
	244, 246, 248, 250, 252, 255, 257, 389, 392	\@glossarystyle	197, 198, 210
\@glo@shortpluralaccess	350, 352, 353	\@glossarysubentryfield	94
\@glo@sort ...	14, 16, 23, 24, 69, 87, 90, 93, 94	\@gls	128
\@glo@sortedinsert	202	\@gls@	99, 101, 128, 270, 271
\@glo@sortentries	204, 205	\@gls@@automake@immediate	179
\@glo@sorthandler@case	205	\@gls@@link	116
\@glo@sorthandler@letter	204	\@gls@Hcounter	118, 119
\@glo@sorthandler@nocase	205	\@gls@ReturnAfterFi	223
\@glo@sorthandler@word	204	\@gls@actualchar	94, 123, 125, 171, 332
\@glo@sortinghandler	201, 202	\@gls@addpredefinedattributes .	166, 175
\@glo@sortinglist	201, 202, 205	\@gls@adjustmode	163
\@glo@sorttype	182, 206, 207, 212	\@gls@after	19
\@glo@storeentry	14–16	\@gls@automake	181
\@glo@suffix	119, 120, 193, 329	\@gls@automake@immediate	179
\@glo@symbol	62, 70, 87, 91, 249, 254	\@gls@before	19
\@glo@symbolaccess	349, 352, 392	\@gls@between	276
\@glo@symbolplural	70, 87, 91	\@gls@body	155, 156
\@glo@symbolpluralaccess	349, 352	\@gls@checkactual	121, 174
\@glo@text	69,	\@gls@checkboxar	121, 174
	87, 90, 91, 129–134, 154–156, 250, 268, 391	\@gls@checkedmkidx	120–126, 173–175
\@glo@textaccess	349, 352, 389–391	\@gls@checkescactual	121, 174
\@glo@thislabel	93	\@gls@checkescbar	121, 174
\@glo@thislettergrp	207, 208	\@gls@checkescquote	121, 173–175
\@glo@thisvalue	63, 64	\@gls@checklevel	121, 174
\@glo@tmp	80, 81, 193	\@gls@checkmkidxchars	
\@glo@type	9, 16, 70,		93, 94, 119, 173, 181, 192, 194, 328, 329
	87, 88, 90–92, 163, 164, 180, 185, 186,	\@gls@checkquote	121, 173
	197–202, 205, 206, 210, 211, 229, 244,	\@gls@classI	167, 168
	246–248, 250, 251, 253, 256–260, 274, 276	\@gls@classII	167, 168
\@glo@types	32,	\@gls@codepage	200
	58, 59, 66, 97, 98, 164, 179–181, 265, 323	\@gls@counter	
\@glo@useri	72, 87, 90		114, 117–119, 163, 185, 193, 194, 329
\@glo@useriaccess	350, 352, 353	\@gls@counterwithin	12, 13
\@glo@userii	73, 87, 90	\@gls@ctr	50
\@glo@useriiaccess	350, 352, 353	\@gls@currentlettergroup	206, 208
\@glo@useriii	73, 87, 90	\@gls@debugfalse	5
\@glo@useriiiaccess	350, 352, 353	\@gls@debugtrue	5, 6
\@glo@useriv	73, 87, 90	\@gls@declareoption	
\@glo@userivaccess	351–353		10, 11, 17, 18, 21, 22, 27, 30–32, 35, 36
\@glo@userv	73, 87, 90	\@gls@default	103
\@glo@uservaccess	351–353	\@gls@default@value	
\@glo@uservi	73, 87, 90		62–64, 75, 76, 87, 89–91, 254, 267
\@glo@userviaccess	351–353	\@gls@deffile	77, 78
\@glodesc	91	\@gls@define@glossaryentrycounter .	
\@glolist@	88		13, 38, 211, 213

<code>\@gls@define@glossarysubentrycounter</code>	<code>\@gls@ifinlist</code>	50
..... 38, 212, 213	<code>\@gls@ifnotmeasuring</code>	95
<code>\@gls@defsort</code>	<code>\@gls@igtype</code>	69
..... 14–16, 91	<code>\@gls@increment@currcount</code>	99
<code>\@gls@defsortcount</code>	<code>\@gls@indexdef</code>	36
..... 14–16, 67	<code>\@gls@initnonumberlist</code>	72, 87
<code>\@gls@do@acronymsdef</code>	<code>\@gls@islistofacronyms</code>	19
..... 17, 18, 38, 68	<code>\@gls@keylist</code>	388
<code>\@gls@do@indexdef</code>	<code>\@gls@keymap</code>	72, 78–81, 267, 351
..... 36–38, 68	<code>\@gls@label</code>	182, 185, 190, 192, 193
<code>\@gls@do@numbersdef</code>	<code>\@gls@labelsfile</code>	32
..... 36, 38, 68	<code>\@gls@langmod</code>	176–178
<code>\@gls@do@symbolsdef</code>	<code>\@gls@levelchar</code>	94, 124, 125, 171, 332
..... 35, 36, 68	<code>\@gls@link</code>	116, 129–134, 147–154, 370–372
<code>\@gls@do@symbolssdef</code>	<code>\@gls@link@checkfirsthyper</code>	129–133
..... 38	<code>\@gls@link@label</code>	117, 245, 251
<code>\@gls@doautomake</code>	<code>\@gls@link@nocheckfirsthyper</code>	134, 147–153
..... 33, 179, 181	<code>\@gls@link@opts</code>	117, 245, 251
<code>\@gls@docheckquotedef</code>	<code>\@gls@list</code>	275, 276
..... 173–175	<code>\@gls@listsuffix</code>	50
<code>\@gls@docloadedfalse</code>	<code>\@gls@loadlist</code>	11, 260
..... 4	<code>\@gls@loadlong</code>	10, 11, 260
<code>\@gls@docloadedtrue</code>	<code>\@gls@loadsuper</code>	11, 260
..... 4	<code>\@gls@loadtree</code>	11, 260
<code>\@gls@dodedeflistparser</code>	<code>\@gls@local@increment@currcount</code>	99
..... 180	<code>\@gls@loclist</code>	183, 184, 207, 208
<code>\@gls@doentrycounterdef</code>	<code>\@gls@map</code>	78, 79
..... 38	<code>\@gls@missinglang@warn</code>	21, 42
<code>\@gls@doentrydef</code>	<code>\@gls@missingnumberlist</code>	91
..... 113	<code>\@gls@noaccess</code>	357
<code>\@gls@dolast</code>	<code>\@gls@noexpand@fields</code>	76
..... 195	<code>\@gls@nohyperlist</code>	20, 68, 117
<code>\@gls@donext</code>	<code>\@gls@noidx@do</code>	206
..... 195	<code>\@gls@noidx@getgrouptitle</code>	182
<code>\@gls@donext@def</code>	<code>\@gls@noidx@sanitizesort</code>	23, 184
..... 162	<code>\@gls@noidx@setsanitizesort</code>	26, 184
<code>\@gls@dosubentrycounterdef</code>	<code>\@gls@noidx@loclist@finalsep</code>	183
..... 38	<code>\@gls@noidx@loclist@prev</code>	183, 209
<code>\@gls@dotothiswrite</code>	<code>\@gls@noidx@loclist@sep</code>	183, 209
..... 177, 178	<code>\@gls@noref@warn</code>	181, 206
<code>\@gls@elem</code>	<code>\@gls@numberlink</code>	222, 223
..... 275	<code>\@gls@numbersdef</code>	36
<code>\@gls@enablesavenonumberlist</code>	<code>\@gls@numlist@lastsep</code>	162
..... 78	<code>\@gls@numlist@nextsep</code>	162
<code>\@gls@encapchar</code>	<code>\@gls@numlist@sep</code>	162
..... 123, 124, 171, 193, 194, 329, 332	<code>\@gls@old@chapter</code>	38
<code>\@gls@entry@count</code>	<code>\@gls@oldnewglossaryentryposthook</code>	352
..... 100	<code>\@gls@oldnewglossaryentryprehook</code>	351, 352
<code>\@gls@entry@field</code>		
..... 80, 81, 98, 99, 155–161, 353–355		
<code>\@gls@escbsdq</code>		
..... 121, 172, 333		
<code>\@gls@expand@fields</code>		
..... 75, 76		
<code>\@gls@expandonce</code>		
..... 76		
<code>\@gls@extramakeindexopts</code>		
..... 179		
<code>\@gls@fetchfield</code>		
..... 64		
<code>\@gls@field@link</code>		
..... 82, 134–146, 372–377		
<code>\@gls@fieldaccess@display</code>		
..... 357–360		
<code>\@gls@firsttok</code>		
..... 207		
<code>\@gls@fixbraces</code>		
..... 92		
<code>\@gls@forbidtexext</code>		
..... 66		
<code>\@gls@get@counterprefix</code>		
..... 193, 194		
<code>\@gls@getbody</code>		
..... 155		
<code>\@gls@getcounterprefix</code>		
..... 190, 192		
<code>\@gls@getgrouptitle</code>		
..... 182, 218, 276		
<code>\@gls@glossary</code>		
..... 186, 187		
<code>\@gls@gobbleopt</code>		
..... 65		
<code>\@gls@grptitle</code>		
..... 218, 274, 276		
<code>\@gls@hyp@opt</code>		
..... 82,		
101, 102, 116, 128–153, 225–228, 269–273		
<code>\@gls@hyp@opt@cs</code>		
..... 115		
<code>\@gls@hypergroup</code>		
..... 275		

