

Documented Code For glossaries v4.44

Nicola L.C. Talbot

Dickimaw Books

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

2019-12-06

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 v4.44: 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 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	37
1.4	Xindy	47
1.5	Loops and conditionals	56
1.6	Defining new glossaries	62
1.7	Defining new entries	67
1.8	Resetting and unsetting entry flags	93
1.9	Keeping Track of How Many Times an Entry Has Been Unset	96
1.10	Loading files containing glossary entries	101
1.11	Using glossary entries in the text	101
1.12	Adding an entry to the glossary without generating text	160
1.13	Creating associated files	162
1.14	Writing information to associated files	182
1.15	Glossary Entry Cross-References	191
1.16	Displaying the glossary	193
1.17	Acronyms	222
1.18	Predefined acronym styles	226
1.19	Predefined Glossary Styles	258
1.20	Debugging Commands	258
1.21	Compatibility with version 2.07 and below	264
2	Prefix Support (glossaries-prefix Code)	265
3	Glossary Styles	272
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	272
3.2	In-line Style (glossary-inline.sty)	274
3.3	List Style (glossary-list.sty)	277
3.4	Glossary Styles using longtable (the glossary-long package)	280
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	286
3.6	Glossary Styles using longtable (the glossary-longragged package)	291
3.7	Glossary Styles using multicol (glossary-mcols.sty)	296
3.8	Glossary Styles using supertabular environment (glossary-super package)	302
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	309
3.10	Tree Styles (glossary-tree.sty)	315

4 Backwards Compatibility	325
4.1 glossaries-compatible-207	325
4.2 glossaries-compatible-307	331
5 Accessibility Support (glossaries-accsupp Code)	345
5.1 Defining Replacement Text	346
5.2 Accessing Replacement Text	349
5.3 Displaying the Glossary	365
5.4 Acronyms	366
5.5 Debugging Commands	381
6 Multi-Lingual Support	383
6.1 Polyglossia Captions	383
Glossary	385
Change History	386
Index	410

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2019/12/06 v4.44 (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}[true]{%
35   \ifcase\gls@debug@nr\relax
36     \@gls@debugtrue
37     \renewcommand*\GlossariesWarning[1]{%
38       \PackageWarning{glossaries}{##1}%
39     }%
40     \renewcommand*\GlossariesWarningNoLine[1]{%
41       \PackageWarningNoLine{glossaries}{##1}%
42     }%
43     \let\@glsshowtarget\@gobble
44     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
45   \or
46     \@gls@debugfalse
47     \let\@glsshowtarget\@gobble
48     \PackageInfo{glossaries}{debug mode OFF}%
49   \or
50     \@gls@debugtrue
51     \renewcommand*\GlossariesWarning[1]{%
```

```

52     \PackageWarning{glossaries}{##1}%
53 }%
54 \renewcommand*{\GlossariesWarningNoLine}[1]{%
55     \PackageWarningNoLine{glossaries}{##1}%
56 }%
57 \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
58 \renewcommand{\@glsshowtarget}{\glsshowtarget}%
59 \fi
60 }

```

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

```

61 \newcommand*{\glsshowtarget}[1]{%
62     \ifmmode
63         \nfss@text{\ttfamily\small [#1]}%
64     \else
65         \ifinner
66             \texttt{\small [#1]}%
67         \else
68             \marginpar{\texttt{\small #1}}%
69         \fi
70     \fi
71 }

```

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

```

72 \newcommand*{\@glsshowtarget}[1]{%

```

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

`gls@see@noindex`

```

73 \newcommand*{\@gls@see@noindex}{%
74     \PackageError{glossaries}%
75     {'\gls@xr@key' key may only be used after \string\makeglossaries\space
76     or \string\makenoidxglossaries\space (or move
77     \string\newglossaryentry\space
78     definitions into the preamble)}}%
79 {You must use \string\makeglossaries\space
80     or \string\makenoidxglossaries\space before defining
81     any entries that have a '\gls@xr@key' key. It may
82     be that the 'see' key has been written to the .glsdefs
83     file from the previous run, in which case you need to
84     move your definitions
85     to the preamble if you don't want to use
86     \string\makeglossaries\space
87     or \string\makenoidxglossaries}%
88 }

```

`seenoinindex`

```

89 \define@choicekey{glossaries.sty}{seenoinindex}%

```

```

90 [\gls@seenoinindex@val\gls@seenoinindex@nr]{error,warn,ignore}{%
91 \ifcase\gls@seenoinindex@nr
92 \renewcommand*{\@gls@see@noindex}{%
93 \PackageError{glossaries}%
94 {\‘\gls@xr@key’ key may only be used after \string\makeglossaries\space
95 or \string\makenoidxglossaries}%
96 {You must use \string\makeglossaries\space
97 or \string\makenoidxglossaries\space before defining
98 any entries that have a ‘\gls@xr@key’ key}%
99 }%
100 \or
101 \renewcommand*{\@gls@see@noindex}{%
102 \GlossariesWarning{\‘\gls@xr@key’ key ignored}%
103 }%
104 \or
105 \renewcommand*{\@gls@see@noindex}{}%
106 \fi
107 }

```

`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.

```
108 \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`.

```
109 \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`.

```

110 \ifcsundef{chapter}%
111 {\newcommand*{\@@glossarysec}{section}}%
112 {\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 redefined `\glossarysection`.

```

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

```

Determine whether or not to use numbered sections.

`glossarysecstar`

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

```

`lossaryseclabel`

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

```

`\glsautoprefix` Prefix to add before label if automatically generated:

```
118 \newcommand*{\glsautoprefix}{}
```

`numberedsection`

```
119 \define@choicekey{glossaries.sty}{numberedsection}%
120 [\gls@numberedsection@val\gls@numberedsection@nr]{%
121 false,nolabel,autolabel,nameref}[nolabel]{%
122   \ifcase\gls@numberedsection@nr\relax
123     \renewcommand*{\@@glossarysecstar}{*}%
124     \renewcommand*{\@@glossaryseclabel}{}%
125   \or
126     \renewcommand*{\@@glossarysecstar}{}%
127     \renewcommand*{\@@glossaryseclabel}{}%
128   \or
129     \renewcommand*{\@@glossarysecstar}{}%
130     \renewcommand*{\@@glossaryseclabel}{%
131       \label{\glsautoprefix@glo@type}}%
132   \or
133     \renewcommand*{\@@glossarysecstar}{*}%
134     \renewcommand*{\@@glossaryseclabel}{%
135       \protected@edef\@currentlabelname{\glossarytoctitle}%
136       \label{\glsautoprefix@glo@type}}%
137   \fi
138 }
```

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`

```
139 \@ifpackageloaded{classicthesis}
140 {\newcommand*{\@glossary@default@style}{index}}
141 {\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`.

```
142 \define@key{glossaries.sty}{style}{%
143   \def\@glossary@default@style{#1}%
144 }
```

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`


```

145 \newcommand*{\@gls@declareoption}[2]{%
146   \DeclareOptionX{#1}{#2}%
147   \DeclareOption{#1}{#2}%
148 }

```

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

```

149 \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).

```

150 \@gls@declareoption{nonumberlist}{%
151   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
152 }

```

savenumberlist Provide means to store the number list for entries.

```

153 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{}
154 \glssavenumberlistfalse

```

eautionumberlist

```

155 \newcommand*{\@glo@seeautionumberlist}{}

```

eautionumberlist Automatically activates number list for entries containing the see key.

```

156 \@gls@declareoption{seeautionumberlist}{%
157   \renewcommand*{\@glo@seeautionumberlist}{%
158     \def\@glo@prefix{\glsnextpages}%
159   }%
160 }

```

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.

```

161 \define@boolkey{glossaries.sty}[gls]{esclocations}[true]{}
162 \glseesclocationstrue

```

\@gls@loadlong

```

163 \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.

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

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

```
165 \IfFileExists{supertabular.sty}{%
166   \newcommand*\@gls@loadsuper{\RequirePackage{glossary-super}}}{%
167   \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.

```
168 \@gls@declareoption{nosuper}{\renewcommand*\@gls@loadsuper{}}
```

`\@gls@loadlist`

```
169 \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`.

```
170 \@gls@declareoption{nolist}{%
171   \renewcommand*\@gls@loadlist{%
172     \ifdefstring{\@glossary@default@style}{list}%
173     {\let\@glossary@default@style\relax}%
174     }%
175   }%
176 }
```

`\@gls@loadtree`

```
177 \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.

```
178 \@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).

```
179 \@gls@declareoption{nostyles}{%
180   \renewcommand*\@gls@loadlong}{}%
181   \renewcommand*\@gls@loadsuper}{}%
182   \renewcommand*\@gls@loadlist}{}%
183   \renewcommand*\@gls@loadtree}{}%
184   \let\@glossary@default@style\relax
185 }
```

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.)

```

186 \newcommand*{\glspostdescription}{%
187   \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
188 }

```

nopostdot Boolean option to suppress post description dot

```

189 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
190 \glsnopostdotfalse

```

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

```

191 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
192 \glsnogroupskipfalse

```

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

```

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

194 \@ifclassloaded{memoir}
195 {%
196   \glsucmarktrue
197 }%
198 {%
199   \glsucmarkfalse
200 }

```

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

```

201 \newcommand*{\@gls@define@glossaryentrycounter}{%
202   \ifglsentrycounter
203     Define the glossaryentry counter if it doesn't already exist.
204     {%
205       \ifx\@gls@counterwithin\@empty
206         \newcounter{glossaryentry}%
207       \else
208         \newcounter{glossaryentry}[\@gls@counterwithin]%
209       \fi
210       \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
211     }%
212   {}%
213   \fi
214 }

```

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.

```

215 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
216 \glentrycounterfalse

```

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

```

217 \define@key{glossaries.sty}{counterwithin}{%
218   \renewcommand*{\@gls@counterwithin}{#1}%
219   \glentrycountertrue
220   \@gls@define@glossaryentrycounter
221 }

```

s@counterwithin The default value is no parent counter:

```

222 \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

```

223 \newcommand{\@gls@define@glossarysubentrycounter}{%
    Check if counter already defined.
224   \ifundef\c@glossarysubentry
225   {%
226     \ifglssubentrycounter
227     \ifglentrycounter
228       \newcounter{glossarysubentry}[glossaryentry]%
229     \else
230       \newcounter{glossarysubentry}%
231     \fi
    As with \theHglossaryentry, this starts with \currentglossary. to help avoid duplicate
    hyper targets.
232     \def\theHglossarysubentry{\currentglossary.\currentglssubentry.\theglossarysubentry}%
233     \fi
234   }%
235   {}%
236 }

```

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.

```

237 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
238 \glssubentrycounterfalse

```

efault@sorttype Initialise default sort for \printnoidxglossary

```

239 \newcommand*{\@glo@default@sorttype}{standard}

```

sort Define the sort method: sort=standard (default), sort=def (order of definition) or sort=use (order of use). If no indexing required, use sort=none.

```
240 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
241   \renewcommand*{\@glo@default@sorttype}{#1}%
242   \csname @gls@setupsort@#1\endcsname
243 }
```

glsprestandardsort `\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.

```
244 \newcommand*{\glsprestandardsort}[3]{%
245   \glsdosanitizesort
246 }
```

check@sortallowed

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

gls@setupsort@standard Set up the macros for default sorting.

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

Store entry information when it's defined.

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

No count register required for standard sort.

```
250   \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.)

```
251   \def\@gls@defsort##1##2{%
252     \ifx\@glo@sort\@glsdefaultsort
253       \let\@glo@sort\@glo@name
254     \fi
```

```
255     \let\glsdosanitizesort\@gls@sanitizesort
256     \glsprestandardsort{\@glo@sort}{##1}{##2}%
257     \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
258   }%
```

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

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

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

```
260   \let\@glo@check@sortallowed\@gobble
261 }
```

Set standard sort as the default:

```
262 \@gls@setupsort@standard
```

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

```
263 \newcommand*{\glssortnumberfmt[1]}{%
264   \ifnum#1<100000 0\fi
265   \ifnum#1<10000 0\fi
266   \ifnum#1<1000 0\fi
267   \ifnum#1<100 0\fi
268   \ifnum#1<10 0\fi
269   \number#1%
270 }
```

`s@setupsort@def` Set up the macros for order of definition sorting.

```
271 \newcommand*{\@gls@setupsort@def}{%
```

Store entry information when it's defined.

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

Defined count register associated with the glossary.

```
273   \def\@gls@defsortcount##1{%
274     \expandafter\global
275     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
276   }%
```

Increment count register associated with the glossary and use as the sort key.