\@gls@onlypremakeg	39	\@gls@updatechecked	120, 121, 173, 174
\@gls@order	176–178	\@gls@usetranslator	27, 28, 41
\@gls@org@LT@output	292	\@gls@value	75, 158
\@gls@org@glsnoidxdisplayloc	184	\@gls@warnonglossdefined	22, 196
\@gls@org@glssseeformat	184	\@gls@warnontheGLOSSdefined	22, 215
\@gls@override@glossary	35	\@gls@write@entrycounts	100
\@gls@patchtabularx	95	\@gls@writedef	77
\@gls@preglossaryhook	199	\@gls@writeisthook	170, 172
\@gls@prevlevel .	303, 304, 323–326, 341, 342	\@gls@xdy@locationlist	167
\@gls@provide@newglossary	66	\@gls@xdycheckbackslash	120
\@gls@quotechar	122–125, 171–175, 332	\@gls@xdycheckquote	120
\@gls@reference	182, 185	\@gls@xref	194
\@gls@removespaces	223	\@gls@Alphacompositor	44, 54, 330
\@gls@renewglossary	176	\@gls@Hlocref	189, 190, 192
\@gls@replacementtext	356, 357	\@gls@sacronymlists	19, 20, 58, 229–231, 244, 246–251, 253, 256–260, 265
\@gls@rest	155	\@gls@saddkey	81
\@gls@restoreat	78	\@gls@saddstoragekey	80
\@gls@roman	53, 329, 330	\@gls@saddxdyattribute	51
\@gls@sanitized@tmp	120	\@gls@defaultsort	14
\@gls@sanitizedesc	29	\@gls@desc	138
\@gls@sanitizesort	14	\@gls@desc@	138, 139
\@gls@sanitizesymbol	29, 30	\@gls@descplural	139
\@gls@saveentrycounter	118, 163	\@gls@descplural@	139
\@gls@savenonumberlist	72	\@gls@disp	133
\@gls@see@noindex	8, 71	\@gls@entry	32, 97, 98, 100
\@gls@setacrstyle	29, 30, 37, 38	\@gls@entrytitlecase	158
\@gls@setcounter	67	\@gls@first	135
\@gls@setdefault@glslink@opts	117	\@gls@first@	135
\@gls@setsort	14–16, 118, 163	\@gls@firstletter	165, 166
\@gls@setupshortcuts	37, 38	\@gls@firstplural	137
\@gls@sort	208	\@gls@firstplural@	137, 373
\@gls@sort@A	203	\@gls@hypernumber	222
\@gls@sort@B	203	\@gls@isacronymlistfalse	19
\@gls@startswithexpandonce	75	\@gls@isacronymlisttrue	19
\@gls@storenonumberlist	72, 91	\@gls@link	118, 128, 163, 274
\@gls@symbolsdef	35	\@gls@localreset	96, 99
\@gls@this	188	\@gls@localunset	96, 99
\@gls@thisHloc	193	\@gls@locref .	185, 189, 190, 192, 193, 328, 329
\@gls@thisfield	64	\@gls@minrange	166–168, 330
\@gls@thislabel	61, 195, 205	\@gls@name	138
\@gls@thislist	162	\@gls@name@	138
\@gls@thisloc	193	\@gls@navhypertarget	274
\@gls@thisval	79	\@gls@nextpages	198
\@gls@title	46	\@gls@nodesc	87, 88, 91
\@gls@tmp ..	7, 16, 41, 54, 76, 120, 187, 275, 276	\@gls@noname	87, 89, 91
\@gls@tmpb	121–126, 173–175	\@gls@nonextpages	198
\@gls@toc	47, 48	\@gls@numberformat	114, 117, 163, 185, 193, 328, 329
\@gls@type	179, 181, 231, 244, 247, 249, 251, 253, 256, 258, 260, 323		

\@glsopenfile	176, 186	\@istfilename	179
\@glsorder	179	\@makecol	292, 293
\@glspl	131	\@makeglossary	31, 32, 180
\@glspl@	99, 102, 131, 270, 272	\@minus	279, 299, 317
\@glsplural	136	\@mkboth	47
\@glsplural@	136	\@newglossary	65, 66
\@glsprefix@record@hook	270–273	\@newglossaryentry@defcounters	92, 98
\@glsreset	95, 99	\@newglossaryentryposthook	80, 81, 92, 267, 352
\@glssee	92, 195	\@newglossaryentryprehook	80, 81, 86, 88, 267, 351
\@glsshowaccsupp	5, 6, 349	\@nil	19, 92, 119–121, 155, 173, 174, 193, 194, 207, 208, 222, 223, 328, 329
\@glsshowtarget	5, 6, 127	\@nnil	19, 195
\@glsymbol	140	\@no@makeglossaries	31, 32, 180, 182
\@glsymbol@	140	\@no@post@desc	334
\@glsymbolplural	141	\@nopostdesc	198
\@glsymbolplural@	141	\@onelevel@sanitize	7, 23, 53, 78, 94, 120, 165, 166, 170, 194, 196, 207, 330, 331
\@glstarget	128, 215, 275	\@onlypreamble	67, 77, 86, 100, 103, 181, 184
\@glstext	134	\@onlypremakeg	43, 44, 50, 51, 55, 67, 172
\@glstext@	134	\@org@glossaryentrynumbers	198, 199
\@glsunset	96, 99	\@org@gls@assign@descplural	244, 252, 253, 255, 257, 389, 392, 393
\@glsuseri	142	\@org@gls@assign@firstpl	244, 246, 248, 250, 252, 255, 257, 389–393
\@glsuseri@	142	\@org@gls@assign@plural	244, 246, 248, 250, 252, 255, 257, 389–393
\@glsuserii	142, 143	\@org@gls@assign@symbolplural	244, 246, 248, 250, 255, 257, 389–391, 393
\@glsuserii@	143	\@org@glsnumberformat	162
\@glsuseriii	143	\@org@newglossaryentryprehook	86
\@glsuseriii@	143	\@outputpage	292, 293
\@glsuseriv	144	\@p@ glossarysection	46
\@glsuseriv@	144	\@pgls	269
\@glsuserv	145	\@pgls@	269, 270
\@glsuserv@	145	\@pglspl	270
\@glsuservi	146	\@pglspl@	270
\@glsuservi@	146	\@plus	279, 299, 317
\@glswidestname	323–325, 341	\@print@glossary	197
\@glswritefiles	33	\@print@noidx@glossary	197
\@glxtr@doaccsupp	347	\@printgloss@setsort	181, 182, 198
\@glxtr@record	269	\@printglossary	197
\@gobble	5, 14–16, 31, 32, 78, 95, 120, 165, 169, 180, 181, 327, 331, 332	\@roman	53, 329
\@gobblethree	5	\@secondofthree	115, 128, 130, 132, 147, 149, 151, 153, 370
\@idxitem	317	\@secondoftwo	24, 27, 28, 41, 78, 79, 127–130, 133, 147, 148, 150–152, 370–372, 396
\@ifclassloaded	4, 12, 46	\@set@glo@numformat	193, 329
\@ifl@t@r	25		
\@ifnextchar	67, 115		
\@ifpackageloaded	4, 9, 27, 28, 41, 57, 95, 161, 172, 347		
\@ifstar	66, 80, 81, 115, 165, 166, 224		
\@ifundefined	40, 275, 282, 293, 304, 311, 324, 325, 341		
\@ignored@glossaries	32, 68, 69		
\@input@	199		

<code>\@sglsaddkey</code>	81	<code>\accsuppglossaryentryfield</code>	347
<code>\@sglsaddstoragekey</code>	80	<code>\accsuppglossarysubentryfield</code>	348
<code>\@tabacckludge</code>	24	<code>\acrfootnote</code>	245, 251
<code>\@text@composite@x</code>	24	<code>\Acrfull</code>	242
<code>\@thirdofthree</code>		<code>\acrfull</code>	242
.....	115, 130, 133, 148, 150, 152, 153, 371	<code>\ACRfullfmt</code>	227, 230, 238, 240, 384, 386
<code>\@this@attr</code>	169	<code>\Acrfullfmt</code>	226, 230, 238, 240, 384, 386
<code>\@this@childlabel</code>	201, 202	<code>\acrfullfmt</code>	226, 229, 238, 239, 384, 386
<code>\@this@counter</code>	51	<code>\acrfullformat</code>	161, 226, 243, 244, 259
<code>\@this@ctr</code>	169	<code>\Acrfullpl</code>	243
<code>\@this@key</code>	79	<code>\acrfullpl</code>	242
<code>\@this@label</code>	201	<code>\ACRfullplfmt</code> ...	228, 230, 238, 240, 384, 386
<code>\@thiscs</code>	39	<code>\Acrfullplfmt</code> ...	228, 230, 238, 240, 384, 386
<code>\@tmp</code>	53, 330	<code>\acrfullplfmt</code> ...	227, 230, 238, 240, 384, 386
<code>\@use@option</code>	37	<code>\acrlinkfootnote</code>	245
<code>\@warn@nomakeglossaries</code>	179, 200	<code>\acrlinkfullformat</code>	226–228
<code>\@wrglossary@pageformat</code>	189	<code>\Acrlong</code>	242
<code>\@wrglossarynumberhook</code>	189, 192	<code>\acrlong</code>	242
<code>\@xdy@main@language</code>	30, 176, 178, 200	<code>\Acrlongpl</code>	242
<code>\@xdy@attributelist</code>	51, 169	<code>\acrlongpl</code>	242
<code>\@xdy@attributes</code>	51, 167, 327, 329	<code>\acrnameformat</code>	250, 390
<code>\@xdy@counters</code>	50, 51, 169	<code>\acronymentry</code>	229, 232,
<code>\@xdy@crossrefhook</code>	168	233, 235, 236, 238–241, 379–382, 385–388
<code>\@xdy@language</code>	200	<code>\acronymfont</code>	110,
<code>\@xdy@lettergroups</code>	58, 170, 332	147–150, 155, 161, 228, 230, 232–236,
<code>\@xdy@locationclassorder</code>	55, 168, 331	238–241, 245–247, 249, 251–256, 367,
<code>\@xdy@locref</code>	51, 170, 327, 331	368, 370–372, 380–382, 384–388, 390–392
<code>\@xdy@numbergrouporder</code>	57, 166	<code>\acronymname</code>	17, 18, 42
<code>\@xdy@requiredstyles</code>	56, 167, 329	<code>\acronymsort</code>	229, 232, 233,
<code>\@xdy@sortrules</code>	56, 170, 332	235, 236, 238, 239, 241, 380–382, 385–388
<code>\@xdystyle</code>	167, 329	<code>\acronymtype</code>	17, 18, 229, 230, 243, 244, 246–
<code>\@xdy@useralphabets</code>	52, 167, 329	248, 250, 252, 253, 255–257, 259, 389–392
<code>\@xdy@userlocationdefs</code>	54, 55, 168, 328, 330	<code>\acrpluralsuffix</code>	
<code>\@xdy@userlocationnames</code>	55, 328	229, 232–234, 238–241, 244–248, 250–
<code>\@xfor@nextelement</code>	195	253, 255–257, 259, 380, 385–387, 389–393
<code>\@</code>	93, 120, 165, 171,	<code>\Acrshort</code>	242
.....	172, 222, 223, 332, 333, 335–337, 345, 346	<code>\acrshort</code>	242
<code>\{</code>	78, 165, 171, 172, 327, 332, 333	<code>\Acrshortpl</code>	242
<code>\}</code>	78, 165, 171, 172, 327, 333	<code>\acrshortpl</code>	242
<code>\^</code>	24	<code>\add@accent@</code>	24
<code>\‘</code>	24	<code>\addcontentsline</code>	49
<code>\ </code>	121, 123, 174	<code>\addglossarytocaptions</code>	41
<code>\~</code>	24	<code>\addtolength</code>	325, 341
		<code>\advance</code>	15, 16, 89, 118, 292
		<code>\AE</code>	24
		<code>\ae</code>	24
<code>\a</code>	24	<code>amsgen package</code>	4, 114
<code>\AA</code>	24	<code>amsmath package</code>	95
<code>\aa</code>	24	<code>\andname</code>	195
<code>accsupp package</code>	347, 348		

A

\AnyTrackedLanguages	41, 396	\csdef	22, 80–82, 92, 93, 98, 99, 201, 202, 221, 231, 333
\appto	20, 25, 72, 80, 81, 267, 351	\csedef	100, 189
array package	289, 293, 311	\csgdef	45, 65, 68, 99, 100, 196, 210
article class	193	\cslet	72, 86, 93, 205
\AtBeginDocument	18, 57, 77, 95, 164, 182	\csname	13–16, 24, 37, 39, 41, 42, 47, 48, 51, 53, 54, 57, 58, 61, 66–68, 74, 75, 80–82, 84, 85, 88– 94, 96, 97, 113, 117–119, 129–134, 147– 154, 162, 163, 167–169, 173–176, 182, 185–187, 189, 193, 194, 197–200, 202, 210, 215, 217, 220, 221, 224, 261–268, 275, 276, 323–325, 327–329, 341, 347, 348, 352, 353, 356, 360, 370–372, 394, 395
\AtEndDocument	32, 33, 77, 100, 181, 185, 200, 275	\csshow	265
B		\csuse	42, 45, 65, 75, 81, 82, 113, 177, 178, 202, 204, 206, 208, 209, 211, 212, 220, 231, 268, 334–346
\b	24	\csxdef	91, 100
babel package	27, 39, 41, 56	\currentglossary	12, 13, 45, 198
\begin	169, 206, 279, 283–288, 291–317, 331	\currentglsentry	13, 213
\BeginAccSupp	348	\CurrentOption	37, 267, 347
\begingroup	5, 7, 187, 191, 223	\CurrentTrackedLanguage	42, 396, 397
\bfseries	284–287, 289, 290, 294–296, 298, 306–311, 313–317	\CurrentTrackedTag	42, 396, 397
\bgroup	24, 86, 162, 198, 201	\CustomAcronymFields	259
bib2gls	24, 61, 190	\CustomNewAcronymDef	260
booktabs package	288–291	D	
\boolean	258	\d	24
\boolfalse	33, 34	datatool package	202
\booltrue	34	datatool-base package	4
\bottomrule	289, 290	\day	167, 171, 329, 332
\box	292	\DeclareAcronymList	17, 18, 20, 229, 230, 244, 246, 248, 251, 253, 256, 258, 260
C		\DeclareListParser	180
\c	24	\DeclareOption	9, 267, 347
\c@equation	118	\DeclareOptionX	9
\c@glossaryentry	12	\DeclareRobustCommand	42, 253
\c@glossarysubentry	13	\def	9, 10, 12–16, 18, 19, 23–25, 30, 31, 34, 35, 37–39, 42, 43, 46, 49, 50, 52–57, 61, 65– 67, 69–73, 76, 84, 86–91, 93, 95, 99–103, 113, 114, 117–127, 129–154, 162, 163, 166, 171–178, 180, 182, 183, 186, 189, 190, 192–195, 197, 198, 201, 205–207, 209, 210, 212, 219–229, 244, 246, 248– 250, 252–257, 259, 267, 270–273, 276– 278, 303, 304, 323–326, 328, 329, 332, 334, 341, 342, 347–352, 370–377, 389–393
\c@page	188, 189, 192	\def@gls@xdycheckbackslash	126, 127
\catcode	78	\DefaultNewAcronymDef	245
\cGls	101		
\cgl	101		
\cGlsformat	99		
\cglformat	99		
\cGlspl	102		
\cglspl	102		
\cGlsplformat	100		
\cglsplformat	99		
\char	219		
classicthesis package	9		
\cleardoublepage	48		
\clearpage	48		
\closeout	32, 77, 170, 172, 176, 186		
\compatglossarystyle	334–346		
\compatibleglossentry	217		
\compatiblesubglossentry	217		
\copy	292, 293		
\count@	207		

<code>\defglentryfmt</code>	67, 68, 113, 231, 243, 245, 247, 249, 251, 254, 256, 259	190, 192, 193, 196, 200–203, 207, 213, 219, 223, 224, 243, 246, 247, 250, 252, 255, 257, 274, 327, 328, 330, 332, 388–392	
<code>\define@boolkey</code>	8, 10, 12, 13, 17, 25, 26, 29, 30, 33, 34, 114, 212	<code>\egroup</code>	24, 86, 162, 199, 202
<code>\define@choicekey</code>	5, 7–9, 13, 26, 28, 30, 33, 35, 72, 211, 212	<code>\else</code> ...	6, 11–13, 16–19, 21, 23, 25, 26, 33, 35, 37–39, 43, 44, 46–53, 55–58, 72, 74, 89, 90, 93–95, 99, 100, 116–118, 120– 127, 129–134, 155, 156, 165–167, 170, 172–178, 186–190, 192–195, 198, 207, 212, 214, 219, 222, 223, 233, 247, 248, 251, 253, 254, 256, 258, 260, 275, 280, 283, 285, 286, 289, 290, 292, 294, 296, 297, 305, 307, 309, 312, 314, 316, 319, 320, 322, 324, 325, 328–334, 339–342, 357
<code>\define@key</code>	9, 13, 20, 26, 30, 31, 69– 73, 80, 81, 114, 163, 210, 212, 267, 349–351	<code>\emph</code>	195, 224
<code>\DefineAcronymSynonyms</code>	37, 243	<code>\empty</code>	223, 347
<code>\delimN</code>	169, 180, 209, 222, 331	<code>\encodingdefault</code>	24
<code>\delimR</code>	169, 222, 331	<code>\end</code>	169, 206, 279, 283–288, 291–317, 331
<code>\DescriptionDUANewAcronymDef</code>	249	<code>\end@doifinlist</code>	50
<code>\DescriptionFootnoteNewAcronymDef</code> .	247	<code>\end@getprefix</code>	193, 194
<code>\descriptionname</code>	42, 284–287, 289, 290, 294–296, 298, 306–311, 313–317	<code>\end@glscislistofacronyms</code>	19
<code>\DescriptionNewAcronymDef</code>	251	<code>\EndAccSupp</code>	348
<code>\DH</code>	25	<code>\endcsname</code>	13–16, 24, 37, 39, 41, 42, 47, 48, 51, 53, 54, 57, 58, 61, 66–68, 74, 75, 80–82, 84, 85, 88– 94, 96, 97, 113, 117–119, 129–134, 147– 154, 162, 163, 167–169, 173–176, 182, 185–187, 189, 193, 194, 197–200, 202, 210, 215, 217, 220, 221, 224, 261–268, 275, 276, 323–325, 327–329, 341, 347, 348, 352, 353, 356, 360, 370–372, 394, 395
<code>\dh</code>	25	<code>\endfoot</code> .	283–285, 287, 289, 290, 294–296, 298
<code>\dimen@</code>	233, 292, 323	<code>\endgroup</code>	5, 7, 188, 192, 223
<code>\disable@keys</code>	37	<code>\endhead</code> .	283–285, 287, 289, 290, 294–296, 298
<code>\do</code>	28, 37, 39, 50, 51, 58, 78, 79, 120, 162, 167, 169, 179–181, 188, 195, 201, 231, 244, 247, 249, 251, 253, 256, 258, 260, 275, 276, 329	<code>\endtheglossary</code>	5
<code>\do@glo@storeentry</code>	14–16, 92	<code>\entryname</code>	42, 284–287, 289, 290, 294–296, 298, 306–311, 313–317
<code>\do@glsc@link@checkfirsthyper</code>	116, 117, 129–134, 147–153, 370–372	<code>\equal</code>	26, 38, 48, 118, 180, 219, 275
<code>\do@glsc@xdycheckbackslash</code>	120	equation (counter)	118, 119
<code>\do@glsc@disablehyperinlist</code>	117	etoolbox package	4
<code>\do@glshaschildren</code>	61	<code>\expandafter</code>	7, 14–16, 23, 24, 37, 39, 41, 51, 53, 54, 56–59, 61, 66, 67, 69, 74, 75, 78–82, 84, 85, 88–92, 94–97, 113, 117–126, 155, 156, 162, 165, 169, 173, 174, 176, 186– 189, 192, 193, 195, 198, 202, 207, 208, 215, 217, 221, 223, 224, 245, 251, 261– 268, 275, 276, 323, 327–329, 331, 332, 347, 348, 352, 353, 356, 357, 388, 394, 395
doc package	4, 5, 17		
<code>\doifglossarynoexistsordo</code>	66		
<code>\dtl@ifsingle</code>	219		
<code>\dtl@insertinto</code>	202		
<code>\dtl@sortresult</code>	203		
<code>\dtlcompare</code>	203		
<code>\dtlicompare</code>	203		
<code>\DTLifinlist</code>	69, 117		
<code>\DTLifint</code>	219		
<code>\dtlletterindexcompare</code>	203		
<code>\DTLsubstituteall</code>	120		
<code>\dtlwordindexcompare</code>	203		
<code>\DUANewAcronymDef</code>	258		
E			
<code>\eappto</code>	68, 93, 189		
<code>\edef</code>	16, 19, 39, 41, 49–56, 58, 61, 66, 68, 69, 75, 78, 79, 83– 85, 87, 88, 93, 94, 113, 117–126, 162– 165, 171–175, 177, 178, 180, 182, 186,		