```
277   \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.

```
278     \ifcsundef{glossary@##1@sortcount}%
279     {\@gls@defsortcount{##1}}%
280     {}%
281     \expandafter\global\expandafter
282     \advance\csname glossary@##1@sortcount\endcsname by 1\relax
283     \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
284       \expandafter\glssortnumberfmt
285       {\csname glossary@##1@sortcount\endcsname}}%
286   }%
```

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

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

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

```
288   \let\@glo@check@sortallowed\@gobble
289 }
```

`s@setupsort@use` Set up the macros for order of use sorting.

```
290 \newcommand*{\@gls@setupsort@use}{%
```

Don't store entry information when it's defined.

```
291   \let\do@glo@storeentry\@gobble
```

Defined count register associated with the glossary.

```
292 \def\@gls@defsortcount##1{%  
293   \expandafter\global  
294   \expandafter\newcount\csname glossary@##1@sortcount\endcsname  
295 }%
```

Initialise the sort key to empty.

```
296 \def\@gls@defsort##1##2{%  
297   \expandafter\gdef\csname glo@##2@sort\endcsname{%}  
298 }%
```

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

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

Get the parent, if one exists

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

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

```
301   \ifx\@glo@parent\@empty  
302   \else  
303     \expandafter\@gls@setsort\expandafter{\@glo@parent}%  
304   \fi
```

Set index information for this entry

```
305   \edef\@glo@type{\csname glo@##1@type\endcsname}%  
306   \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%  
307   \ifx\@gls@tmp\@empty  
308     \expandafter\global\expandafter  
309     \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax  
310     \expandafter\protected\xdef\csname glo@##1@sort\endcsname{%  
311       \expandafter\glssortnumberfmt  
312       {\csname glossary@\@glo@type @sortcount\endcsname}}%  
313     \@glo@storeentry{##1}%  
314   \fi  
315 }%
```

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

```
316 \let\@glo@check@sortallowed\@gobble  
317 }
```

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

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

Don't store entry index information.

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

No count register required for standard sort.

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

Don't modify sort value.

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

```

322 \expandafter\global\expandafter\let\csname glo###2@sort\endcsname\@glo@sort
323 }%

```

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

```

324 \def\@gls@setsort##1{%

```

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

```

325 \renewcommand\@glo@check@sortallowed[1]{\PackageError{glossaries}
326 {Option sort=none not allowed with \string##1}%
327 {(Use sort=def instead)}}%
328 }

```

`\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.)

```

329 \newcommand*\glsdefmain{%
330 \if@gls@docloaded
331 \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
332 \else
333 \newglossary{main}{gls}{glo}{\glossaryname}%
334 \fi

```

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

```

335 \newcommand*\gls@tr@set@main@toctitle{%
336 \translatelet{\glossarytoctitle}{Glossary}%
337 }%
338 }

```

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`

```

339 \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`

```

340 \newcommand*\acronymtype{\glsdefaulttype}

```

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

```

341 \@gls@declareoption{nomain}{%
342 \let\glsdefaulttype\relax
343 \renewcommand*\glsdefmain}{%
344 }

```


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.

```

345 \define@boolkey{glossaries.sty}{gls}{acronym}[true]{%
346   \ifglsacronym
347     \renewcommand{\@gls@do@acronymsdef}{%
348       \DeclareAcronymList{acronym}%
349       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
350       \renewcommand*\acronymtype{acronym}%

```

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

```

351     \newcommand*\@gls@tr@set@acronym@toctitle{%
352       \translatelet{\glossarytoctitle}{Acronyms}%
353     }%
354   }%
355 \else
356   \let\@gls@do@acronymsdef\relax
357 \fi
358 }

```

\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.

```

359 \AtBeginDocument{%
360   \ifglsacronym
361     \ifbool{glscompatible-3.07}%
362       {}%
363     {%
364       \providecommand*\printacronyms[1][]{%
365         \printglossary[type=\acronymtype,#1]}%
366     }%
367 \fi
368 }

```

@do@acronymsdef Set default value

```

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

```

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

```

370 \@gls@declareoption{acronyms}{%
371   \glsacronymtrue
372   \renewcommand{\@gls@do@acronymsdef}{%
373     \DeclareAcronymList{acronym}%
374     \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
375     \renewcommand*\acronymtype{acronym}%

```

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

```

376     \newcommand*\@gls@tr@set@acronym@toctitle{%
377       \translatelet{\glossarytoctitle}{Acronyms}%
378     }%
379   }%
380 }

```

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

```
381 \newcommand*{\@glsacronymlists}{}
```

`addtoacronymlists`

```
382 \newcommand*{\@addtoacronymlists}[1]{%
383   \ifx\@glsacronymlists\@empty
384     \protected@xdef\@glsacronymlists{#1}%
385   \else
386     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
387   \fi
388 }
```

`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.)

```
389 \newcommand*{\DeclareAcronymList}[1]{%
390   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
391 }
```

`IfListOfAcronyms`

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

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

```
392 \newcommand{\glsIfListOfAcronyms}[1]{%
393   \edef\@do@gls@islistofacronyms{%
394     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
395   \@do@gls@islistofacronyms
396 }
```

Internal command requires label and list to be expanded:

```
397 \newcommand{\@gls@islistofacronyms}[4]{%
398   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
399     \def\@before{##1}\def\@after{##2}}%
400   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
401   \ifx\@after\@nnil
```

Not found

```
402   #4%
403   \else
```

Found

```
404   #3%
405   \fi
406 }
```

`glsisacronymlist` Convenient boolean.

```
407 \newif\if@glsisacronymlist
```

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

```
408 \newcommand*{\gls@ckisacronymlist}[1]{%
409   \glsIfListOfAcronyms{#1}%
410   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
411 }
```

`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.)

```
412 \newcommand*{\SetAcronymLists}[1]{%
413   \renewcommand*{\@glsacronymlists}{#1}%
414 }
```

`acronymlists`

```
415 \define@key{glossaries.sty}{acronymlists}{%
416   \DeclareAcronymList{#1}%
417 }
```

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`

```
418 \newcommand{\glscounter}{page}
```

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

```
419 \define@key{glossaries.sty}{counter}{%
420   \renewcommand*{\glscounter}{#1}%
421 }
```

`gls@nohyperlist`

```
422 \newcommand*{\@gls@nohyperlist}{}%
```

`lareNoHyperList`

```
423 \newcommand*{\GlsDeclareNoHyperList}[1]{%
424   \ifdefempty\@gls@nohyperlist
425   {%
426     \renewcommand*{\@gls@nohyperlist}{#1}%
427   }%
428   {%
429     \appto\@gls@nohyperlist{,#1}%
430   }%
431 }
```

`nohypertypes`

```
432 \define@key{glossaries.sty}{nohypertypes}{%
433   \GlsDeclareNoHyperList{#1}%
434 }
```

`glossariesWarning` Prints a warning message.

```

435 \newcommand*{\GlossariesWarning}[1]{%
436   \PackageWarning{glossaries}{#1}%
437 }
```

`glossariesWarningNoLine` Prints a warning message without the line number.

```

438 \newcommand*{\GlossariesWarningNoLine}[1]{%
439   \PackageWarningNoLine{glossaries}{#1}%
440 }
```

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

```

441 \newcommand{\glosortentrieswarning}{%
442   \typeout{Using TeX to sort glossary entries---this may
443   take a while}%
444 }
```

`nowarn` Define package option to suppress warnings

```

445 \@gls@declareoption{nowarn}{%
446   \if@gls@debug
447     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
448   \else
449     \renewcommand*{\GlossariesWarning}[1]{}%
450     \renewcommand*{\GlossariesWarningNoLine}[1]{}%
451     \renewcommand*{\glosortentrieswarning}{}%
452     \renewcommand*{\@gls@missinglang@warn}[2]{}%
453   \fi
454 }
```

`missinglang@warn` Missing language warning.

```

455 \newcommand*{\@gls@missinglang@warn}[2]{%
456   \PackageWarningNoLine{glossaries}%
457   {No language module detected for '#1'.\MessageBreak
458   Language modules need to be installed separately.\MessageBreak
459   Please check on CTAN for a bundle called\MessageBreak
460   'glossaries-#2' or similar}%
461 }
```

`nolangwarn` Suppress warning if language support not found.

```

462 \@gls@declareoption{nolangwarn}{%
463   \renewcommand*{\@gls@missinglang@warn}[2]{}%
464 }
```

`nonglossdefined` Issue a warning if overriding `\printglossary`

```

465 \newcommand*{\@gls@warnonglossdefined}{%
466   \GlossariesWarning{Overriding \string\printglossary}%
467 }
```

theglossdefined Issue a warning if overriding theglossary

```

468 \newcommand*{\@gls@warnontheglossdefined}{%
469   \GlossariesWarning{Overriding 'theglossary' environment}%
470 }
```

noredefwarn Suppress warning on redefinition of \printglossary

```

471 \@gls@declareoption{noredefwarn}{%
472   \renewcommand*{\@gls@warnonglossdefined}{}%
473   \renewcommand*{\@gls@warnontheglossdefined}{}%
474 }
```

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

```

475 \newcommand*{\@gls@sanitizedesc}{%
476 }
```

lssetexpandfield `\glssetexpandfield{<field>}`

Sets field to always expand.

```

477 \newcommand*{\glssetexpandfield}[1]{%
478   \csdef{gls@assign@#1@field}##1##2{%
479     \@gls@expand@field{##1}{#1}{##2}%
480   }%
481 }
```

setnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```

482 \newcommand*{\glssetnoexpandfield}[1]{%
483   \csdef{gls@assign@#1@field}##1##2{%
484     \@gls@noexpand@field{##1}{#1}{##2}%
485   }%
486 }
```

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

```

487 \glssetexpandfield{type}
```

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

```

488 \glssetnoexpandfield{desc}
```

escplural@field

```

489 \glssetnoexpandfield{descplural}
```

```

ls@sanitizename
490 \newcommand*{\@gls@sanitizename}{}

sign@name@field Don't expand name by default.
491 \glssetnoexpandfield{name}

@sanitizesymbol
492 \newcommand*{\@gls@sanitizesymbol}{}

gn@symbol@field Don't expand symbol by default.
493 \glssetnoexpandfield{symbol}

bolplural@field
494 \glssetnoexpandfield{symbolplural}

Sanitizing stuff:

ls@sanitizesort
495 \newcommand*{\@gls@sanitizesort}{%
496   \ifglssanitizesort
497     \@gls@sanitizesort
498   \else
499     \@gls@nosanitizesort
500   \fi
501 }

ls@sanitizesort
502 \newcommand*\@gls@sanitizesort{%
503   \@onelevel@sanitize\@glo@sort
504 }

@nosanitizesort
505 \newcommand*{\@gls@nosanitizesort}{}

dx@sanitizesort Remove braces around first character (if present) before sanitizing.
506 \newcommand*\@gls@noidx@sanitizesort{%
507   \ifdefvoid\@glo@sort
508   {%
509     {%
510       \expandafter\@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
511     }%
512   }
513 \def\@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
514   \def\@glo@sort{#1#2}%
515   \@onelevel@sanitize\@glo@sort
516 }

```

@nosanitizesort

```
517 \newcommand*{\@@gls@noidx@nosanitizesort}{%
518   \ifdefvoid\@glo@sort
519   {}%
520   {%
521     \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
522   }%
523 }
524 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
525   \bgroup
526     \glsnoidxstripaccents
527     \protected@xdef\@@glo@sort{#1#2}%
528   \egroup
529   \let\@glo@sort\@@glo@sort
530 }
```

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.

```
531 \newcommand*\glsnoidxstripaccents{%
532   \let\IeC\@firstofone
533   \let\add@accent\@secondoftwo
534   \let\@text@composite\@secondoftwo
535   \let\@tabacckludge\@secondoftwo
536   \expandafter\def\csname \encodingdefault-cmd\endcsname##1##2##3{##3}%
537   \expandafter\def\csname OT1-cmd\endcsname##1##2##3{##3}%
538   \expandafter\def\csname T1-cmd\endcsname##1##2##3{##3}%
539   \expandafter\def\csname PD1-cmd\endcsname##1##2##3{##3}%
540   \let\' \@firstofone
541   \let\' \@firstofone
542   \let\^ \@firstofone
543   \let\" \@firstofone
544   \let\u \@firstofone
545   \let\t \@firstofone
546   \let\d \@firstofone
547   \let\r \@firstofone
548   \let\= \@firstofone
549   \let\.\@firstofone
550   \let\~ \@firstofone
551   \let\v \@firstofone
552   \let\H \@firstofone
553   \let\c \@firstofone
554   \let\b \@firstofone

555   \let\a \@secondoftwo
556   \def\AE{AE}%
557   \def\ae{ae}%
558   \def\OE{OE}%
```

```

559 \def\oe{oe}%
560 \def\AA{AA}%
561 \def\aa{aa}%
562 \def\L{L}%
563 \def\l{l}%
564 \def\O{O}%
565 \def\o{o}%
566 \def\SS{SS}%
567 \def\ss{ss}%
568 \def\th{th}%

569 \def\TH{TH}%
570 \def\dh{dh}%
571 \def\DH{DH}%
572 }

```

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

```

573 \@ifl@t@r\fmtversion{2019/10/01}
574 {%
575 \appto\glsnoidxstripaccents{\let\UTFviii@two@octets\UTFviii@two@octets@combine}%
576 }
577 {}