<code>\expandonce</code>	75, 76, 120, 173–175, 189, 203, 215, 217, 229, 243, 244, 246, 248, 250, 252, 255, 257, 347, 348
F	
<code>\fi</code>	5, 6, 8, 9, 11–19, 21, 23, 25, 26, 28, 33, 35–39, 42–44, 47–58, 67, 72, 74, 88–91, 93–95, 99, 100, 116–127, 129–134, 156, 164– 167, 170, 172–181, 186–190, 192–196, 198, 200, 207, 211–214, 220, 222, 223, 233, 243, 244, 247, 248, 251, 253, 254, 256, 258, 260, 266, 275, 280, 283, 285, 286, 289, 290, 292–294, 296, 297, 305, 307, 309, 312, 314, 316, 319, 320, 322– 325, 327–331, 333, 334, 339–342, 347, 357
file types	
<code>.aux</code>	199
<code>.glo</code>	93
<code>.ist</code>	164, 165, 175
<code>.toc</code>	49
<code>.xdy</code>	43
<code>glo</code>	266
<code>\firstacronymfont</code>	112, 232, 233, 239, 245, 249, 251, 254, 369, 370, 379, 380, 385, 386
<code>\fmtversion</code>	25
<code>\footnote</code>	239, 245, 385, 386
<code>\FootnoteNewAcronymDef</code>	253
<code>\footnotesize</code>	6
<code>\forall glossaries</code>	59, 185, 197, 323
<code>\forall gloss entries</code>	32, 97, 98, 100, 164
<code>\forall Each Tracked Dialect</code>	41, 396, 397
<code>\forall gloss entries</code>	59, 61, 93, 205, 323
<code>\forall list cs loop</code>	201, 206
<code>\forall list loop</code>	183, 184, 209
G	
<code>garamondx</code> package	225
<code>\gdef</code>	15, 51, 66, 84, 89, 90, 187, 212, 275
<code>\Genacrfullformat</code>	111, 230, 232, 233, 239, 369, 379, 380, 385
<code>\genacrfullformat</code>	111, 112, 230, 232, 233, 239, 369, 379, 380, 385
<code>\GenericAcronymFields</code>	229, 232, 233, 235, 236, 238, 239, 241, 379–382, 384, 385, 387, 388
<code>\Genplacrfullformat</code>	111, 230, 232, 233, 239, 369, 379, 380, 385
<code>\genplacrfullformat</code>	111, 112, 230, 232, 233, 239, 368, 369, 379, 380, 385
<code>\glo@desc</code>	334
<code>\glo@do@compare</code>	203
<code>\glo@grabfirst</code>	208
<code>\glo@label</code>	61, 93
<code>\glo@list</code>	93
<code>\glo@name</code>	215
<code>\glo@parent</code>	61
<code>\glo@type</code>	93
<code>\glo@value</code>	78
<code>\global</code>	15, 16, 74, 77, 86, 91, 96, 97, 187, 199, 207, 208, 213, 292, 293
<code>\glo link prefix</code>	118, 163, 215
<code>\glosortentrieswarning</code>	21, 201
<code>glossaries</code> package	35, 37, 56, 57, 166, 260, 267, 279, 327, 347
<code>glossaries-accsupp</code> package	93, 347, 348
<code>glossaries-extra</code> package	77, 168, 176, 269, 347, 356
<code>\GlossariesWarning</code> ...	5, 6, 8, 21, 25, 26, 31, 32, 34, 35, 46, 49, 60, 64, 71, 74, 101, 102, 113, 115, 161, 177, 178, 181, 183– 185, 187, 194, 197, 199, 217, 220, 327, 347
<code>\GlossariesWarningNoLine</code>	5, 6, 21, 180, 182, 186, 200, 275
<code>\glossary</code>	34, 35, 328, 329
<code>glossary</code> package	1, 35, 224
<code>glossary</code> styles:	
<code>altlist</code>	280, 281, 335
<code>altlistgroup</code>	281, 335
<code>altlisthypergroup</code>	281, 335
<code>altlong4col</code>	287, 288, 296, 337
<code>altlong4col-booktabs</code>	290, 292
<code>altlong4colborder</code>	288, 337
<code>altlong4colheader</code>	288, 290, 337
<code>altlong4colheaderborder</code>	288, 337
<code>altlongragged4col</code> ...	291, 296–298, 338
<code>altlongragged4col-booktabs</code>	291
<code>altlongragged4colborder</code>	298, 338
<code>altlongragged4colheader</code>	297, 338
<code>altlongragged4colheaderborder</code>	298, 339
<code>altsuper4col</code>	310, 315, 346
<code>altsuper4colborder</code>	310, 346
<code>altsuper4colheader</code>	310, 346
<code>altsuper4colheaderborder</code> ...	310, 346
<code>altsuperragged4col</code>	315, 316, 344
<code>altsuperragged4colborder</code> ...	316, 344
<code>altsuperragged4colheader</code> ...	316, 344
<code>altsuperragged4colheaderborder</code> .	316, 344

alttree	303, 318, 323, 325, 341
alttreegroup	326, 342
alttreehypergroup	326, 342
index	9, 299, 317–320, 339
indexgroup	319, 339
indexhypergroup	319, 339
inline	334
list	9, 11, 279–281, 334
listdotted	281, 282, 335
listgroup	280, 334
listhypergroup	280, 335
long	283, 284, 289, 293, 335, 337
long-booktabs	289, 291
long3col	284, 285, 289, 336
long3col-booktabs	289, 291
long3colborder	285, 336
long3colheader	285, 289, 336
long3colheaderborder	285, 336
long4col	286, 287, 290, 336
long4col-booktabs	290
long4colborder	287, 337
long4colheader	286, 290, 337
long4colheaderborder	287, 337
longborder	283, 336
longheader	283, 289, 336
longheaderborder	284, 336
longragged	291, 293–295
longragged-booktabs	291
longragged3col	291, 295, 296, 338
longragged3col-booktabs	291
longragged3colborder	296, 338
longragged3colheader	296, 338
longragged3colheaderborder	296, 338
longraggedborder	294, 337
longraggedheader	294, 338
longraggedheaderborder	295, 338
mcolalttree	303, 343
mcolalttreegroup	303, 343
mcolalttreehypergroup	303, 304, 343
mcolindex	299, 342
mcolindexgroup	299, 342
mcolindexhypergroup	299, 300, 342
mcoltree	300, 342
mcoltreegroup	342
mcoltreehypergroup	301, 342
mcoltreenoname	302, 343
mcoltreenonamegroup	302, 343
mcoltreenonamehypergroup	302, 343
sublistdotted	335
super	305, 306, 313, 345
super3col	306–308, 345
super3colborder	307, 345
super3colheader	307, 345
super3colheaderborder	308, 345
super4col	308–310, 346
super4colborder	309, 346
super4colheader	309, 346
super4colheaderborder	309, 346
superborder	305, 345
superheader	306, 345
superheaderborder	306, 345
superragged	311, 313, 343
superragged3col	313–315, 344
superragged3colborder	314, 344
superragged3colheader	314, 344
superragged3colheaderborder	315, 344
superraggedborder	312, 343
superraggedheader	313, 343
superraggedheaderborder	313, 344
tree	300, 320, 321, 323, 339
treegroup	301, 321, 340
treehypergroup	321, 340
treenoname	301, 318, 321, 322, 340
treenonamegroup	322, 341
treenonamehypergroup	322, 341
glossary-hypernav package	165
glossary-list package	9, 11, 279
glossary-long package	10, 282, 296, 304, 305
glossary-longragged package	293
glossary-mcols package	298
glossary-super package	11, 282, 304, 311, 315
glossary-superragged package	311
glossary-tree package	11, 317
\glossaryentry	193, 194, 329
glossaryentry (counter)	12, 213, 214
\glossaryentryfield	215, 334–341, 343–346, 379
\glossaryentrynumbers	10, 169, 198, 199, 208, 212, 213, 331
\glossaryheader	169, 206, 277, 279–281, 283–287, 289, 290, 293–299, 301–303, 305, 306, 308, 312, 313, 315, 318–323, 326, 331
\glossarymark	46
\glossaryname	17, 41, 42
\glossarypostamble	169, 206, 331
\glossarypreamble	168, 206, 331

<code>\glossarysection</code>	168, 206, 331	<code>\gls@disablepagerefexpansion</code> ..	187, 192
<code>\glossarysubentry (counter)</code>	13, 213, 214	<code>\gls@do@addxdyattribute</code>	51
<code>\glossarysubentryfield</code>		<code>\gls@doclearpage</code>	49
.....	217, 334–341, 343–346, 379	<code>\gls@dosubst</code>	120
<code>\glossarytitle</code> ..	168, 197, 198, 206, 210, 331	<code>\gls@dotocitle</code>	198, 210
<code>\glossarytoctitle</code>	9, 17, 18,	<code>\gls@end@sanitizesort</code>	23, 24
36, 39, 42, 46, 168, 197, 206, 210, 211, 331		<code>\gls@endcheck</code>	75, 76
<code>\glossentry</code>	93, 199, 208, 217,	<code>\gls@glossary</code>	34, 35, 193, 194
277, 279–284, 286, 294, 295, 297, 305,		<code>\gls@gobbleopt</code>	67
307, 308, 312, 313, 315, 318, 320, 321, 324		<code>\gls@grplabel</code>	274
<code>\Glossentrydesc</code>	378	<code>\gls@hypergroup prerun</code>	275
<code>\glossentrydesc</code>		<code>\gls@ifnotmeasuring</code>	95, 96
. 277–284, 286, 294, 295, 297, 305, 307,		<code>\gls@inlinepostchild</code>	277, 278, 334
308, 312–315, 318–320, 322, 324, 325, 378		<code>\gls@inlinesep</code>	277, 334
<code>\glossentryname</code>		<code>\gls@inlinesubsep</code>	277, 278, 334
. 277–284, 286, 294, 295, 297, 305, 307,		<code>\gls@islistofacronyms</code>	19
308, 312, 313, 315, 318–321, 324, 325, 378		<code>\gls@istfilebase</code>	43, 177, 178
<code>\Glossentrysymbol</code>	378	<code>\gls@label</code>	224
<code>\glossentrysymbol</code>	277, 278, 286,	<code>\gls@level</code>	89, 90, 208
297, 308, 315, 318–320, 322, 324, 325, 378		<code>\gls@noidxglossary</code>	182
<code>\Gls</code>	101, 224, 243	<code>\gls@nonumberlist@nr</code>	72
<code>\gls</code>	34, 101, 181, 214, 224, 243	<code>\gls@nonumberlist@val</code>	72
<code>\gls@accessibility</code>	349	<code>\gls@nosetquote</code>	87, 171, 172, 175
<code>\gls@accsupp@engine</code>	348	<code>\gls@number</code>	192
<code>\gls@Alphpage</code>	188, 192	<code>\gls@numberedsection@nr</code>	9, 211
<code>\gls@alphpage</code>	188, 192	<code>\gls@numberedsection@val</code>	9, 211
<code>\gls@arabicpage</code>	188, 192	<code>\gls@numberpage</code>	188, 192
<code>\gls@assign@desc</code>	86, 91	<code>\gls@org@glossaryentryfield</code>	199
<code>\gls@assign@descplural</code>		<code>\gls@org@glossarysubentryfield</code>	199
.... 244, 252, 253, 255, 257, 389, 392, 393		<code>\gls@org@insert</code>	249, 251, 254
<code>\gls@assign@field</code>		<code>\gls@orgAlph</code>	191, 192
..... 76, 80, 81, 86, 88, 90, 91, 267, 268		<code>\gls@orgalph</code>	191, 192
<code>\gls@assign@firstpl</code>		<code>\gls@orgarabic</code>	191, 192
244, 246, 248, 250, 252, 255, 257, 389–393		<code>\gls@orgnumber</code>	191, 192
<code>\gls@assign@plural</code>		<code>\gls@orgRoman</code>	191, 192
244, 246, 248, 250, 252, 255, 257, 389–393		<code>\gls@orgromannumeral</code>	191, 192
<code>\gls@assign@symbolplural</code>		<code>\gls@orgthe</code>	191, 192
244, 246, 248, 250, 255, 257, 389–391, 393		<code>\gls@original@glossary</code>	35
<code>\gls@automake@nr</code>	33, 179	<code>\gls@original@makeglossary</code>	35
<code>\gls@automake@val</code>	33	<code>\gls@protected@pagefmts</code>	120, 188, 189
<code>\gls@begindocdefs</code>	77	<code>\gls@Romanpage</code>	188, 192
<code>\gls@checkisacronymlist</code>	116	<code>\gls@romanpage</code>	188, 192
<code>\gls@checkseeallowed</code>	71, 77, 180, 182	<code>\gls@save@numberlist</code>	10
<code>\gls@checkseeallowed@preambleonly</code> ..	77	<code>\gls@seen@index@nr</code>	7
<code>\gls@codepage</code>	57, 177, 178, 200	<code>\gls@seen@index@val</code>	7
<code>\gls@debug@nr</code>	5, 35	<code>\gls@set@xr@key</code>	71
<code>\gls@debug@val</code>	5, 35	<code>\gls@suffixF</code>	44, 170, 172, 331, 333
<code>\gls@defdocnewglossaryentry</code>	78, 98	<code>\gls@suffixFF</code>	44, 170, 172, 331, 333
<code>\gls@defglossaryentry</code>	76–78, 86	<code>\gls@text</code>	112