```

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.

```

578 \define@boolkey[glS]{sanitize}{description}[true]{%
579 \GlossariesWarning{sanitize={description} package option deprecated}%
580 \ifglS@sanitize@description
581 \glSsetnoexpandfield{desc}%
582 \glSsetnoexpandfield{descplural}%
583 \else
584 \glSsetexpandfield{desc}%
585 \glSsetexpandfield{descplural}%
586 \fi
587 }

588 \define@boolkey[glS]{sanitize}{name}[true]{%
589 \GlossariesWarning{sanitize={name} package option deprecated}%
590 \ifglS@sanitize@name
591 \glSsetnoexpandfield{name}%
592 \else
593 \glSsetexpandfield{name}%
594 \fi
595 }

596 \define@boolkey[glS]{sanitize}{symbol}[true]{%
597 \GlossariesWarning{sanitize={symbol} package option deprecated}%
598 \ifglS@sanitize@symbol
599 \glSsetnoexpandfield{symbol}%

```



```

600 \glsssetnoexpandfield{symbolplural}%
601 \else
602 \glsssetexpandfield{symbol}%
603 \glsssetexpandfield{symbolplural}%
604 \fi
605 }

```

sanitizesort

```

606 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
607 \ifglsssanitizesort
608 \glsssetnoexpandfield{sortvalue}%
609 \renewcommand*{\@gls@noidx@setsanitizesort}{%
610 \glsssanitizesorttrue
611 \glsssetnoexpandfield{sortvalue}%
612 }%
613 \else
614 \glsssetexpandfield{sortvalue}%
615 \renewcommand*{\@gls@noidx@setsanitizesort}{%
616 \glsssanitizesortfalse
617 \glsssetexpandfield{sortvalue}%
618 }%
619 \fi
620 }

```

Default setting:

```

621 \glsssanitizesorttrue
622 \glsssetnoexpandfield{sortvalue}%

```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```

623 \newcommand*{\@gls@noidx@setsanitizesort}{%
624 \glsssanitizesortfalse
625 \glsssetexpandfield{sortvalue}%
626 }

627 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
628 \setbool{glsssanitizesort}{#1}%
629 \ifglsssanitizesort
630 \glsssetnoexpandfield{sortvalue}%
631 \else
632 \glsssetexpandfield{sortvalue}%
633 \fi
634 \GlossariesWarning{sanitize={sort} package option
635 deprecated. Use sanitizesort instead}%
636 }

```

sanitize

```

637 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
638 \ifthenelse{\equal{#1}{none}}{}%
639 {%
640 \GlossariesWarning{sanitize package option deprecated}%

```

```

641     \glssetexpandfield{name}%
642     \glssetexpandfield{symbol}%
643     \glssetexpandfield{symbolplural}%
644     \glssetexpandfield{desc}%
645     \glssetexpandfield{descplural}%
646 }%
647 {%
648     \setkeys[gls]{sanitize}{#1}%
649 }%
650 }

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

otranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator
652 \newcommand*\@gls@usetranslator{%
    polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
    polyglossia as well.
653     \@ifpackageloaded{polyglossia}%
654     {%
655         \let\glsifusetranslator\@secondoftwo
656     }%
657     {%
658         \@ifpackageloaded{babel}%
659         {%
660             \IfFileExists{translator.sty}%
661             {%
662                 \RequirePackage{translator}%
663                 \let\glsifusetranslator\@firstoftwo
664             }%
665         }%
666     }%
667     {}%
668 }%
669 }

dtranslatordict Checks if given translator dictionary has been loaded.
670 \newcommand{\glsifusedtranslatordict}[3]{%
671     \glsifusetranslator
672     {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
673     {#3}%
674 }

notranslate Provide a synonym for translate=false that can be passed via the document class.
675 \@gls@declareoption{notranslate}{%

```

```

676 \glstranslatefalse
677 \let\@gls@usetranslator\relax
678 \let\glsifusetranslator\@secondoftwo
679 }

```

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

```

680 \define@choicekey{glossaries.sty}{translate}%
681 [\gls@translate@val\gls@translate@nr]%
682 {true,false,babel}[true]%
683 {%
684   \ifcase\gls@translate@nr\relax
685     \glstranslatetrue
686     \renewcommand*\@gls@usetranslator{%
687       \@ifpackageloaded{polyglossia}%
688       {%
689         \let\glsifusetranslator\@secondoftwo
690       }%
691       {%
692         \@ifpackageloaded{babel}%
693         {%
694           \IfFileExists{translator.sty}%
695           {%
696             \RequirePackage{translator}%
697             \let\glsifusetranslator\@firstoftwo
698           }%
699           {}%
700         }%
701         {}%
702       }%
703     }%
704   \or
705     \glstranslatefalse
706     \let\@gls@usetranslator\relax
707     \let\glsifusetranslator\@secondoftwo
708   \or
709     \glstranslatetrue
710     \let\@gls@usetranslator\relax
711     \let\glsifusetranslator\@secondoftwo
712   \fi
713 }

```

Set the default value:

```

714 \glstranslatefalse
715 \let\glsifusetranslator\@secondoftwo
716 \@ifpackageloaded{translator}%
717 {%
718   \glstranslatetrue
719   \let\glsifusetranslator\@firstoftwo
720 }%

```

```

721 {%
722   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
723   {
724     \@ifpackageloaded{\gls@thissty}%
725     {%
726       \glstranslatetrue
727       \@endfortrue
728     }%
729   }%
730 }
731 }

```

indexonlyfirst Set whether to only index on first use.

```

732 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
733 \glsindexonlyfirstfalse

```

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

```

734 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
735 \glshyperfirsttrue

```

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

```

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

```

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

```

737 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}
738 \ifbool{glsacrdescription}%
739 {}%
740 {}%
741 \renewcommand*{\@gls@sanitizedesc}{}%
742 }%
743 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
744 }

```

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

```

745 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}
746 \renewcommand*{\@gls@sanitizesymbol}{}%
747 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
748 }

```

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

```

749 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}
750 \renewcommand*{\@gls@sanitizesymbol}{}%
751 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
752 }

```

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

```

753 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
754   \renewcommand*{\@gls@sanitizesymbol}{}}%
755   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
756 }

```

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

```

757 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
758   \renewcommand*{\@gls@sanitizesymbol}{}}%
759   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
760 }

```

shortcuts Define acronym shortcuts.

```

761 \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.

```

762 \newcommand*{\glsorder}{word}

```

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

```

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

```

order

```

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

```

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

```

766 \newif\ifglxindy

```

The default is makeindex:

```

767 \glxindyfalse

```

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

```

768 \@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.

```

769 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
770 \gls@xindy@glsnumberstrue

```

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.)

```

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

```

Define key to set the language

```

772 \define@key[gls]{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.

```

773 \ifcsundef{inputencodingname}{%
774   \def\gls@codepage{}}{%
775   \def\gls@codepage{\inputencodingname}
776 }

```

Define a key to set the code page.

```

777 \define@key[gls]{xindy}{codepage}{\def\gls@codepage{#1}}

```

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

```

778 \define@key{glossaries.sty}{xindy}[]{%
779   \glsxindytrue
780   \setkeys[gls]{xindy}{#1}%
781 }

```

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

```

782 \@gls@declareoption{xindygloss}{%
783   \glsxindytrue
784 }

```

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

```

785 \@gls@declareoption{xindynoglsnumbers}{%
786   \glsxindytrue
787   \gls@xindy@glsnumbersfalse
788 }

```

`\ifglsautomake`

```

789 \newif\ifglsautomake

```

`gls@automake@nr`

```

790 \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.

```

791 \define@choicekey{glossaries.sty}{automake}{%
792   [\gls@automake@val\gls@automake@nr]{true,false,immediate}[true]{%
793     \ifnum\gls@automake@nr=1\relax
794       \glsautomakefalse
795     \else
796       \glsautomaketrue
797     \fi
798   \ifglsautomake
799     \renewcommand*{\@gls@doautomake}{%
800       \PackageError{glossaries}{You must use
801         \string\makeglossaries\space with automake=true}

```

```

802      {%
803          Either remove the automake=true setting or
804          add \string\makeglossaries\space to your document preamble.%
805      }%
806  }%
807  \else
808      \renewcommand*{\@gls@doautomake}{}%
809  \fi
810 }
811 \glsautomakefalse

@gls@doautomake
812 \newcommand*{\@gls@doautomake}{}
813 \AtEndDocument{\@gls@doautomake}

savewrites The savewrites package option is provided to save on the number of write registers.
814 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
815     \ifglssavewrites
816         \renewcommand*{\glswritefiles}{\@glswritefiles}%
817     \else
818         \let\glswritefiles\@empty
819     \fi
820 }

Set default:
821 \glssavewritesfalse
822 \let\glswritefiles\@empty

compatible-3.07
823 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{}
824 \boolfalse{glscompatible-3.07}

compatible-2.07
825 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
    Also set 3.07 compatibility if this option is set.
826     \ifbool{glscompatible-2.07}%
827     {%
828         \booltrue{glscompatible-3.07}%
829     }%
830     {}%
831 }
832 \boolfalse{glscompatible-2.07}

al@makeglossary Store the original definition.
833 \let\gls@original@makeglossary\makeglossary

iginal@glossary Store the original definition.
834 \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.)

```
835 \def\makeglossary{%
836   \GlossariesWarning{Use of \string\makeglossary\space with
837   glossaries.sty is \MessageBreak deprecated. Use \string\makeglossaries\space
838   instead. If you \MessageBreak need the original definition of
839   \string\makeglossary\space use \MessageBreak the package options
840   kernelglossredefs=false (to \MessageBreak restore the former definition of
841   \string\makeglossary) and \MessageBreak nomain (if the file extensions cause a
842   conflict)}}%
843 \makeglossaries
844 }
```

`\erride@glossary`

```
845 \newcommand*{\@gls@override@glossary}[1][main]{%
846   \GlossariesWarning{Use of \string\glossary\space with
847   glossaries.sty is deprecated. \MessageBreak Indexing should be performed
848   with the user level \MessageBreak commands, such as \string\gls\space or
849   \string\glsadd. If you need the \MessageBreak original definition of
850   \string\glossary\space use the package \MessageBreak options
851   kernelglossredefs=false (to restore the \MessageBreak former definition of
852   \string\glossary) and nomain (if the \MessageBreak file extensions cause a
853   conflict)}}%
854 \gls@glossary{#1}%
855 }
```

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`

```
856 \if@gls@docloaded
857 \else
858   \def\glossary{\@gls@override@glossary}
859 \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.


```

860 \define@choicekey{glossaries.sty}{kernelglossredefs}%
861 [\gls@debug@val\gls@debug@nr]{true,false,nowarn}[true]%
862 {%
863   \ifcase\gls@debug@nr\relax
864   \def\glossary{\@gls@override@glossary}%
865   \def\makeglossary{%
866     \GlossariesWarning{Use of \string\makeglossary\space with
867       glossaries.sty is deprecated. Use \string\makeglossaries\space
868       instead. If you need the original definition of
869       \string\makeglossary\space use the package options
870       kernelglossredefs=false (to prevent redefinition of
871       \string\makeglossary) and nomain (if the file extensions cause a
872       conflict)}}%
873   \makeglossaries
874   }%
875   \or
876   \let\glossary\gls@original@glossary
877   \let\makeglossary\gls@original@makeglossary
878   \or
879   \def\makeglossary{\makeglossaries}%
880   \renewcommand*{\@gls@override@glossary}[1][main]{%
881     \gls@glossary{##1}%
882   }%
883   \fi
884 }

```

symbols Create a “symbols” glossary type

```

885 \@gls@declareoption{symbols}{%
886   \let\@gls@do@symbolsdef\@gls@symbolsdef
887 }

```

Default is not to define the symbols glossary:

```

888 \newcommand*{\@gls@do@symbolsdef}{}

```

@gls@symbolsdef

```

889 \newcommand*{\@gls@symbolsdef}{%
890   \newglossary[slg]{symbols}{sls}{slo}{\glssymbolsgroupname}%
891   \newcommand*{\printsymbols}[1][ ]{\printglossary[type=symbols,##1]}%

```

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

```

892   \newcommand*{\gls@tr@set@symbols@toctitle}{%
893     \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
894   }%
895 }%

```

numbers Create a “symbols” glossary type

```

896 \@gls@declareoption{numbers}{%
897   \let\@gls@do@numbersdef\@gls@numbersdef
898 }

```

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

903   \newcommand*{\gls@tr@set@numbers@toctitle}{%
904     \translatelet{\glossarytoctitle}{Numbers (glossaries)}%
905   }%
906 }
```

index Create an “index” glossary type

```
907 \@gls@declareoption{index}{%
908   \ifx\@gls@do@indexdef\@empty
909     \let\@gls@do@indexdef\@gls@indexdef
910   \fi
911 }
```

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

```
912 \@gls@declareoption{noglossaryindex}{%
913   \let\@gls@do@indexdef\relax
914 }
```

Default is not to define index glossary:

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

\@gls@indexdef \indexname isn't set by glossaries.

```
916 \newcommand*{\@gls@indexdef}{%
917   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
918   \newcommand*{\printindex}[1][\printglossary[type=index,##1]]%
919   \newcommand*{\newterm}[2][\%
920     \newglossaryentry{##2}%
921     {type={index},name={##2},description={\nopostdesc},##1}}
922   \let\@gls@do@indexdef\relax
923 }
```

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

```
924 \@for\CurrentOption :=\@declaredoptions\do{%
925   \ifx\CurrentOption\@empty
926   \else
927     \@expandtwoargs
928     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
929   \ifin@
930     \@use@ption
```

```

931     \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
932   \fi
933 \fi
934 }

```

Now process options passed to the package:

```
935 \ProcessOptionsX
```

Load backward compatibility stuff:

```
936 \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:

```

937 \disable@keys{glossaries.sty}{compatible-2.07,%
938 xindy,xindygloss,xindynoglsnumbers,makeindex,%
939 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,%
940 nomain,noglossaryindex}

```

Now define `\setupglossaries`:

```

941 \newcommand*{\setupglossaries}[1]{%
942   \renewcommand*{\@gls@setacrstyle}{}%
943   \ifglsacrshortcuts
944     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
945   \else
946     \def\@gls@setupshortcuts{%
947       \ifglsacrshortcuts
948         \DefineAcronymSynonyms
949       \fi
950     }%
951   \fi
952   \glsacrshortcutsfalse
953   \let\@gls@do@numbersdef\relax
954   \let\@gls@do@symbolssdef\relax
955   \let\@gls@do@indexdef\relax
956   \let\@gls@do@acronymsdef\relax
957   \ifglsentrycounter
958     \let\@gls@doentrycounterdef\relax
959   \else
960     \let\@gls@doentrycounterdef\@gls@define@glossaryentrycounter
961   \fi
962   \ifglssubentrycounter
963     \let\@gls@dosubentrycounterdef\relax
964   \else
965     \let\@gls@dosubentrycounterdef\@gls@define@glossarysubentrycounter
966   \fi
967   \setkeys{glossaries.sty}{#1}%
968   \@gls@setacrstyle
969   \@gls@setupshortcuts
970   \@gls@do@acronymsdef
971   \@gls@do@numbersdef

```

```

972 \@gls@do@symbolssdef
973 \@gls@do@indexdef
974 \@gls@doentrycounterdef
975 \@gls@dosubentrycounterdef
976 }

```

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.⟨n⟩.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 `{⟨section-level⟩.⟨n⟩.0}` non-existent warning (e.g. `\gls[counter=chapter]{label}`).

```

977 \ifthenelse{\equal{\glscounter}{section}}{%
978 {%
979   \ifcsundef{chapter}{}%
980   {%
981     \let\@gls@old@chapter\@chapter
982     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
983       \ifcsundef{hyperdef}{}{\hyperdef{section}{\thesection}{}}}%
984   }%
985 }%
986 {}

```

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

```

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

```

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

```

988 \newcommand*{\@onlypremakeg}[1]{%
989   \ifx\@gls@onlypremakeg\@empty
990     \def\@gls@onlypremakeg{#1}%
991   \else
992     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
993     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
994   \fi
995 }

```

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

```

996 \newcommand*{\@disable@onlypremakeg}{%
997   \@for\@thiscs:=\@gls@onlypremakeg\do{%
998     \expandafter\@disable@premakecs\@thiscs%
999   }}

```

`\@disable@premakecs` Disables the given command.

```

1000 \newcommand*{\@disable@premakecs}[1]{%

```

```

1001 \def#1{\PackageError{glossaries}{\string#1\space may only be
1002 used before \string\makeglossaries}{You can't use
1003 \string#1\space after \string\makeglossaries}}%
1004 }

```

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`

```
1005 \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`

```
1006 \providecommand*\acronymname{Acronyms}
```

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

```

1007 \newcommand*\glstocctitle[1]{%
1008 \def\glossarytocctitle{\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`

```
1009 \providecommand*\entryname{Notation}
```

`\descriptionname`

```
1010 \providecommand*\descriptionname{Description}
```

`\symbolname`

```
1011 \providecommand*\symbolname{Symbol}
```

`\pagelistname`

```
1012 \providecommand*\pagelistname{Page List}
```

Labels for `makeindex`'s symbol and number groups:

`\symbolsgroupname`

```
1013 \providecommand*\glsymbolsgroupname{Symbols}
```

`\numbersgroupname`

```
1014 \providecommand*\glnumbersgroupname{Numbers}
```

`glspluralsuffix` The default plural is formed by appending `\glspluralsuffix` to the singular form.

```
1015 \newcommand*{\glspluralsuffix}{s}
```

`acrpluralsuffix` Default plural suffix for acronyms

```
1016 \newcommand*{\glsacrpluralsuffix}{\glspluralsuffix}
```

`acrpluralsuffix`

```
1017 \newcommand*{\glsupacrpluralsuffix}{\glstextup{\glsacrpluralsuffix}}
```

`\seename`

```
1018 \providecommand*{\seename}{see}
```

`\andname`

```
1019 \providecommand*{\andname}{\&}
```

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

`eGlossariesLang`

```
1020 \newcommand*{\RequireGlossariesLang}[1]{%
1021   \@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}%
1022 }
```

`sGlossariesLang`

```
1023 \newcommand*{\ProvidesGlossariesLang}[1]{%
1024   \ProvidesFile{glossaries-#1.ldf}%
1025 }
```

`ssarytocaptions` Does nothing if translator hasn't been loaded.

```
1026 \newcommand*{\addglossarytocaptions}[1]{}
```

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

```
1027 \ifglstranslate
```

Load tracklang

```
1028 \RequirePackage{tracklang}
```

Load translator if required.

```
1029 \@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.)

```
1030 \@ifpackageloaded{translator}
1031 {%
```

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.

```

1032   \ifboolexpr
1033   {
1034     test {\ifdefstring{\trans@languages}{English}}
1035     and not
1036     test {\ifdefstring{\bbl@loaded}{english}}
1037   }
1038   {%
1039     \let\glsifusetranslator\@secondoftwo
1040   }%
1041   {%
1042     \usedictionary{glossaries-dictionary}%
1043     \renewcommand*{\addglossarytocaptions}[1]{%
1044       \ifcsundef{captions#1}{}%
1045       {%
1046         \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
1047         \expandafter\toks@\expandafter{\@gls@tmp
1048           \renewcommand*{\glossaryname}{\translate{Glossary}}}%
1049       }%
1050       \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
1051     }%
1052   }%
1053 }%
1054 }%
1055 {}%

```

Check for tracked languages

```

1056 \AnyTrackedLanguages
1057 {%
1058   \ForEachTrackedDialect{\this@dialect}{%
1059     \IfTrackedLanguageFileExists{\this@dialect}%
1060     {glossaries-}% prefix
1061     {.ldf}%
1062     {%
1063       \RequireGlossariesLang{\CurrentTrackedTag}%
1064     }%
1065     {%
1066       \@gls@missinglang@warn\this@dialect\CurrentTrackedLanguage
1067     }%
1068   }%
1069 }%
1070 {}%

```

if using translator use translator interface.

```

1071 \glsifusetranslator
1072 {%
1073   \renewcommand*{\glssettoctitle}[1]{%

```

```

1074     \ifcsdef{gls@tr@set@#1@toctitle}%
1075     {%
1076     \csuse{gls@tr@set@#1@toctitle}%
1077     }%
1078     {%
1079     \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
1080     }%
1081     }%
1082     \renewcommand*{\glossaryname}{\translate{Glossary}}}%
1083     \renewcommand*{\acronymname}{\translate{Acronyms}}}%
1084     \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
1085     \renewcommand*{\descriptionname}{%
1086     \translate{Description (glossaries)}}}%
1087     \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
1088     \renewcommand*{\pagelistname}{%
1089     \translate{Page List (glossaries)}}}%
1090     \renewcommand*{\glssymbolsgroupname}{%
1091     \translate{Symbols (glossaries)}}}%
1092     \renewcommand*{\glsnumbersgroupname}{%
1093     \translate{Numbers (glossaries)}}}%
1094     }{}%
1095 \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.

```
1096 \DeclareRobustCommand*{\nopostdesc}{}

```

`\@nopostdesc` Suppress next description terminator.

```

1097 \newcommand*{\@nopostdesc}{%
1098   \let\org@glspostdescription\glspostdescription
1099   \def\glspostdescription{%
1100     \let\glspostdescription\org@glspostdescription}%
1101 }

```

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

```
1102 \newcommand*{\@no@post@desc}{\nopostdesc}

```

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

```
1103 \newcommand{\glspar}{\par}

```

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

```

1104 \newcommand{\setStyleFile}[1]{%
1105   \renewcommand*{\gls@istfilebase}{#1}%
1106   Just in case \istfilename has been modified.
1107   \ifglxsindy
1108     \def\istfilename{\gls@istfilebase.xdy}
1109   \else
1110     \def\istfilename{\gls@istfilebase.ist}

```



```
1110 \fi
1111 }
```

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

```
1112 \@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`

```
1113 \ifglsxindy
1114 \def\istfilename{\gls@istfilebase.xdy}
1115 \else
1116 \def\istfilename{\gls@istfilebase.ist}
1117 \fi
```

`gls@istfilebase`

```
1118 \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`

```
1119 \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`

```
1120 \newcommand*{\glscompositor}{.}
```

`lsSetCompositor` Sets the compositor.

```
1121 \newcommand*{\glsSetCompositor}[1]{%
1122 \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
1123 \@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 $\langle letter \rangle \langle compositor \rangle \langle number \rangle$. 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.

```
1124 \newcommand*{\@glsAlphacompositor}{\glscompositor}
```

AlphaCompositor Sets the alpha compositor.

```
1125 \ifglsxindy
1126   \newcommand*\glsSetAlphaCompositor[1]{%
1127     \renewcommand*\@glsAlphacompositor{#1}}
1128 \else
1129   \newcommand*\glsSetAlphaCompositor[1]{%
1130     \glsnxindywarning\glsSetAlphaCompositor}
1131 \fi
```

Can only be used before `\makeglossaries`

```
1132 \@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.

```
1133 \newcommand*{\gls@suffixF}{}
```

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

```
1134 \newcommand*{\glsSetSuffixF}[1]{%
1135   \renewcommand*{\gls@suffixF}{#1}}
```

Only has an effect when used before `\makeglossaries`

```
1136 \@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.

```
1137 \newcommand*{\gls@suffixFF}{}
```

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

```
1138 \newcommand*{\glsSetSuffixFF}[1]{%
1139   \renewcommand*{\gls@suffixFF}{#1}%
1140 }
```

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”.

```
1141 \ifcsundef{hyperlink}%
1142 {%
1143   \newcommand*\glsnumberformat[1]{#1}%
1144 }%
1145 {%
```

```

1146 \newcommand*{\glsnumberformat}[1]{\glshyphernumber{#1}}%
1147 }

```

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
1148 \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
1149 \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`.

```

\glossarypreamble
1150 \newcommand*{\glossarypreamble}{%
1151   \csuse{@glossarypreamble@currentglossary}%
1152 }

```

```

\glossarypreamble \setglossarypreamble[<type>]{<text>}

```

Code provided by Michael Pock.