<code>\gls@the</code>	192	<code>\glsdefaultshortaccess</code>	388–392
<code>\gls@thissty</code>	28	<code>\glsdefaultttype</code>	17, 45, 57, 58, 65, 66, 88, 103, 113, 186, 196, 197
<code>\gls@tmp</code>	186, 253, 254	<code>\glsdefmain</code>	17, 68
<code>\gls@tmplen</code>	127, 323–325, 341, 342	<code>\glsdescriptionaccessdisplay</code>	362–364, 374, 378, 379
<code>\gls@tr@set@acronym@toctitle</code>	18	<code>\glsdescriptionpluralaccessdisplay</code>	361, 362, 374, 375
<code>\gls@tr@set@main@toctitle</code>	17	<code>\glsdescwidth</code>	283– 285, 287, 288, 291–298, 305–308, 310–317
<code>\gls@tr@set@numbers@toctitle</code>	36	<code>\glsdetoklabel</code>	59–64, 72, 78, 83–87, 93, 96–100, 117, 154, 155, 162, 163, 182–184, 190, 192, 199, 202, 203, 207, 208, 210, 213–215, 261–265, 268, 269, 323, 347, 348, 357–360, 394, 395
<code>\gls@tr@set@symbols@toctitle</code>	36	<code>\glsdisplay</code>	104, 113
<code>\gls@translate@nr</code>	28	<code>\glsdisplayfirst</code>	104, 113
<code>\gls@translate@val</code>	28	<code>\glsdisplaynumberlist</code>	183
<code>\gls@wrglossary</code>	187	<code>\glsdohyperlink</code>	127, 128
<code>\gls@xdystring</code>	120	<code>\glsdohypertarget</code>	128
<code>\gls@xindy@glsnumbersfalse</code>	31	<code>\glsdoifexists</code>	61–63, 83–85, 95, 96, 129–134, 147–153, 158, 161–163, 183, 184, 270–273, 370–372, 378
<code>\gls@xindy@glsnumberstrue</code>	30	<code>\glsdoifexistsordo</code>	116, 154
<code>\gls@xr@key</code>	7, 8, 71	<code>\glsdoifexistsorwarn</code>	210, 215, 216
<code>\glsaccessibility</code>	356, 357	<code>\glsdoifnoexists</code>	76, 86
<code>\glsacccsupp</code>	356, 357	<code>\glsdonohyperlink</code>	118, 127, 128
<code>\glsacronymtrue</code>	18	<code>\glsdosanitizesort</code>	14
<code>\glsacrpluralsuffix</code>	40, 225, 234, 238, 239, 241, 245	<code>\glsentryaccess</code>	357
<code>\glsacrshortcutsfalse</code>	37	<code>\glsentrycounter</code>	220, 223
<code>\glsacrshortcutstrue</code>	37	<code>\glsentrycounterfalse</code>	12
<code>\glsacspace</code>	232, 233, 235	<code>\glsentrycounterlabel</code>	214
<code>\glsadd</code>	34, 164	<code>\GlsEntryCounterLabelPrefix</code> ...	213, 214
<code>\glsadd options</code> counter	163	<code>\glsentrycountertrue</code>	13
format	163, 221	<code>\glsentrycurrcount</code>	98, 100
<code>\glsaddall options</code> types	163, 164	<code>\Glsentrydesc</code>	139, 216, 374, 378
<code>\GlsAddXdyAttribute</code>	50, 52, 327, 328	<code>\glsentrydesc</code>	106, 107, 139, 216, 362–364, 374, 378
<code>\GlsAddXdyCounters</code>	50, 51, 67	<code>\glsentrydescaccess</code>	358
<code>\glsautomakefalse</code>	33, 179	<code>\Glsentrydescplural</code>	140, 374
<code>\glsautomaketrue</code>	33	<code>\glsentrydescplural</code>	104, 105, 139, 140, 361, 362, 374, 375
<code>\glsautoprefix</code>	9, 211	<code>\glsentrydescpluralaccess</code>	359
<code>\glscapscase</code>	104, 105, 108–111, 129–133, 147– 153, 237, 249, 254, 360, 362, 365–372, 383	<code>\Glsentryfirst</code> 102, 106, 109, 136, 363, 366, 373	
<code>\glscategory</code>	356	<code>\glsentryfirst</code>	101, 106, 107, 109, 135, 136, 363, 364, 366, 367, 373
<code>\glsclearpage</code>	48	<code>\glsentryfirstaccess</code>	358
<code>\glsclosebrace</code>	54, 55, 169, 170, 331, 332	<code>\Glsentryfirstplural</code>	103, 105, 108, 137, 361, 365, 373
<code>\glscompositor</code>	44, 54, 172, 330, 333		
<code>\glscounter</code>	20, 38, 50, 67, 90, 118, 327		
<code>\glscurrententrylabel</code>	196, 199		
<code>\glscurrentfieldvalue</code>	63, 64		
<code>\glscustomtext</code>	104, 107, 109, 110, 112, 129– 133, 147–154, 237, 238, 245, 249, 251, 252, 254, 360, 364, 367, 369–372, 382, 384		
<code>\GlsDeclareNoHyperList</code>	20		

<code>\glsentryfirstplural</code>	102, 104, 105, 108, 137, 361, 362, 365, 366, 373	<code>\glsentryshortpl</code>	110, 112, 149, 150, 161, 232, 233, 238– 240, 259, 367, 368, 370, 379, 380, 384–387
<code>\glsentryfirstpluralaccess</code>	358	<code>\glsentryshortpluralaccess</code>	359
<code>\glsentryfmt</code>	67, 68	<code>\Glsentrysymbol</code>	140, 216, 375, 378
<code>\Glsentryfull</code>	230, 238, 240, 384, 386	<code>\glsentrysymbol</code>	106, 107, 140, 141, 216, 245, 249, 254, 362–364, 375, 378
<code>\glsentryfull</code>	230, 238, 240, 384, 386	<code>\glsentrysymbolaccess</code>	358
<code>\Glsentryfullpl</code>	230, 238, 240, 384, 387	<code>\Glsentrysymbolplural</code>	141, 375
<code>\glsentryfullpl</code>	230, 238, 240, 384, 386	<code>\glsentrysymbolplural</code>	104, 105, 141, 142, 245, 249, 254, 361, 362, 375
<code>\glsentryitem</code>	277, 279–284, 286, 294, 295, 297, 305, 307, 308, 312, 313, 315, 318, 320, 321, 324, 334–341, 343–346	<code>\glsentrysymbolpluralaccess</code>	358
<code>\Glsentrylong</code>	102, 151, 155, 161, 232, 237, 238, 369, 372, 379, 383, 384	<code>\Glsentrytext</code> ...	106, 109, 135, 363, 366, 372
<code>\glsentrylong</code>	101, 112, 150, 152, 155, 161, 232, 233, 235–241, 252, 369, 371, 372, 379–388	<code>\glsentrytext</code>	106, 107, 109, 134, 135, 163, 196, 362, 363, 366, 367, 372
<code>\glsentrylongaccess</code>	359	<code>\glsentrytextaccess</code>	358
<code>\Glsentrylongpl</code>	103, 153, 161, 232, 233, 237, 238, 370, 379, 383, 384	<code>\glsentrytype</code>	88
<code>\glsentrylongpl</code>	102, 112, 152, 154, 161, 232, 233, 237–240, 252, 259, 370, 379, 380, 383–387	<code>\Glsentryuseri</code>	142, 376
<code>\glsentrylongpluralaccess</code>	359	<code>\glsentryuseri</code>	142, 375, 376
<code>\Glsentryname</code>	138, 216, 374, 378	<code>\glsentryuseriaccess</code>	359
<code>\glsentryname</code>	138, 323, 374, 378	<code>\Glsentryuserii</code>	143, 376
<code>\glsentrynumberlist</code>	162, 183	<code>\glsentryuserii</code>	143, 376
<code>\Glsentryplural</code> .	105, 108, 136, 361, 365, 373	<code>\glsentryuseriaccess</code>	359
<code>\glsentryplural</code>	104, 105, 108, 136, 137, 361, 362, 365, 373	<code>\Glsentryuseriii</code>	144, 376
<code>\glsentrypluralaccess</code>	358	<code>\glsentryuseriii</code>	143, 144, 376
<code>\Glsentryprefix</code>	271	<code>\glsentryuseriiiaccess</code>	360
<code>\glsentryprefix</code>	270, 272	<code>\Glsentryuseriv</code>	144, 377
<code>\Glsentryprefixfirst</code>	271	<code>\glsentryuseriv</code>	144, 145, 377
<code>\glsentryprefixfirst</code>	270, 273	<code>\glsentryuserivaccess</code>	360
<code>\Glsentryprefixfirstplural</code>	272	<code>\Glsentryuserv</code>	145, 377
<code>\glsentryprefixfirstplural</code>	270, 273	<code>\glsentryuserv</code>	145, 146, 377
<code>\Glsentryprefixplural</code>	272	<code>\glsentryuservaccess</code>	360
<code>\glsentryprefixplural</code>	270, 273	<code>\Glsentryuservi</code>	146, 377
<code>\glsentryprevcount</code>	98–100	<code>\glsentryuservi</code>	146, 377
<code>\Glsentryshort</code>	110, 147, 155, 156, 233, 239, 240, 368–370, 380, 385, 386	<code>\glsentryuserviaccess</code>	360
<code>\glsentryshort</code>	110, 112, 147, 148, 155, 156, 161, 230, 232, 233, 235, 236, 238–241, 368–371, 379–382, 384–388	<code>\glsesclocationsfalse</code>	181
<code>\glsentryshortaccess</code>	359	<code>\glsesclocationstrue</code>	10
<code>\Glsentryshortpl</code>	110, 149, 233, 239, 240, 367, 380, 386, 387	<code>\glsfieldaccsupp</code>	356, 357
		<code>\glsfieldfetch</code>	158
		<code>\glsfirstaccessdisplay</code>	363, 364, 366, 367, 373
		<code>\glsfirstpluralaccessdisplay</code>	361, 362, 365, 373
		<code>\glsfirstpluralacessdisplay</code>	366
		<code>\glsngenacfmt</code>	232, 233, 239, 379, 380, 385
		<code>\glsngenentryfmt</code>	232, 233, 238, 239, 243, 245, 247, 249, 251, 254, 256, 259, 379, 380, 383, 385