```

1153 \newcommand{\setglossarypreamble}[2][\glsdefaultttype]{%
1154   \ifglossaryexists{#1}{%
1155     \csgdef{@glossarypreamble@#1}{#2}%
1156   }{%
1157     \GlossariesWarning{%
1158       Glossary ‘#1’ is not defined%
1159     }%
1160   }%
1161 }

```

The glossary postamble is given by `\glossarypostamble`. This is provided to allow the user to add something after the end of the `theglossary` 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`

```
1162 \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`.

```
1163 \newcommand*{\glossarysection}[2][\@gls@title]{%
1164   \def\@gls@title{#2}%
1165   \ifcsundef{phantomsection}%
1166   {%
1167     \@glossarysection{#1}{#2}%
1168   }%
1169   {%
1170     \p@glossarysection{#1}{#2}%
1171   }%

1172   \glsglossarymark{\glossarytoctitle}%
1173 }
```

`glsglossarymark`

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

```
1174 \ifcsundef{glossarymark}%
1175 {%
1176   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1177 }%
1178 {%
1179   \@ifclassloaded{memoir}
1180   {%
1181     \newcommand{\glsglossarymark}[1]{%
1182       \ifglsucmark
1183         \markboth{\memUHead{#1}}{\memUHead{#1}}%
1184       \else
1185         \markboth{#1}{#1}%
1186       \fi
1187     }
1188   }%
1189   {%
1190     \newcommand{\glsglossarymark}[1]{%
1191       \ifglsucmark
1192         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1193       \else
1194         \@mkboth{#1}{#1}%
1195       \fi
1196     }
1197   }%
1198 }
```

```

1196     }
1197   }
1198 }

```

`\glossarymark` Provided for backward compatibility:

```

1199 \providecommand{\glossarymark}[1]{%
1200   \ifglsucmark
1201     \mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1202   \else
1203     \mkboth{#1}{#1}%
1204   \fi
1205 }

```

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`

```

1206 \newcommand*{\setglossarysection}[1]{%
1207 \setkeys{glossaries.sty}{section=#1}}

```

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

`\glossarysection`

```

1208 \newcommand*{\@glossarysection}[2]{%
1209   \ifdefempty\@glossarysecstar
1210   {%
1211     \csname\@glossarysec\endcsname[#1]{#2}%
1212   }%
1213   {%
1214     \csname\@glossarysec\endcsname*{#2}%
1215     \gls@toc{#1}{\@glossarysec}%
1216   }%

```

Do automatic labelling if required

```

1217   \@glossaryseclabel
1218 }

```

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`

```

1219 \newcommand*{\@pglossarysection}[2]{%
1220   \glsclearpage
1221   \phantomsection
1222   \ifdefempty\@glossarysecstar
1223   {%

```

```

1224 \csname\@glossarysec\endcsname{#2}%
1225 }%
1226 {%
1227 \@gls@toc{#1}{\@glossarysec}%
1228 \csname\@glossarysec\endcsname*{#2}%
1229 }%

```

Do automatic labelling if required

```

1230 \@glossaryseclabel
1231 }

```

`\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.

```

1232 \newcommand*{\gls@doclearpage}{%
1233 \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1234 {%
1235 \ifcsundef{cleardoublepage}%
1236 {%
1237 \clearpage
1238 }%
1239 {%
1240 \ifcsdef{if@openright}%
1241 {%
1242 \if@openright
1243 \cleardoublepage
1244 \else
1245 \clearpage
1246 \fi
1247 }%
1248 {%
1249 \cleardoublepage
1250 }%
1251 }%
1252 }%
1253 {}%
1254 }

```

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

```

1255 \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`

```

1256 \newcommand*{\@gls@toc}[2]{%
1257 \ifglstoc

```

```

1258     \ifglslnumberline
1259         \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1260     \else
1261         \addcontentsline{toc}{#2}{#1}%
1262     \fi
1263 \fi
1264 }

```

1.4 Xindy

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

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

```

1265 \newcommand*{\glsnoxywarning}[1]{%
1266     \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1267 }

```

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

```

1268 \newcommand*{\glsnomakeindexwarning}[1]{%
1269     \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1270 }

```

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

```

1271 \ifglslxindy
1272     \edef\@xdyattributes{\string"default\string"}%
1273 \fi

```

dyattributelist Comma-separated list of attributes.

```

1274 \ifglslxindy
1275     \edef\@dyattributelist{}%
1276 \fi

```

\@xdylocref Define list of markup location references.

```

1277 \ifglslxindy
1278     \def\@xdylocref{}
1279 \fi

```

\@gls@ifinlist

```

1280 \newcommand*{\@gls@ifinlist}[4]{%
1281     \def\@do@ifinlist##1,#1,##2\end@do@ifinlist{%
1282         \def\@gls@listsuffix{##2}%
1283         \ifx\@gls@listsuffix\@empty
1284             #4%
1285         \else
1286             #3%

```

```

1287     \fi
1288   }%
1289   \@do@ifinlist,#2,#1,\end@do@ifinlist
1290 }

```

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.

```

1291 \ifglxsindy
1292   \newcommand*{\@xdycounters}{\@glscounter}
1293   \newcommand*\GlsAddXdyCounters[1]{%
1294     \@for\@gls@ctr:=#1\do{%
1295       \edef\@do@addcounter{%
1296         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1297         {%
1298           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1299             \noexpand\@gls@ctr}%
1300         }%
1301       }%
1302       \@do@addcounter
1303     }
1304   }

```

Only has an effect before `\writeist`:

```

1305   \@onlypremakeg\GlsAddXdyCounters
1306 \else
1307   \newcommand*\GlsAddXdyCounters[1]{%
1308     \glsnoxindywarning\GlsAddXdyAttribute
1309   }
1310 \fi

```

saddxdycounters Counters must all be identified before adding attributes.

```

1311 \newcommand*\@disabled@glssaddxdycounters{%
1312   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1313     can't be used after \string\GlsAddXdyAttribute}{Move all
1314     occurrences of \string\GlsAddXdyCounters\space before the first
1315     instance of \string\GlsAddXdyAttribute}%
1316 }

```

AddXdyAttribute Adds an attribute.

```

1317 \ifglxsindy
1318   \newcommand*\@glssaddxdyattribute[2]{%
1319     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1320       \string"#2#1\string"}%

```

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

Add to xindy attribute list

Add to xindy markup location.

```

1321 \expandafter\toks@\expandafter{\@xdylocref}%
1322 \edef\@xdylocref{\the\toks@ ^^J%
1323 (markup-locref
1324 :open \string"glstildechar n%
1325 \expandafter\string\csname glsX#2X#1\endcsname
1326 \string" ^^J
1327 :close \string"\string" ^^J
1328 :attr \string"#2#1\string")}%

```

Define associated attribute command $\text{\glsX}\langle counter \rangle X \langle attribute \rangle \{ \langle Hprefix \rangle \} \{ \langle n \rangle \}$

```

1329 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1330 \setentrycounter{##1}{#2}\csname #1\endcsname{##2}%
1331 }%
1332 }

```

High-level command:

```

1333 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

1334 \ifx\@xdyattributelist\@empty
1335 \edef\@xdyattributelist{#1}%
1336 \else
1337 \edef\@xdyattributelist{\@xdyattributelist,#1}%
1338 \fi

```

Iterate through all specified counters and add counter-dependent attributes:

```

1339 \@for\@this@counter:=\@xdycounters\do{%
1340 \protected@edef\gls@do@addxdyattribute{%
1341 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1342 }
1343 \gls@do@addxdyattribute
1344 }%

```

All occurrences of `\GlsAddXdyCounters` must be used before this command

```

1345 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1346 }

```

Only has an effect before `\writeist`:

```

1347 \@onlypremakeg\GlsAddXdyAttribute
1348 \else
1349 \newcommand*\GlsAddXdyAttribute[1]{%
1350 \glsnoxindywarning\GlsAddXdyAttribute}
1351 \fi

```

`definedattributes` Add known attributes for all defined counters

```

1352 \ifglsxindy
1353 \newcommand*\@gls@addpredefinedattributes{%
1354 \GlsAddXdyAttribute{glsnumberformat}
1355 \GlsAddXdyAttribute{textrm}
1356 \GlsAddXdyAttribute{textsf}

```

```

1357 \GlsAddXdyAttribute{texttt}
1358 \GlsAddXdyAttribute{textbf}
1359 \GlsAddXdyAttribute{textmd}
1360 \GlsAddXdyAttribute{textit}
1361 \GlsAddXdyAttribute{textup}
1362 \GlsAddXdyAttribute{textsl}
1363 \GlsAddXdyAttribute{textsc}
1364 \GlsAddXdyAttribute{emph}
1365 \GlsAddXdyAttribute{glshypernumber}
1366 \GlsAddXdyAttribute{hyperrrm}
1367 \GlsAddXdyAttribute{hypersf}
1368 \GlsAddXdyAttribute{hypertt}
1369 \GlsAddXdyAttribute{hyperbf}
1370 \GlsAddXdyAttribute{hypermd}
1371 \GlsAddXdyAttribute{hyperit}
1372 \GlsAddXdyAttribute{hyperup}
1373 \GlsAddXdyAttribute{hypersl}
1374 \GlsAddXdyAttribute{hypersc}
1375 \GlsAddXdyAttribute{hyperemph}

1376 \GlsAddXdyAttribute{glsglignore}
1377 }
1378 \else
1379 \let\@gls@addpredefinedattributes\relax
1380 \fi

```

dyuseralphabets List of additional alphabets

```
1381 \def\@xdyuseralphabets{}
```

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

```

1382 \ifglsxindy
1383 \newcommand*{\GlsAddXdyAlphabet}[2]{%
1384 \edef\@xdyuseralphabets{%
1385 \@xdyuseralphabets ^^J
1386 (define-alphabet "#1" (#2))}}
1387 \else
1388 \newcommand*{\GlsAddXdyAlphabet}[2]{%
1389 \glsnnoxindywarning\GlsAddXdyAlphabet}
1390 \fi

```

This code is only required for xindy:

```
1391 \ifglsxindy
```

dy@locationlist List of predefined location names.

```

1392 \newcommand*{\@gls@xdy@locationlist}{%
1393 roman-page-numbers,%
1394 Roman-page-numbers,%
1395 arabic-page-numbers,%

```

```

1396     alpha-page-numbers,%
1397     Alpha-page-numbers,%
1398     Appendix-page-numbers,%
1399     arabic-section-numbers%
1400 }

```

Each location class *<name>* has the format stored in `\@gls@xdy@Lclass@<name>`. Set up pre-defined 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.

```

1401 \protected@edef\@gls@roman{\@roman{0}\string"
1402     \string"roman-numbers-lowercase\string" :sep \string"}}%
1403 \@onelevel@sanitize\@gls@roman
1404 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1405     :sep \string"%}
1406 \@onelevel@sanitize\@tmp
1407 \ifx\@tmp\@gls@roman
1408     \expandafter
1409         \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1410             \string"roman-numbers-lowercase\string"%
1411         }%
1412 \else
1413     \expandafter
1414         \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1415             :sep \string"\@gls@roman\string"%
1416         }%
1417 \fi

```

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

```

1418 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1419     \string"roman-numbers-uppercase\string"%
1420 }%

```

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

```

1421 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1422     \string"arabic-numbers\string"%
1423 }%

```

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

```

1424 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1425     \string"alpha\string"%
1426 }%

```

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

```

1427 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1428     \string"ALPHA\string"%
1429 }%

```

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

```

1430 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1431   \string"ALPHA\string"
1432   :sep \string"\@glsAlphacompositor\string"
1433   \string"arabic-numbers\string"%
1434 }

```

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

```

1435 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1436   \string"arabic-numbers\string"
1437   :sep \string"\glscompositor\string"
1438   \string"arabic-numbers\string"%
1439 }%

```

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

```

1440 \def\@xdyuserlocationdefs{}

```

erlocationnames List of additional user location names

```

1441 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

```

1442 \fi

```

xdycrossrefhook Hook used after writing cross-reference class information.

```

1443 \ifglsxindy
1444 \newcommand\@xdycrossrefhook{}
1445 \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.)

```

1446 \ifglsxindy
1447 \newcommand*\GlsAddXdyLocation[3][{}]{%
1448   \def\@gls@tmp{#1}%
1449   \ifx\@gls@tmp\@empty
1450     \edef\@xdyuserlocationdefs{%
1451       \@xdyuserlocationdefs ^^J%
1452       (define-location-class \string"#2\string"^^J\space\space
1453       \space(:sep \string"{}\glsopenbrace\string" #3
1454       :sep \string"\glsclosebrace\string"))
1455     }%
1456   \else
1457     \edef\@xdyuserlocationdefs{%
1458       \@xdyuserlocationdefs ^^J%
1459       (define-location-class \string"#2\string"^^J\space\space
1460       \space(:sep "\glsopenbrace"
1461       #1

```

```

1462             :sep "\glsclosebrace\glsopenbrace" #3
1463             :sep "\glsclosebrace"))
1464     }%
1465 \fi

1466 \edef\@xdyuserlocationnames{%
1467     \@xdyuserlocationnames^^J\space\space\space
1468     \string"#2\string"}%
1469 }

```

Only has an effect before \writeist:

```

1470 \@onlypremakeg\GlsAddXdyLocation
1471 \else
1472 \newcommand*\GlsAddXdyLocation[2]{%
1473     \glsnoxywarning\GlsAddXdyLocation}
1474 \fi

```

ationclassorder Define location class order

```

1475 \ifglxindy
1476 \def\@xdylocationclassorder{^^J\space\space\space
1477     \string"roman-page-numbers\string"^^J\space\space\space
1478     \string"arabic-page-numbers\string"^^J\space\space\space
1479     \string"arabic-section-numbers\string"^^J\space\space\space
1480     \string"alpha-page-numbers\string"^^J\space\space\space
1481     \string"Roman-page-numbers\string"^^J\space\space\space
1482     \string"Alpha-page-numbers\string"^^J\space\space\space
1483     \string"Appendix-page-numbers\string"
1484     \@xdyuserlocationnames^^J\space\space\space
1485     \string"see\string"
1486 }
1487 \fi

```

Change the location order.

ationClassOrder

```

1488 \ifglxindy
1489 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1490     \def\@xdylocationclassorder{#1}}
1491 \else
1492 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1493     \glsnoxywarning\GlsSetXdyLocationClassOrder}
1494 \fi

```

\@xdysortrules Define sort rules

```

1495 \ifglxindy
1496 \def\@xdysortrules{}
1497 \fi

```

\GlsAddSortRule Add a sort rule

```

1498 \ifglxindy
1499   \newcommand*\GlsAddSortRule[2]{%
1500     \expandafter\toks@\expandafter{\@xdysortrules}%
1501     \protected@edef\@xdysortrules{\the\toks@ ^^J
1502       (sort-rule \string"#1\string" \string"#2\string"))}%
1503   }
1504 \else
1505   \newcommand*\GlsAddSortRule[2]{%
1506     \glsnxindywarning\GlsAddSortRule}
1507 \fi

```

yrequiredstyles Define list of required styles (this should be a comma-separated list of xindy styles)

```

1508 \ifglxindy
1509   \def\@xdyrequiredstyles{tex}
1510 \fi

```

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

```

1511 \ifglxindy
1512   \newcommand*\GlsAddXdyStyle[1]{%
1513     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1514 \else
1515   \newcommand*\GlsAddXdyStyle[1]{%
1516     \glsnxindywarning\GlsAddXdyStyle}
1517 \fi

```

GlsSetXdyStyles Reset the list of required styles

```

1518 \ifglxindy
1519   \newcommand*\GlsSetXdyStyles[1]{%
1520     \edef\@xdyrequiredstyles{#1}}
1521 \else
1522   \newcommand*\GlsSetXdyStyles[1]{%
1523     \glsnxindywarning\GlsSetXdyStyles}
1524 \fi

```

indrootlanguage 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.