<code>\glsgetgrouptitle</code>	<code>\glslinkvar</code>	115
.... 276, 280, 281, 299–304, 319–322, 326	<code>\glslistdottedwidth</code>	281, 282, 335
<code>\gls glossarymark</code>	<code>\glslistgroupheaderfmt</code>	280, 281
46	<code>\glslistnavigationitem</code>	280, 281
<code>\gls groupheading</code> 170, 208, 277, 279–281,	<code>\glslocalreset</code>	97
283, 284, 286, 294, 295, 297, 299–306,	<code>\glslocalunset</code>	98, 129–134
308, 312, 313, 315, 318–323, 325, 326, 332	<code>\gls longaccessdisplay</code> 369, 371, 372, 379–389	
<code>\gls groupskip</code> 169, 170, 208, 278, 280, 283,	<code>\gls longkey</code>	393
285, 286, 289, 290, 294–297, 305, 307,	<code>\gls longpluralaccessdisplay</code>	
309, 312, 314, 316, 319, 320, 322, 325, 331 370, 379, 380, 383–387, 389	
<code>\glshyperfirstfalse</code>	<code>\gls longpluralkey</code>	393
239, 385	<code>\gls longtok</code>	
<code>\glshyperfirsttrue</code> 229, 230, 232, 233, 238, 239, 243–	
29	253, 255–260, 379, 380, 384, 385, 388–393	
<code>\glshyperlink</code>	<code>\glsLTpenaltycheck</code>	292, 293
196	<code>\gls mcols</code>	299–304
<code>\glshypernavsep</code>	<code>\gls nameaccessdisplay</code>	374, 378, 379
276	<code>\gls namefont</code>	215–217, 347, 348, 378
<code>\glshypernumber</code>	<code>\gls navhyperlink</code>	276
45, 223, 224	<code>\gls navhyperlinkname</code>	274, 275
<code>\gls ifhyperon</code>	<code>\gls navhypertarget</code>	
115 280, 281, 300–304, 320–322, 326	
<code>\gls ifListOfAcronyms</code>	<code>\gls navigation</code>	
19 280, 281, 299–304, 319, 321, 322, 326	
<code>\gls ifplural</code>	<code>\gls nextpages</code>	10, 72, 198
104, 107, 110,	<code>\gls nogroupskipfalse</code>	12
111, 129–133, 147–153, 237, 245, 249,	<code>\gls noidxdisplayloc</code>	184, 185
252, 254, 360, 364, 367, 368, 370–372, 382	<code>\gls noidxdisplaylocclishandler</code>	183
<code>\gls ifusetranslator</code> 27, 28, 41, 42, 396	<code>\gls noidxloclist</code>	183, 208
<code>\gls indexonlyfirstfalse</code>	<code>\gls noidxloclisthandler</code>	209
29	<code>\gls noidxnumberlistloophandler</code>	184
<code>\gls inlinedescformat</code>	<code>\gls noidxstripaccents</code>	24
277, 334	<code>\gls nomakeindexwarning</code>	172
<code>\gls inlinedopostchild</code>	<code>\gls nonextpages</code>	72, 198
277, 334	<code>\gls nopostdotfalse</code>	12
<code>\gls inlineemptydescformat</code>	<code>\gls noxindywarning</code> 44, 50–52, 55–57, 165, 166	
277, 334	<code>\gls numberformat</code>	162
<code>\gls inlineformat</code>	<code>\gls numberlistloop</code>	184
277, 334	<code>\gls numbersgroupname</code>	36, 42, 219
<code>\gls inlineparentchildseparator</code> 277, 334	<code>\gls numlistlastsep</code>	162, 183
<code>\gls inlinepostchild</code>	<code>\gls numlistparser</code>	162, 180
277, 334	<code>\gls numlistsep</code>	162, 183
<code>\gls inlineseparator</code>	<code>\gls openbrace</code> 54, 55, 169, 170, 331, 332	
277, 334	<code>\gls order</code>	30, 176–179, 204
<code>\gls inlinesubdescformat</code>	<code>\gls org@endtheglossary</code>	5
278, 334	<code>\gls org@PrintChanges</code>	5
<code>\gls inlinesubnameformat</code>	<code>\gls org@theglossary</code>	5
277, 334	<code>\gls pagelistwidth</code> 284, 285, 287, 288, 291,	
<code>\gls inlinesubseparator</code>	292, 295–298, 306–308, 310, 311, 313–317	
278, 334		
<code>\gls insert</code>		
. 104–111, 129–133, 147–153, 237, 238,		
245, 249, 251, 252, 254, 361–372, 383, 384		
<code>\gls keylisttok</code>		
.... 229, 244, 246–253, 255–260, 388–393		
<code>\gls label</code> 87, 104–111, 116–118, 147–		
153, 232, 233, 237, 239, 245, 249, 251,		
252, 254, 361–369, 379, 380, 382, 383, 385		
<code>\gls labeltok</code>		
229,		
243, 245–247, 249–253, 255–260, 389–392		
<code>\gls link</code>		
230, 238–240, 245, 384, 386		
<code>\gls link options</code>		
counter		114, 128, 266
format		114, 128, 221
hyper		114, 116, 117, 128
local		114
<code>\gls linkcheckfirsthyperhook</code>		116
<code>\gls linkpostsetkeys</code>		117

<code>\glspatchLToutput</code>	289–291	<code>\glssstepentry</code>	214
<code>\glspenaltygroupskip</code>	289, 290	<code>\glssstepsubentry</code>	214
<code>\glspersentchar</code>	78, 169, 171	<code>\glsssubentrycounterfalse</code>	13
<code>\Glspl</code>	102, 243	<code>\glsssubentrycounterlabel</code>	214
<code>\glspl</code>	102, 243	<code>\glsssubentryitem</code> 278, 279, 281–284, 286, 294, 295, 297, 305, 307, 308, 312, 314, 315, 319, 320, 322, 324, 334–341, 343–346	
<code>\glspluralaccessdisplay</code> 361, 362, 365, 373		<code>\glssymbolaccessdisplay</code>	
<code>\glspluralsuffix</code>	40, 90, 232, 233, 380, 385–387 362–364, 375, 378, 379	
<code>\glspostdescription</code>	42, 278–281, 283, 294, 305, 312, 318– 320, 322, 324, 325, 334–337, 339–343, 345	<code>\glssymbolpluralaccessdisplay</code>	361, 362, 375
<code>\glspostinline</code>	277	<code>\glssymbolsgroupname</code>	36, 42, 219
<code>\glspostlinkhook</code>	116, 129–134, 147–154, 370–372	<code>\glstarget</code>	217, 278–284, 286, 294, 295, 297, 305, 307, 308, 312–315, 318–322, 324, 325, 334–346
<code>\glsprefixsep</code>	270–273	<code>\glstextaccessdisplay</code> 362, 363, 366, 367, 372	
<code>\glsprestandardsort</code>	14	<code>\glstextformat</code>	116, 118
<code>\glreset</code>	97	<code>\glstextup</code>	40, 387
<code>\glresetentrycounter</code>	213	<code>\glstildechar</code>	51, 169, 170
<code>\glresetentrylist</code> 169, 206, 212, 331		<code>\glstranslatefalse</code>	27, 28
<code>\glresetsubentrycounter</code> ... 214, 277, 334		<code>\glstranslatetrue</code>	28, 29
<code>\glssanitizesortfalse</code>	26	<code>\glstreechildpredesc</code>	319, 320
<code>\glssanitizesorttrue</code>	26	<code>\glstreegroupheaderfmt</code>	
<code>\glssavenumberlistfalse</code>	10 299–304, 319, 321, 322, 326	
<code>\glssavewritesfalse</code>	33	<code>\glstreeindent</code> .. 320, 322, 324, 325, 340–342	
<code>\glseeefmt</code>	168, 182, 184, 331	<code>\glstreeitem</code>	299, 300, 318
<code>\glseeitem</code>	195	<code>\glstreenamebox</code>	324, 325
<code>\glseeitemformat</code>	196	<code>\glstreenamefmt</code>	317–321, 323–325
<code>\glseeelastsep</code>	195	<code>\glstreenavigationfmt</code>	
<code>\glseeelist</code>	195 299–304, 319, 321, 322, 326	
<code>\glseeesep</code>	195	<code>\glstreepredesc</code>	318, 320, 322
<code>\glssetexpandfield</code>	22, 25, 26	<code>\glstreesubitem</code>	299, 318
<code>\glssetnoexpandfield</code>	22, 23, 25, 26	<code>\glstreesubsubitem</code>	299, 318
<code>\GlsSetQuote</code>	87, 171	<code>\glstype</code> . 116, 117, 129–134, 147–154, 370–372	
<code>\glssettocitle</code>	42, 198	<code>\glsucmarkfalse</code>	12
<code>\glsshortaccessdisplay</code>	156, 368–371, 379–382, 384–389	<code>\glsucmarktrue</code>	12
<code>\glsshortaccsupp</code>	357	<code>\glsunset</code>	95, 97, 99, 100, 129–134
<code>\glsshortkey</code>	393	<code>\glsupacrpluralsuffix</code>	234, 240, 247, 251, 253, 256
<code>\glsshortpluralaccessdisplay</code>	367, 368, 370, 379, 380, 384–387, 389	<code>\GlsUseAcrEntryDisplayStyle</code>	231, 234–236, 239–241, 381, 382, 385, 387, 388
<code>\glsshortpluralkey</code>	393	<code>\GlsUseAcrStyleDefs</code>	231, 234–236, 239–241, 381, 382, 385, 387, 388
<code>\glsshorttok</code> 229, 243–253, 255–260, 388–393		<code>\glssuseriaccessdisplay</code>	375, 376
<code>\glsshowaccsupp</code>	6	<code>\glssuseriiaccessdisplay</code>	376
<code>\glsshowtarget</code>	7	<code>\glssuseriiiaccessdisplay</code>	376
<code>\glsshowtargetfont</code>	6	<code>\glssuserivaccessdisplay</code>	377
<code>\glsshowtargetouter</code>	6	<code>\glssuservaccessdisplay</code>	377
<code>\glsshowtargetsymbols</code>	6	<code>\glssuserviaccessdisplay</code>	377
<code>\glssortnumberfmt</code>	15, 16		
<code>\glsspace</code>	226		

<code>\glswrallowprimitivemodstrue</code>	190	110, 179, 181, 207, 208, 229, 230, 237,
<code>\glswrite</code> ...	166–172, 179, 185, 186, 329–333	245, 249, 251, 253, 254, 360, 364, 367, 382
<code>\glswritedefhook</code>	78	<code>\ifdefequal</code>
<code>\glswriteentry</code>	188	. 31, 32, 61–64, 75, 76, 79, 89, 93, 208, 357
<code>\glswritefiles</code>	33, 185	<code>\ifdefstrequal</code>
<code>\glsxindyfalse</code>	30	85
<code>\glsxindytrue</code>	31	<code>\ifdefstring</code>
		11,
		41, 65, 176–178, 180, 204, 205, 207, 209, 348
		<code>\ifdefvoid</code>
		23, 92, 208
		<code>\ifdim</code>
		233, 292, 323
		<code>\iffalse</code>
		91, 97
<code>\H</code>	24	<code>\IfFileExists</code>
<code>\hangindent</code>		11, 27, 28, 177, 178, 199
	303, 304, 317, 320–322, 324–326, 339–342	<code>\ifglossaryexists</code> ..
<code>\hbox</code>	95, 281, 282, 335	45, 57, 61, 176–178, 198
<code>\hfill</code>	281, 282, 335	<code>\ifgl@sanitize@description</code>
<code>\hline</code> ...	283–285, 287, 294–296, 298, 305–317	25
<code>\hsize</code>	282, 293, 304, 311	<code>\ifgl@sanitize@name</code>
<code>\hspace</code>	318	25
<code>\hss</code>	281, 282, 335	<code>\ifgl@sanitize@symbol</code>
<code>\ht</code>	292	25
<code>\hyperdef</code>	38	<code>\ifgl@xindy@gl@numbers</code>
<code>\hyperlink</code>	114, 127, 223	57
<code>hyperref</code> package	193, 196, 221, 222, 266	<code>\ifgl@acr@description</code>
<code>\hypertarget</code>	127	258
		<code>\ifgl@acr@dua</code>
		247, 253, 256, 258
		<code>\ifgl@acr@footnote</code>
		116, 258
		<code>\ifgl@acr@synonym</code>
		17, 18
		<code>\ifgl@acr@shortcuts</code>
		37, 243
		<code>\ifgl@acr@smallcaps</code> ..
		247, 248, 251, 253, 256
		<code>\ifgl@acr@smaller</code>
		247, 248, 251, 253
		<code>\ifgl@automake</code>
		33, 181
		<code>\ifgl@desc@suppressed</code>
		277
		<code>\ifgl@sentrycounter</code>
		12, 13, 38, 213, 214
		<code>\ifgl@sentryexists</code>
		60, 77, 78, 86, 89
		<code>\ifgl@esc@locations</code>
		189
		<code>\ifgl@has@children</code>
		277, 334
		<code>\ifgl@has@desc</code>
		277
		<code>\ifgl@has@long</code>
		101–103, 155,
		232, 233, 237, 239, 251, 379, 380, 382, 385
		<code>\ifgl@has@parent</code>
		202, 207, 323
		<code>\ifgl@has@prefix</code>
		270–272
		<code>\ifgl@has@prefix@first</code>
		270, 271, 273
		<code>\ifgl@has@prefix@first@plural</code> ..
		270, 272, 273
		<code>\ifgl@has@prefix@plural</code>
		270, 272, 273
		<code>\ifgl@has@symbol</code>
	 245, 249, 254, 318–320, 322, 324, 325
		<code>\ifgl@hyper@first</code>
		116
		<code>\ifgl@index@only@first</code>
		188
		<code>\ifgl@no@group@skip</code>
		280, 283,
		285, 286, 289, 290, 294, 295, 297, 305,
		307, 309, 312, 314, 316, 319, 320, 322, 325
		<code>\ifgl@no@number@list</code>
		212
		<code>\ifgl@no@post@dot</code>
		11
		<code>\ifgl@number@line</code>
		49
		<code>\ifgl@ss@sanitizesort</code>
		23, 26
		<code>\ifgl@ss@avenumber@list</code>
		74, 180, 196

\ifglssavewrites	33, 176, 187	
\ifglssubentrycounter	13, 38, 213, 214	
\ifglstoc	49	
\ifglstranslate	41	
\ifglsucmark	46, 47	
\ifglused	100, 104–110, 116, 164, 188, 245, 249, 251, 254, 270–273, 361–367	
\ifglswrallowprimitivemods	191	
\ifglsxindy	43, 44, 49, 50, 52, 54–57, 67, 93, 94, 121, 165, 166, 172, 176, 177, 193, 194, 199, 327–329	
\ifignoredglossary	88, 92, 187	
\ifin@	37	
\ifinlistcs	205, 210	
\ifinner	6	
\ifKV@glslink@hyper	117, 118	
\ifKV@glslink@local	129–134	
\ifmeasuring@	95	
\ifmmode	6	
\ifnum	14, 15, 33, 99, 100, 179, 207, 292, 320, 322, 324, 340, 341	
\ifstrempty	7, 334	
\ifstrequal	219	
\ifthenelse	26, 38, 48, 118, 180, 219, 258, 275	
\IfTrackedLanguage	172	
\IfTrackedLanguageFileExists	41, 396, 397	
\iftrue	92, 96	
\ifundef	12, 13, 66, 77, 88, 166, 171, 179	
\ifvmode	164	
\ifvoid	292	
\ifx	12, 14, 16, 19, 36, 37, 39, 50, 51, 53, 54, 57, 58, 88–91, 94, 118, 119, 121–126, 155, 156, 167, 170, 172–175, 186, 189, 190, 192– 195, 197, 198, 219, 220, 222, 223, 244, 246, 248, 250, 253, 254, 256, 257, 259, 260, 329–331, 333, 334, 339–342, 347, 357	
\immediate	32, 77, 78, 100, 176, 178, 186, 200	
\in@	37	
\indexname	37	
\indexspace	280, 299–304, 319–322, 325, 326	
\input	40, 103	
\inputencodingname	31	
\InputIfFileExists	78	
\istfilename	43, 167, 171, 177–179, 329, 332	
\item	279–282, 299, 300, 318, 319, 334, 335, 339	
J		
\jobname	32, 43, 77, 78, 167, 171, 176–178, 199, 329, 332	
		K
\key@ifundefined	80, 81	
\KV@glslink@hyperfalse	114, 116, 117, 128	
\KV@glslink@hypertrue	114, 128	
		L
\L	24	
\l	24	
\label	9, 211, 213	
\language	30	
\leaders	281, 282, 335	
\leavevmode	86, 117	
\let	5, 11, 14–18, 24, 25, 27, 28, 32–38, 41, 42, 51, 52, 63–65, 74, 76, 77, 86–89, 91, 92, 95–98, 100, 103, 115–118, 120, 127–134, 147–153, 155, 156, 162, 170–172, 175–178, 180–182, 184, 187–189, 191, 195, 197–199, 208, 210, 213, 217, 229, 242–244, 246, 248– 255, 257, 267, 275, 276, 292, 299, 300, 318, 333, 351, 352, 370–372, 389–393, 396	
\letcs	61– 64, 78, 80, 81, 84, 90, 154, 155, 177, 178, 183, 184, 201, 203, 207, 208, 215, 219, 323	
link text	<u>103</u>	
\listcsadd	205	
\listcsgadd	210	
\listcsxadd	201	
\listead	205	
\loadglsentries	103	
\long	86, 223	
\longnewglossaryentry	86	
longtable package	<u>282</u> , <u>289</u> , <u>293</u>	
\LT@end@pen	292	
\LT@err	292	
\LT@foot	292, 293	
\LT@head	292, 293	
\LT@lastfoot	292	
\LT@output	292	
		M
\makeatletter	78, 199	
\makebox	281, 282, 323–325, 335, 341, 342	
makeglossaries	30, 43, 56, 57, 66, 172, 179, 199	
\makeglossaries	7, 8, 31–35, 39, 71, 176, 182, 183, 185, 200	
\makeglossary	34, 35	
makeindex	398	

makeindex	10, 14, 30, 33, 40, 43–45, 49, 65–67, 69, 94, 119, 122, 164, 168, 171, 172, 175, 186, 190–193, 218, 328, 329
delim_n	45
delim_r	45
page_compositor	43
special characters	121, 164
\makenoidxglossaries	7, 8, 31, 32, 71, 180, 184, 185
\MakeTextUppercase	4
\MakeUppercase	362–364, 371, 372
\marginpar	6
\markboth	46
\mbox	164, 280, 303, 304, 323, 335
memoir class	186
\memUChad	46
\MessageBreak	21, 34, 65, 197, 198, 347, 396, 397
mfistuc package	1, 155
\mfistucMakeUppercase	4, 47, 82, 105, 107–111, 135–146, 148, 150, 152, 154, 230, 237, 238, 240, 249, 254, 272, 273, 365–369, 372–377, 383, 384, 386
\midrule	289, 290
\month	167, 171, 329, 332
multicol package	298
N	
\n	171, 332
\NeedsTeXFormat	4, 267, 327, 333, 347, 396
\new@glossaryentry	77, 183
\new@ifnextchar	65, 81, 82, 101, 102, 128–132, 134–153, 225–228, 269–273
\newacronym	224, 229, 244, 246, 248, 250, 253, 256, 257, 259
\newacronymhook	229, 245, 247, 249, 251, 253, 256, 258, 260, 388
\newacronymstyle	231–237, 239–241
\newcommand	6–24, 26, 27, 29, 30, 32–34, 36–40, 42–62, 64–72, 74–86, 92, 93, 95–98, 100–104, 107, 110, 112, 113, 115–118, 120, 121, 127–166, 172, 175–179, 181, 184–189, 191–197, 199, 201–207, 209–221, 223–231, 233, 241, 243–254, 256–259, 261–266, 268–276, 278, 279, 292, 299, 317, 318, 323, 333, 349, 353–357, 379, 388, 393–395
\newcount	15, 74
\newcounter	12, 13
\newenvironment	214
\newglossary	17, 18, 36, 37, 67, 180
\newglossaryentry	7, 37, 74, 77, 98, 229, 243, 246, 247, 250, 252, 255, 257, 259, 389–392
\newglossaryentry options	
access	351, 353
counter	70
description	29, 69, 74, 76, 87, 138, 156, 225, 253, 350
descriptionaccess	354, 358
descriptionplural	139, 350
descriptionpluralaccess	354, 358
entrycounter	211
first	70, 90, 128, 135, 157, 250, 256, 349
firstaccess	354, 358
firstplural	70, 137, 158, 349
firstpluralaccess	354, 358
format	167
long	110, 160, 350
longaccess	355, 359
longplural	161, 350
longpluralaccess	355, 359
name	69, 74, 76, 87, 138, 155, 196, 349, 351
nonumberlist	72
parent	71, 76
plural	70, 90, 136, 349
pluralaccess	354, 358
prefix	267
prefixfirst	267
prefixfirstplural	268
prefixplural	268
see	7, 10, 71, 77, 180, 182
short	110, 160, 350, 356
shortaccess	354, 359, 388
shortpl	356
shortplural	160, 350
shortpluralaccess	354, 359
sort	69, 158, 187, 218
symbol	69, 70, 140, 246–248, 250, 255, 286, 308, 349, 351, 356
symbolaccess	354, 358
symbolplural	141, 349
symbolpluralaccess	354, 358
text	69, 70, 128, 134, 156, 246, 250, 349
textaccess	353, 357
type	17, 70, 103, 158
user1	142, 159, 350
user1access	355, 359
user2	142, 159, 350