```

1525 \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.

```

1526 \def\@xdylanguage#1#2{}

```

sSetXdyLanguage 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.

```

1527 \ifglxindy
1528   \newcommand*\GlsSetXdyLanguage[2][\glsdefaultttype]{%
1529     \ifglossaryexists{#1}{%
1530       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1531     }{%
1532       \PackageError{glossaries}{Can't set language type for
1533         glossary type '#1' --- no such glossary}{%
1534         You have specified a glossary type that doesn't exist}}
1535 \else
1536   \newcommand*\GlsSetXdyLanguage[2][{}]{%
1537     \glsnoxywarning\GlsSetXdyLanguage}
1538 \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.

```

1539 \def\@gls@codepage#1#2{}

```

`sSetXdyCodePage` Define command to set the code page.

```

1540 \ifglxindy
1541   \newcommand*\GlsSetXdyCodePage[1]{%
1542     \renewcommand*\@gls@codepage{#1}%
1543   }

```

Suggested by egreg:

```

1544   \AtBeginDocument{%
1545     \ifx\gls@codepage\@empty
1546       \@ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{}%
1547     \fi
1548   }
1549 \else
1550   \newcommand*\GlsSetXdyCodePage[1]{%
1551     \glsnoxywarning\GlsSetXdyCodePage}
1552 \fi

```

`xdylettergroups` Store letter group definitions.

```

1553 \ifglxindy
1554   \ifglx@xindy@glxnumbers
1555     \def\@xdylettergroups{(define-letter-group
1556       \string"glxnumbers\string"^^J\space\space\space
1557       :prefixes (\string"0\string" \string"1\string"
1558       \string"2\string" \string"3\string" \string"4\string"
1559       \string"5\string" \string"6\string" \string"7\string"
1560       \string"8\string" \string"9\string")^^J\space\space\space
1561       \@xdynumbergrouporder)}
1562   \else
1563     \def\@xdylettergroups{}
1564   \fi
1565 \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.

```
1566 \newcommand*\GlsAddLetterGroup[2]{%
1567   \expandafter\toks@\expandafter{\@xdylettergroups}%
1568   \protected@edef\@xdylettergroups{\the\toks@^^J%
1569   (define-letter-group \string"#1\string"^^J\space\space\space#2)}}%
1570 }
```

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.

```
1571 \newcommand*\forallglossaries[3][\@glo@types]{%
1572   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1573 }
```

`\forallacronyms`

```
1574 \newcommand*\forallacronyms[2]{%
1575   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1576 }
```

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

```
\forallglsentries[<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.

```
1577 \newcommand*\forallglsentries[3][\glsdefaulttype]{%
1578   \edef\@glo@list{\csname glist@#1\endcsname}%
1579   \@for#2:=\@glo@list\do
1580   {%
1581     \ifdefempty{#2}{#3}%
1582   }%
1583 }
```

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

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

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


```

1584 \newcommand*{\forallglsentries}[3][\@glo@types]{%
1585   \expandafter\forallglossaries\expandafter[#1]{\@this@glo@}%
1586   {%
1587     \forallglsentries[\@this@glo@]{#2}{#3}%
1588   }%
1589 }

```

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

```
\ifglossaryexists{<type>}{<true-text>}{<false-text>}
```

where *<type>* is the glossary's label.

```

1590 \newcommand{\ifglossaryexists}[3]{%
1591   \ifcsundef{@glo@#1@out}{#3}{#2}%
1592 }

```

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 `\glsentrytext` 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*{\glsdetoklabel}[1]{\scantokens{#1\noexpand}}
```

(Note, don't use `\detokenize` or it will cause commands like `\glsaddall` to fail.) Since re-defining `\glsdetoklabel` 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.

`\glsdetoklabel`

```
1593 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

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

where *<label>* is the entry's label.

```

1594 \newcommand{\ifglsentryexists}[3]{%
1595   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1596 }

```

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

```
\ifglsused{<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>*.

```
1597 \newcommand*{\ifglsused}[3]{%
```

```

1598 \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1599 }

```

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

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

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

```

1600 \newcommand{\glsdoifexists}[2]{%
1601   \ifglentryexists{#1}{#2}{%
1602     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1603       has not been defined}{You need to define a glossary entry before you
1604       can use it.}}%
1605 }

```

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

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

```

1606 \newcommand{\glsdoifnoexists}[2]{%
1607   \ifglentryexists{#1}{%
1608     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1609     been defined}{}}{#2}%
1610 }

```

```
\doifexistsorwarn \glsdoifexistsorwarn{<label>}{<code>}
```

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

```

1611 \newcommand{\glsdoifexistsorwarn}[2]{%
1612   \ifglentryexists{#1}{#2}{%
1613     \GlossariesWarning{Glossary entry '\glsdetoklabel{#1}'
1614       has not been defined}%
1615   }%
1616 }

```

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

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

```

1617 \newcommand{\glsdoifexistsordo}[3]{%
1618   \ifglentryexists{#1}{#2}{%
1619     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1620       has not been defined}{You need to define a glossary entry before you
1621       can use it.}}%
1622   #3%

```

```

1623 }%
1624 }

```

sarynoexistsordo

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

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

```

1625 \newcommand{\doifglossarynoexistsordo}[3]{%
1626   \ifglossaryexists{#1}%
1627   {%
1628     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{%
1629       #3}%
1630   }%
1631   {#2}%
1632 }

```

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

```

1633 \newcommand{\ifglshaschildren}[3]{%
1634   \glstoifexists{#1}%
1635   {%
1636     \def\do@glshaschildren{#3}%
1637     \edef\@gls@thislabel{\glstoiflabel{#1}}%
1638     \expandafter\for@glstentries\expandafter
1639     [\csname glo@\@gls@thislabel @type\endcsname]
1640     {\glo@label}%
1641     {%
1642       \letcs\glo@parent{glo@\glo@label @parent}%
1643       \ifdefequal\@gls@thislabel\glo@parent
1644       {%
1645         \def\do@glshaschildren{#2}%
1646         \@endfortrue
1647       }%
1648       {}%
1649     }%
1650     \do@glshaschildren
1651   }%
1652 }

```

\ifglshasparent

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

```

1653 \newcommand{\ifglshasparent}[3]{%
1654   \glstoifexists{#1}%
1655   {%
1656     \ifcsemt{glo@\glstoiflabel{#1}@parent}{#3}{#2}%
1657   }%
1658 }

```

```

\ifglshasdesc \ifglshasdesc{<label>}{<true part>}{<false part>}
1659 \newcommand*{\ifglshasdesc}[3]{%
1660   \ifcsequal{glo@glstdetoklabel{#1}@desc}%
1661   {#3}%
1662   {#2}%
1663 }

sdescsuppressed \ifglstdescsuppressed{<label>}{<true part>}{<false part>} Does <true part> if the descrip-
tion is just \nopostdesc otherwise does <false part>.
1664 \newcommand*{\ifglstdescsuppressed}[3]{%
1665   \ifcsequal{glo@glstdetoklabel{#1}@desc}{@no@post@desc}%
1666   {#2}%
1667   {#3}%
1668 }

\ifglshassymbol \ifglshassymbol{<label>}{<true part>}{<false part>}
1669 \newcommand*{\ifglshassymbol}[3]{%
1670   \letcs{\@glo@symbol}{glo@glstdetoklabel{#1}@symbol}%
1671   \ifdefempty\@glo@symbol
1672   {#3}%
1673   {%
1674     \ifdefequal\@glo@symbol\@gls@default@value
1675     {#3}%
1676     {#2}%
1677   }%
1678 }

\ifglshaslong \ifglshaslong{<label>}{<true part>}{<false part>}
1679 \newcommand*{\ifglshaslong}[3]{%
1680   \letcs{\@glo@long}{glo@glstdetoklabel{#1}@long}%
1681   \ifdefempty\@glo@long
1682   {#3}%
1683   {%
1684     \ifdefequal\@glo@long\@gls@default@value
1685     {#3}%
1686     {#2}%
1687   }%
1688 }

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1689 \newcommand*{\ifglshasshort}[3]{%
1690   \letcs{\@glo@short}{glo@glstdetoklabel{#1}@short}%
1691   \ifdefempty\@glo@short
1692   {#3}%
1693   {%
1694     \ifdefequal\@glo@short\@gls@default@value
1695     {#3}%
1696     {#2}%

```

```

1697 }%
1698 }

```

```

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

```

```

1699 \newcommand*{\ifglshasfield}[4]{%
1700   \glstoifexists{#2}%
1701   {%
1702     \letcs{\@glo@thisvalue}{glo\glstetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1703   \ifdef\@glo@thisvalue
1704   {%

```

Is defined, so now check if empty.

```

1705     \ifdefempty\@glo@thisvalue
1706     {%

```

Is empty, so doesn't have field set.

```

1707       #4%
1708     }%
1709   {%

```

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

```

1710     \ifdefequal\@glo@thisvalue\@gls@default@value
1711     {%

```

Value is set to the default value.

```

1712       #4%
1713     }%
1714   {%

```

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

```

1715     \let\glscurrentfieldvalue\@glo@thisvalue
1716     #3%
1717   }%
1718 }%
1719 }%
1720 {%

```

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

```

1721   \@gls@fetchfield{\@gls@thisfield}{#1}%

```

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

```

1722   \ifdef\@gls@thisfield
1723   {%

```

Is defined, so now check if empty.

```

1724     \letcs{\@glo@thisvalue}{glo\glstetoklabel{#2}@\@gls@thisfield}%
1725     \ifdefempty\@glo@thisvalue
1726     {%

```

Is empty so field hasn't been set.

```
1727         #4%
1728     }%
1729     {%
```

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

```
1730     \ifdefequal\@glo@thisvalue\@gls@default@value
1731     {%
```

Value is set to the default value.

```
1732         #4%
1733     }%
1734     {%
```

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

```
1735     \let\glscurrentfieldvalue\@glo@thisvalue
1736     #3%
1737     }%
1738     }%
1739     }%
1740     {%
```

Not defined.

```
1741     \GlossariesWarning{Unknown entry field '#1'}%
1742     #4%
1743     }%
1744     }%
1745     }%
1746 }
```

urrentfieldvalue

```
1747 \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

```
1748 \newcommand*{\@glo@types}{,}
```

ide@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.

```
1749 \newcommand*\@gls@provide@newglossary{%
1750     \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%

```

Only need to do this once.

```
1751 \let\@gls@provide@newglossary\relax
1752 }
```

`\defglsentryfmt` Allow different glossaries to have different display styles.

```
1753 \newcommand*{\defglsentryfmt}[2][\glsdefaulttype]{%
1754   \csgdef{gls@#1@entryfmt}{#2}%
1755 }
```

`\gls@doentryfmt`

```
1756 \newcommand*{\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}
```

`\ls@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.

```
1757 \newcommand*{\@gls@forbidtextext}[1]{%
1758   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1759               or test {\ifdefstring{#1}{TEX}}}
1760   {%
1761     \def#1{nottex}%
1762     \PackageError{glossaries}%
1763       {Forbidden '.tex' extension replaced with '.nottex'}%
1764       {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1765         Don't use '.tex' as an extension for a temporary file.}%
1766   }%
1767   {%
1768   }%
1769 }
```

`\gls@gobbleopt` Discard optional argument.

```
1770 \newcommand*{\gls@gobbleopt}{\new@ifnextchar[{\@gls@gobbleopt}{} }
1771 \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`

```
1772 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}
```

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

```

1773 \newcommand*{\s@newglossary}[2]{%
1774   \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1775 }

```

`\ns@newglossary` Define the unstarred version.

```

1776 \newcommand*{\ns@newglossary}[5][glg]{%
1777   \doifglossarynoexistsordo{#2}%
1778   {%

```

Check if default has been set

```

1779   \ifundef\glsdefaultttype
1780   {%
1781     \gdef\glsdefaultttype{#2}%
1782   }{}%

```

Add this to the list of glossary types:

```

1783   \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.

```

1784   \expandafter\gdef\csname glolist@#2\endcsname{,}%

```

Store the file extensions:

```

1785   \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1786   \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1787   \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1788   \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
1789   \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
1790   \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname

```

Store the title:

```

1791   \expandafter\def\csname @glotype@#2@title\endcsname{#5}%

```

```

1792   \@gls@provide@newglossary
1793   \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.

```

1794   \ifcsundef{gls@#2@entryfmt}%
1795   {%
1796     \defglsentryfmt[#2]{\glsentryfmt}%
1797   }%
1798   {}%

```

Define sort counter if required:

```

1799   \@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.)

```
1800 \ifnextchar[{\@gls@setcounter{#2}}{%
1801   {\@gls@setcounter{#2}[\glscounter]}}%
1802 }%
1803 {%
1804   \gls@gobbleopt
1805 }%
1806 }
```

`\altnewglossary`

```
1807 \newcommand*{\altnewglossary}[3]{%
1808   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1809 }
```

Only define new glossaries in the preamble:

```
1810 \@onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1811 \@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`

```
1812 \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`

```
1813 \def\@gls@setcounter#1[#2]{%
1814   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%
```

Add counter to xindy list, if not already added:

```
1815   \ifglsxindy
1816     \GlsAddXdyCounters{#2}%
1817   \fi
1818 }
```

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

`@gls@getcounter`

```
1819 \newcommand*{\@gls@getcounter}[1]{%
1820   \csname @glotype@#1@counter\endcsname
1821 }
```

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

```
1822 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1823 \@gls@do@acronymsdef
```

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

```
1824 \@gls@do@symbolsdef
```

```
1825 \@gls@do@numbersdef
```

```
1826 \@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.

```
1827 \newcommand*{\newignoredglossary}[1]{%
1828   \ifdefempty\@ignored@glossaries
1829   {%
1830     \edef\@ignored@glossaries{#1}%
1831   }%
1832   {%
1833     \eappto\@ignored@glossaries{,#1}%
1834   }%
1835   \csgdef{glolist@#1}{,}%
1836   \ifcsundef{gls@#1@entryfmt}%
1837   {%
1838     \defglsentryfmt[#1]{\glsentryfmt}%
1839   }%
1840   {}%
1841   \ifdefempty\@gls@nohyperlist
1842   {%
1843     \renewcommand*{\@gls@nohyperlist}{#1}%
1844   }%
1845   {%
1846     \eappto\@gls@nohyperlist{,#1}%
1847   }%
1848 }
```

ignored@glossaries List of ignored glossaries.

```
1849 \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.

```
1850 \newcommand*{\ifignoredglossary}[3]{%
1851   \edef\@gls@igtype{#1}%
1852   \expandafter\DTLifinlist\expandafter
1853   {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1854 }
```

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.

```
1855 \define@key{glossentry}{name}{%  
1856 \def\@glo@name{#1}%  
1857 }
```

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`.

```
1858 \define@key{glossentry}{description}{%  
1859 \def\@glo@desc{#1}%  
1860 }
```

descriptionplural

```
1861 \define@key{glossentry}{descriptionplural}{%  
1862 \def\@glo@descplural{#1}%  
1863 }
```

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 `\langle name \rangle \langle description \rangle`.

```
1864 \define@key{glossentry}{sort}{%  
1865 \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.

```
1866 \define@key{glossentry}{text}{%  
1867 \def\@glo@text{#1}%  
1868 }
```

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.

```
1869 \define@key{glossentry}{plural}{%  
1870 \def\@glo@plural{#1}%  
1871 }
```

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.

```
1872 \define@key{glossentry}{first}{%
1873 \def\@glo@first{#1}%
1874 }
```

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.

```
1875 \define@key{glossentry}{firstplural}{%
1876 \def\@glo@firstplural{#1}%
1877 }
```

s@default@value

```
1878 \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).

```
1879 \define@key{glossentry}{symbol}{%
1880 \def\@glo@symbol{#1}%
1881 }
```

symbolplural

```
1882 \define@key{glossentry}{symbolplural}{%
1883 \def\@glo@symbolplural{#1}%
1884 }
```

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

```
1885 \define@key{glossentry}{type}{%
1886 \def\@glo@type{#1}}
```

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

```
1887 \define@key{glossentry}{counter}{%
1888   \ifcsundef{c@#1}%
1889   {%
1890     \PackageError{glossaries}%
1891     {There is no counter called ‘#1’}%
1892     {%
1893       The counter key should have the name of a valid counter
1894       as its value%
1895     }%
1896   }%
```

```

1897  {%
1898    \def\@glo@counter{#1}%
1899  }%
1900 }

```

see The see key specifies a list of cross-references

```

1901 \define@key{glossentry}{see}{%
1902   \gls@set@xr@key{see}{\@glo@see}{#1}%
1903 }

```

<code>\gls@set@xr@key</code>	<code>\gls@set@xr@key{<key name>}{<cs>}{<value>}</code>
------------------------------	---

Assign a cross-reference key.

```

1904 \newcommand*{\gls@set@xr@key}[3]{%
1905   \renewcommand*{\gls@xr@key}{#1}%
1906   \gls@checkseeallowed
1907   \def#2{#3}%
1908   \@glo@seeautonumberlist
1909 }

```

`\gls@xr@key`

```

1910 \newcommand*{\gls@xr@key}{see}

```

`checkseeallowed`

```

1911 \newcommand*{\gls@checkseeallowed}{%
1912   \@gls@see@noindex
1913 }

```

`ed@preambleonly`

```

1914 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1915   \GlossariesWarning{glossaries}%
1916   {'\gls@xr@key' key doesn't have any effect when used in the document
1917     environment. Move the definition to the preamble
1918     after \string\makeglossaries\space
1919     or \string\makenoidxglossaries}%
1920 }

```

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

```

1921 \define@key{glossentry}{parent}{%
1922 \def\@glo@parent{#1}}

```

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

```

1923 \define@choicekey{glossentry}{nonumberlist}%
1924   [\gls@nonumberlist@val\gls@nonumberlist@nr]{true,false}[true]%
1925 {%
1926   \ifcase\gls@nonumberlist@nr\relax
1927     \def\@glo@prefix{\glsnonextpages}%

```

```

1928 \gls@savenonumberlist{true}%
1929 \else
1930 \def\glo@prefix{\glsnextpages}%
1931 \gls@savenonumberlist{false}%
1932 \fi
1933 }

```

`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.

```

1934 \newcommand*{\@gls@savenonumberlist}[1]{%

```

`initnonumberlist`

```

1935 \newcommand*{\@gls@initnonumberlist}{%

```

`initnonumberlist`

```

1936 \newcommand*{\@gls@storenonumberlist}[1]{%

```

`savenonumberlist` Allow the `nonumberlist` value to be saved.

```

1937 \newcommand*{\@gls@enablesavenonumberlist}{%
1938 \renewcommand*{\@gls@initnonumberlist}{%
1939 \undef\glo@nonumberlist
1940 }%
1941 \renewcommand*{\@gls@savenonumberlist}[1]{%
1942 \def\glo@nonumberlist{##1}%
1943 }%
1944 \renewcommand*{\@gls@storenonumberlist}[1]{%
1945 \ifdef\glo@nonumberlist
1946 {%
1947 \cslet{glo@glstetoklabel{##1}@nonumberlist}{\glo@nonumberlist}%
1948 }%
1949 }%
1950 }%
1951 \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1952 }

```

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

`user1`

```

1953 \define@key{glossentry}{user1}{%
1954 \def\glo@useri{#1}%
1955 }

```

`user2`

```

1956 \define@key{glossentry}{user2}{%
1957 \def\glo@userii{#1}%
1958 }

```

user3

```
1959 \define@key{glossentry}{user3}{%
1960   \def\@glo@useriii{#1}%
1961 }
```

user4

```
1962 \define@key{glossentry}{user4}{%
1963   \def\@glo@useriv{#1}%
1964 }
```

user5

```
1965 \define@key{glossentry}{user5}{%
1966   \def\@glo@userv{#1}%
1967 }
```

user6

```
1968 \define@key{glossentry}{user6}{%
1969   \def\@glo@uservi{#1}%
1970 }
```

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.

```
1971 \define@key{glossentry}{short}{%
1972   \def\@glo@short{#1}%
1973 }
```

shortplural This key is provided for use by \newacronym.

```
1974 \define@key{glossentry}{shortplural}{%
1975   \def\@glo@shortpl{#1}%
1976 }
```

long This key is provided for use by \newacronym.

```
1977 \define@key{glossentry}{long}{%
1978   \def\@glo@long{#1}%
1979 }
```

longplural This key is provided for use by \newacronym.

```
1980 \define@key{glossentry}{longplural}{%
1981   \def\@glo@longpl{#1}%
1982 }
```

\@glsnname Define command to generate error if name key is missing.

```
1983 \newcommand*{\@glsnname}{%
1984   \PackageError{glossaries}{name key required in
1985     \string\newglossaryentry\space for entry '@glo@label'}{You
1986     haven't specified the entry name}}
```

`\@glsnodesc` Define command to generate error if description key is missing.

```

1987 \newcommand*\@glsnodesc{%
1988   \PackageError{glossaries}
1989   {%
1990     description key required in \string\newglossaryentry\space
1991     for entry '\@glo@label'%
1992   }%
1993   {%
1994     You haven't specified the entry description%
1995   }%
1996 }%
```

`lsdefaultplural` Now obsolete. Don't use.

```

1997 \newcommand*\@glsdefaultplural{}
```

`issingnumberlist` Define a command to generate warning when numberlist not set.

```

1998 \newcommand*\@gls@missingnumberlist}[1]{%
1999   ??%
2000   \ifglssavenumberlist
2001     \GlossariesWarning{Missing number list for entry '#1'.
2002       Maybe makeglossaries + rerun required}%
2003   \else
2004     \PackageError{glossaries}%
2005     {Package option 'savenumberlist=true' required}%
2006     {%
2007       You must use the 'savenumberlist' package option
2008       to reference location lists.%
2009     }%
2010   \fi
2011 }
```

`@glsdefaultsort` Define command to set default sort.

```

2012 \newcommand*\@glsdefaultsort{\@glo@name}
```

`\gls@level` Register to increment entry levels.

```

2013 \newcount\gls@level
```

`@noexpand@field`

```

2014 \newcommand{\@@gls@noexpand@field}[3]{%
2015   \expandafter\global\expandafter
2016   \let\csname glo@#1@#2\endcsname#3%
2017 }
```

`noexpand@fields`

```

2018 \newcommand{\@gls@noexpand@fields}[4]{%
2019   \ifcsdef{gls@assign@#3@field}
2020   {%
2021     \ifdefequal{#4}{\@gls@default@value}%

```



```

2022    {%
2023        \edef\@gls@value{\expandonce{#1}}%
2024        \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
2025    }%
2026    {%
2027        \csuse{gls@assign@#3@field}{#2}{#4}%
2028    }%
2029 }%
2030 {%
2031     \ifdefequal{#4}{\@gls@default@value}%
2032     {%
2033         \edef\@gls@value{\expandonce{#1}}%
2034         \@gls@noexpand@field{#2}{#3}{\@gls@value}%
2035     }%
2036     {%
2037         \@gls@noexpand@field{#2}{#3}{#4}%
2038     }%
2039 }%
2040 }

```

ls@expand@field

```

2041 \newcommand{\@gls@expand@field}[3]{%
2042     \expandafter
2043     \protected@xdef\csname glo@#1@#2\endcsname{#3}%
2044 }

```

s@expand@fields

```

2045 \newcommand{\@gls@expand@fields}[4]{%
2046     \ifcsdef{gls@assign@#3@field}
2047     {%
2048         \ifdefequal{#4}{\@gls@default@value}%
2049         {%
2050             \edef\@gls@value{\expandonce{#1}}%
2051             \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
2052         }%
2053         {%
2054             \expandafter\@gls@startswitexpandonce#4\relax\relax\gls@endcheck
2055             {%
2056                 \@gls@expand@field{#2}{#3}{#4}%
2057             }%
2058             {%
2059                 \csuse{gls@assign@#3@field}{#2}{#4}%
2060             }%
2061         }%
2062     }%
2063     {%
2064         \ifdefequal{#4}{\@gls@default@value}%
2065         {%

```

```

2066      \@gls@expand@field{#2}{#3}{#1}%
2067    }%
2068    {%
2069      \@gls@expand@field{#2}{#3}{#4}%
2070    }%
2071  }%
2072 }

```

switexpandonce

```

2073 \def\@gls@expandonce{\expandonce}
2074 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
2075   \def\@gls@tmp{#1}%
2076   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
2077 }

```

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.

```
2078 \let\gls@assign@field\@gls@expand@fields
```

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

```

2079 \newcommand*{\glsexpandfields}{%
2080   \let\gls@assign@field\@gls@expand@fields
2081 }

```

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

```

2082 \newcommand*{\glsnoexpandfields}{%
2083   \let\gls@assign@field\@gls@noexpand@fields
2084 }

```

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

```
2085 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```

2086   \glsdoifnoexists{#1}%
2087   {%
2088     \gls@defglossaryentry{#1}{#2}%
2089   }%
2090 }

```

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.