user2access	355, 359	\ns@Acrshortpl	149
user3	143, 159, 350	\ns@acrshortpl	148
user3access	355, 359	\ns@newglossary	66
user4	144, 159, 351	\null	120–127, 173–175, 199
user4access	355, 360	\number	15, 78, 90, 100, 188, 191, 192, 217, 348
user5	145, 160, 351	\numberline	49
user5access	355, 360	\numexpr	100
user6	146, 160, 351		
user6access	355, 360		
\newglossarystyle	276, 279–291, 293–316, 318–323, 325, 326	O	
\newif	4, 5, 19, 27, 30, 32, 190	\O	24
\newlength	127, 282, 293, 304, 311, 321	\o	24
\newrobustcmd	7, 61–63, 76, 77, 82, 101, 102, 116, 128–153, 155–161, 163, 164, 195, 196, 225–228, 268–273, 323, 357–360	\OE	24
\newterm	37	\oe	24
\newtoks	121, 176, 228, 229	\openout	32, 77, 167, 171, 176, 329, 332
\newwrite	32, 77, 166, 171, 176, 179	\OR	258
\nfss@text	6	\or	5, 6, 8, 9, 28, 35, 211, 319, 339
ngerman package	172	\org@glossaryentrynumbers	198, 213
\noalign	292	\org@glossarytitle	197, 198
\nobreak	280, 293, 335	\org@glspostdescription	42
\noexpand	19, 39, 50, 51, 78, 91, 92, 113, 118–120, 126, 127, 162, 173–175, 177, 178, 180, 189, 190, 192, 196, 200, 203, 215, 217, 224, 229, 243, 244, 246–248, 250, 252, 255, 257, 259, 327, 347, 348, 389–393	\org@ifKV@glslink@hyper	117, 118
\nohyperpage	222	\outputpenalty	292
\noindent	217, 300–302, 304, 321, 322		
\noist	332, 333	P	
\nopostdesc	37, 43, 86, 198, 334	\p@	279, 299, 317, 318
\normalbaselineskip	292	\p@glshyp@opt	115
\ns@ACRfull	226, 227	package options:	
\ns@Acrfull	226	acronym	17, 18, 39, 197, 225
\ns@acrfull	225	true	18
\ns@ACRfullpl	228	counter	20
\ns@Acrfullpl	227	debug	
\ns@acrfullpl	227	showaccsupp	7
\ns@ACRlong	151	showtargets	6, 7
\ns@Acrlong	151	description	250, 251
\ns@acrlong	150	disablemakegloss	32
\ns@ACRlongpl	153	dua	249–251
\ns@Acrlongpl	153	entrycounter	12, 211, 213
\ns@acrlongpl	152	true	12
\ns@ACRshort	148	esclocations	421
\ns@Acrshort	147	false	10
\ns@acrshort	146, 147	footnote	129–134, 247, 249, 250, 253
\ns@ACRshortpl	149	hyperfirst	
		false	129–134
		index	36
		indexonlyfirst	405
		kernelglossredefs	
		nowarn	35
		makeindex	169, 266
		nogroupskip	283, 285, 286, 289, 290, 294, 295, 297, 305, 307, 309, 312, 314, 316

nolist	260	\printglossary	18, 21, 36, 37, 180, 197, 212
nolong	260, 282	\printglossary options	
nomain	17	entrycounter	211
nonumberlist	10	nogroupskip	211
nosuper	260	nonumberlist	212
notree	260	nopostdot	211
nowarn	5	numberedsection	211
numberline	8	style	210
record	269	subentrycounter	211
sanitize	25, 69, 155, 156	title	210
sanitizesort	22	toctitle	210
savewrites	33, 402	type	17, 196, 210
false	176	\printindex	37
true	179, 185	\printnoidxglossaries	182
section	8, 47	\printnoidxglossary	
sort		181, 182, 185, 197, 204, 205, 212
def	13, 14	\printnoidxglossary options	
none	13	sort	212
standard	13	\printnumbers	36
use	13, 14, 421	\printsymbols	36
style	9, 260	\ProcessOptions	267, 347
subentrycounter	13, 211, 213	\ProcessOptionsX	37
toc	8	\protect	49, 112, 232, 233, 239, 245, 249, 251, 369, 370, 379, 380, 385, 386
true	8	\protected@csedef	83
translate	28	\protected@csxdef	83
false	27	\protected@edef	7, 9, 51, 53, 56, 58, 88, 92, 93, 104–106, 112, 119, 162, 187, 190, 192, 211, 215, 217, 220, 229, 253, 259, 268, 274, 328, 329, 347, 348, 356, 357
translator	27	\protected@write	65, 66, 167, 169, 179, 182, 185, 187, 196, 275, 329
xindy	30, 31, 169, 266	\protected@xdef	14–16, 19, 24, 75, 94, 192, 352, 353
\PackageError	7, 8, 16, 33, 39, 50, 57, 60, 61, 65, 71, 74, 80–86, 88, 89, 98, 114, 154, 175, 176, 180, 183–185, 204–206, 210, 212, 220, 221, 230, 231, 247, 248, 253, 256, 333, 360	\providecommand	18, 31, 39, 40, 47, 65, 100, 128, 169, 179, 182, 185, 200, 215, 216, 269, 279, 299, 317, 348
\PackageInfo	5, 6, 32, 176, 187	\ProvidesFile	40
\PackageWarning	5, 6, 20	\ProvidesPackage	4, 267, 274, 276, 279, 282, 288, 293, 298, 304, 311, 317, 327, 333, 347, 396
\PackageWarningNoLine	5, 6, 21, 396, 397		
\pagegoal	292		
\pagelistname	42, 285, 287, 289, 290, 296, 298, 307–311, 314–317		
\par	43, 217, 279–281, 299, 301–304, 317, 318, 320–326, 335, 340–342		
\parindent	299–304, 318, 320–322, 324–326, 340–342		
\parskip	299–302, 318, 320, 321		
\PassOptionsToPackage	267, 347		
\penalty	292		
\phantomsection	48		
polyglossia package	27, 41		
\printglossaries	180		

R

\r	24
\raggedright	291–298, 312–317
\raisebox	127
\ref	214
\refstepcounter	213
\relax	5, 9, 11, 15–18, 27, 28, 32, 33, 35–38, 52, 65, 70, 72, 75, 78,

<code>\tablehead</code>	305–317	<code>\ttfamily</code>	6
<code>\tabletail</code>	305–317	<code>\TX@trial</code>	95
<code>\tabularnewline</code>	283–287, 289, 290, 294–298, 305–317, 337, 338, 343, 344	<code>\typeout</code>	21
<code>\texorpdfstring</code>	158	U	
<code>\textbar</code>	276	<code>\u</code>	24
<code>\textbf</code>	217, 223, 317, 339–342	<code>\uccode</code>	207
<code>textcase</code> package	4	<code>\undef</code>	72, 78, 196
<code>\textit</code>	223	<code>\unskip</code>	86, 281, 282, 335
<code>\textmd</code>	223	<code>\unvbox</code>	292, 293
<code>\textrm</code>	223	<code>\usedictionary</code>	41
<code>\textsc</code> ..	224, 234, 240, 247, 251, 253, 256, 387	<code>\usepackage</code>	204, 205
<code>\textsf</code>	223	<code>\UTFviii@two@octets</code>	25
<code>\textsl</code>	223	<code>\UTFviii@two@octets@combine</code>	25
<code>\textsmaller</code> ..	234, 240, 247, 251, 253, 256, 387	V	
<code>\texttt</code>	223	<code>\v</code>	24
<code>\textulc</code>	225	<code>\vbox</code>	292, 293
<code>\textup</code>	224, 225	<code>\vsize</code>	292, 293
<code>\TH</code>	25	<code>\vskip</code>	279, 292, 299, 317
<code>\th</code>	25	<code>\vss</code>	292, 293
<code>\the</code>	39, 41, 51, 56, 58, 66, 121–126, 167, 171, 173–175, 186, 187, 191, 192, 196, 207, 215, 217, 223, 229, 230, 232, 233, 238, 239, 243, 244, 246–248, 250, 252, 255, 257, 259, 327, 329, 332, 348, 379, 380, 384, 385, 388–393	W	
<code>\the@numberlist</code>	162	<code>\warn@nomakeglossaries</code>	180–182
<code>\theglossary</code>	5	<code>\warn@noprintglossary</code>	180–182, 199
<code>\theglossaryentry</code>	12, 214	<code>\write</code>	32, 78, 100, 167–172, 177, 178, 182, 186, 200, 329–333
<code>\theglossarysubentry</code>	13, 214	<code>\writeist</code>	175, 176, 333
<code>\theglentrycounter</code>	118, 119, 189, 192, 328, 329	X	
<code>\theH</code>	194	<code>\x</code>	223
<code>\theHglossaryentry</code>	12	<code>\xatlevel@</code>	118
<code>\theHglossarysubentry</code>	13	<code>\xcapitalisewords</code>	158
<code>\theHglentrycounter</code>	119, 189, 192	<code>\xdef</code>	88–91, 199, 275
<code>\thesection</code>	38	<code>\xglaccsupp</code>	357
<code>\this@dialect</code>	41, 42, 396, 397	<code>\xifinlistcs</code>	201, 202, 205
<code>\tiny</code>	6	<code>xindy</code>	398
<code>\toks@</code>	39, 41, 51, 56, 58, 66, 121–126, 173–175, 196, 215–217, 223, 327, 347, 348	<code>xindy</code>	10, 14, 24, 30, 31, 33, 43, 44, 49, 52, 54, 56–58, 94, 125, 126, 165, 166, 168, 186, 190, 193, 199, 218, 266, 328
<code>\toprule</code>	289, 290	<code>\xmakefirstuc</code> ...	104–106, 112, 154, 156, 268
<code>tracklang</code> package	41, 396	<code>\xspace</code>	224
<code>\trans@languages</code>	41	<code>xspace</code> package	4, 224
<code>\translate</code>	41, 42	Y	
<code>\translatelet</code>	17, 18, 36	<code>\year</code>	167, 171, 329, 332
<code>translator</code> package	17, 18, 27, 36, 40–42, 196	Z	
<code>\triangleright</code>	6	<code>\z@</code>	292