```

2091 \newcommand*{\gls@defdocnewglossaryentry}{%
2092   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
2093   \let\newglossaryentry\new@glossaryentry
2094 }

```

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

```

2095 \newrobustcmd{\provideglossaryentry}[2]{%
2096   \ifglstryexists{#1}%
2097   {}%
2098   {%
2099     \gls@defglossaryentry{#1}{#2}%
2100   }%
2101 }
2102 \@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 \LaTeX run. This means that any glossaries at the start of the document can access the entry information.

```

2103 \newrobustcmd{\new@glossaryentry}[2]{%
2104   \ifundef\@gls@deffile
2105   {%
2106     \global\newwrite\@gls@deffile
2107     \immediate\openout\@gls@deffile=\jobname.glsdefs
2108   }%
2109   {}%
2110   \ifglstryexists{#1}{}%
2111   {%
2112     \gls@defglossaryentry{#1}{#2}%
2113   }%
2114   \@gls@writedef{#1}%
2115 }

```

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`.

```

2116 \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.

```

2117 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}

```

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

```

2118 \newcommand*{\gls@begindocdefs}{%
2119   \@gls@enablesavenonumberlist
2120   \edef\@gls@restoreat{\noexpand\catcode'\noexpand\@=\number\catcode'\@relax}%
2121   \makeatletter
2122   \InputIfFileExists{\jobname.glsdefs}{\@gls@restoreat}{\@gls@restoreat}
2123 }

```

```

2124 \undef\@gls@restoreat
2125 \gls@defdocnewglossaryentry
2126 }

```

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

```

2127 \newcommand*{\@gls@writedef}[1]{%
2128   \immediate\write\@gls@deffile
2129   {%
2130     \string\ifglsentryexists{#1}{}\glspercentchar^^J%
2131     \expandafter\@gobble\string\{\glspercentchar^^J%
2132     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
2133     \expandafter\@gobble\string\{\glspercentchar%
2134   }%

  Write key value information:

2135   \@for\@gls@map:=\@gls@keymap\do
2136   {%
2137     \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
2138     \ifdef\glo@value
2139     {%
2140       \@onelevel@sanitize\glo@value
2141       \immediate\write\@gls@deffile
2142       {%
2143         \expandafter\@firstoftwo\@gls@map
2144         =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
2145         \glspercentchar
2146       }%
2147     }%
2148   }%
2149 }%

```

Provide hook:

```

2150 \gls@writedefhook
2151 \immediate\write\@gls@deffile
2152 {%
2153   \glspercentchar^^J%
2154   \expandafter\@gobble\string\}\glspercentchar^^J%
2155   \expandafter\@gobble\string\}\glspercentchar%
2156 }%
2157 }

```

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

```

2158 \newcommand*{\@gls@keymap}{%
2159   {name}{name},%
2160   {sort}{sortvalue},% unescaped sort value
2161   {type}{type},%
2162   {first}{first},%
2163   {firstplural}{firstpl},%
2164   {text}{text},%

```

```

2165 {plural}{plural},%
2166 {description}{desc},%
2167 {descriptionplural}{descplural},%
2168 {symbol}{symbol},%
2169 {symbolplural}{symbolplural},%
2170 {user1}{useri},%
2171 {user2}{userii},%
2172 {user3}{useriii},%
2173 {user4}{useriv},%
2174 {user5}{userv},%
2175 {user6}{uservi},%
2176 {long}{long},%
2177 {longplural}{longpl},%
2178 {short}{short},%
2179 {shortplural}{shortpl},%
2180 {counter}{counter},%
2181 {parent}{parent}%
2182 }

```

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

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

```
2183 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
2184 \edef\@gls@thisval{#2}%
```

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

```

2185 \@for\@gls@map:=\@gls@keymap\do{%
2186 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2187 \ifdefequal{\@this@key}{\@gls@thisval}%
2188 {%

```

Found it.

```
2189 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```

2190 \endfortrue
2191 }%
2192 {}%
2193 }%
2194 }

```

`glsaddstoragekey` `\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.

```
2195 \newcommand*{\glsaddstoragekey}{\@ifstar\sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```

2196 \newcommand*{\@sglsaddstoragekey}[1]{%
2197   \key@ifundefined{glossentry}{#1}%
2198   {%
2199     \expandafter\newcommand\expandafter*\expandafter
2200     {\csname gls@assign@#1@field\endcsname}[2]{%
2201       \@gls@expand@field{##1}{#1}{##2}%
2202     }%
2203   }%
2204   {}%
2205   \@glsaddstoragekey{#1}%
2206 }

```

Unstarred version doesn't override default expansion.

```

2207 \newcommand*{\@glsaddstoragekey}[3]{%

```

Check the specified key doesn't already exist.

```

2208   \key@ifundefined{glossentry}{#1}%
2209   {%

```

Set up the key.

```

2210     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2211     \appto\@gls@keymap{,{#1}{#1}}%

```

Set the default value.

```

2212     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

2213     \appto\@newglossaryentryposthook{%
2214       \letcs{\@glo@tmp}{@glo@#1}%
2215       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2216     }%

```

Define the no-link commands.

```

2217     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2218   }%
2219   {%
2220     \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2221   }%
2222 }

```

\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.

```

2223 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}

```

Starred version switches on expansion for this key.

```

2224 \newcommand*{\@sglsaddkey}[1]{%
2225   \key@ifundefined{glossentry}{#1}%

```

```

2226 {%
2227   \expandafter\newcommand\expandafter*\expandafter
2228     {\csname gls@assign@#1@field\endcsname}[2]{%
2229       \@gls@expand@field{##1}{#1}{##2}%
2230     }%
2231   }%
2232   {}%
2233   \@gls@saddkey{#1}%
2234 }

```

Unstarred version doesn't override default expansion.

```

2235 \newcommand*{\@gls@saddkey}[7]{%

```

Check the specified key doesn't already exist.

```

2236   \key@ifundefined{glossentry}{#1}%
2237   {%

```

Set up the key.

```

2238     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2239     \appto\@gls@keymap{,{#1}{#1}}%

```

Set the default value.

```

2240     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

2241     \appto\@newglossaryentryposthook{%
2242       \letcs{\@glo@tmp}{@glo@#1}%
2243       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2244     }%

```

Define the no-link commands.

```

2245     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2246     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2247     \ifcsdef{@gls@user@#1@}%
2248     {%
2249       \PackageError{glossaries}%
2250       {Can't define '\string#5' as helper command
2251       '\expandafter\string\csname @gls@user@#1\endcsname' already exists}%
2252       {}%
2253     }%
2254     {%
2255       \expandafter\newcommand\expandafter*\expandafter
2256         {\csname @gls@user@#1\endcsname}[2][{}%
2257         \new@ifnextchar[%
2258           {\csuse{@gls@user@#1@}{##1}{##2}}%
2259           {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
2260       \csdef{@gls@user@#1@}##1##2[##3]{%
2261         \@gls@field@link{##1}{##2}{#3{##2}##3}%
2262       }%

```

```

2263     \newrobustcmd*{#5}{%
2264     \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2265 }%

```

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

```

2266     \ifcsdef{@Gls@user@#1@}%
2267     {%
2268     \PackageError{glossaries}%
2269     {Can't define '\string#6' as helper command
2270     '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
2271     }%
2272 }%
2273 {%
2274     \expandafter\newcommand\expandafter*\expandafter
2275     {\csname @Gls@user@#1\endcsname}[2][ ]{%
2276     \new@ifnextchar[%
2277     {\csuse{@Gls@user@#1@}{##1}{##2}}%
2278     {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}}%
2279     \csdef{@Gls@user@#1@}##1##2[##3]{%
2280     \@gls@field@link{##1}{##2}{#4{##2}##3}%
2281     }%
2282     \newrobustcmd*{#6}{%
2283     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2284 }%

```

Finally the all caps version:

```

2285     \ifcsdef{@GLS@user@#1@}%
2286     {%
2287     \PackageError{glossaries}%
2288     {Can't define '\string#7' as helper command
2289     '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2290     }%
2291 }%
2292 {%
2293     \expandafter\newcommand\expandafter*\expandafter
2294     {\csname @GLS@user@#1\endcsname}[2][ ]{%
2295     \new@ifnextchar[%
2296     {\csuse{@GLS@user@#1@}{##1}{##2}}%
2297     {\csuse{@GLS@user@#1@}{##1}{##2}[ ]}}%
2298     \csdef{@GLS@user@#1@}##1##2[##3]{%
2299     \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2300     }%
2301     \newrobustcmd*{#7}{%
2302     \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2303 }%
2304 }%
2305 {%
2306     \PackageError{glossaries}{Key '#1' already exists}{}%

```



```

2307 }%
2308 }

```

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

```

2309 \newcommand{\glsfieldxdef}[3]{%
2310   \glsdoifexists{#1}%
2311   {%
2312     \edef\@glo@label{\glsdetoklabel{#1}}%
2313     \ifcsdef{glo@\@glo@label @#2}%
2314     {%
2315       \protected@csxdef{glo@\@glo@label @#2}{#3}%
2316     }%
2317     {%
2318       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2319     }%
2320   }%
2321 }

```

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

```

2322 \newcommand{\glsfielddedef}[3]{%
2323   \glsdoifexists{#1}%
2324   {%
2325     \edef\@glo@label{\glsdetoklabel{#1}}%
2326     \ifcsdef{glo@\@glo@label @#2}%
2327     {%
2328       \protected@csedef{glo@\@glo@label @#2}{#3}%
2329     }%
2330     {%
2331       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2332     }%
2333   }%
2334 }

```

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

```

2335 \newcommand{\glsfieldgdef}[3]{%
2336   \glsdoifexists{#1}%
2337   {%
2338     \edef\@glo@label{\glsdetoklabel{#1}}%
2339     \ifcsdef{glo@\@glo@label @#2}%
2340     {%

```

```

2341     \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2342 }%
2343 {%
2344     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2345 }%
2346 }%
2347 }

```

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

```

2348 \newcommand{\glsfielddef}[3]{%
2349   \glsdoifexists{#1}%
2350   {%
2351     \edef\@glo@label{\glsdetoklabel{#1}}%
2352     \ifcsdef{glo@\@glo@label @#2}%
2353     {%
2354       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2355     }%
2356     {%
2357       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2358     }%
2359   }%
2360 }

```

`\glsfieldfetch` `\glsfieldfetch{<label>}{<field>}{<cs>}`

Fetches the value of the given field and stores in the given control sequence.

```

2361 \newcommand{\glsfieldfetch}[3]{%
2362   \glsdoifexists{#1}%
2363   {%
2364     \edef\@glo@label{\glsdetoklabel{#1}}%
2365     \ifcsdef{glo@\@glo@label @#2}%
2366     {%
2367       \letcs#3{glo@\@glo@label @#2}%
2368     }%
2369     {%
2370       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2371     }%
2372   }%
2373 }

```

`\ifglsfieldeq` `\ifglsfieldeq{<label>}{<field>}{<string>}{<true>}{<false>}`

Tests if the value of the given field is equal to the given string.

```

2374 \newcommand{\ifglsfieldefeq}[5]{%
2375   \glsdoifexists{#1}%
2376   {%
2377     \edef\@glo@label{\glsdetoklabel{#1}}%
2378     \ifcsdef{glo@\@glo@label @#2}%
2379     {%
2380       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2381     }%
2382   }%
2383   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2384 }%
2385 }%
2386 }

```

`\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.

```

2387 \newcommand{\ifglsfieldddefeq}[5]{%
2388   \glsdoifexists{#1}%
2389   {%
2390     \edef\@glo@label{\glsdetoklabel{#1}}%
2391     \ifcsdef{glo@\@glo@label @#2}%
2392     {%
2393       \expandafter\ifdefstrequal
2394       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2395     }%
2396   }%
2397   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2398 }%
2399 }%
2400 }

```

`\ifglsfielddcseq` `\ifglsfielddcseq{<label>}{<field>}{<cs name>}{<true>}{<false>}`

As above but uses `\ifcsstrequal` instead of `\ifdefstrequal`

```

2401 \newcommand{\ifglsfielddcseq}[5]{%
2402   \glsdoifexists{#1}%
2403   {%
2404     \edef\@glo@label{\glsdetoklabel{#1}}%
2405     \ifcsdef{glo@\@glo@label @#2}%
2406     {%
2407       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2408     }%
2409   }%
2410   \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2411 }%

```

```

2412 }%
2413 }

```

gls.writedefhook

```

2414 \newcommand*{\gls.writedefhook}{}

```

gls.assign@desc

```

2415 \newcommand*{\gls.assign@desc}[1]{%
2416   \gls.assign@field{}{#1}{desc}{\@glo@desc}%
2417   \gls.assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2418 }

```

ewglossaryentry

```

2419 \newcommand{\longnewglossaryentry}[3]{%
2420   \glsdoifnoexists{#1}%
2421   {%
2422     \bgroup
2423     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2424     \long\def\@newglossaryentryprehook{%
2425       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2426       \@org@newglossaryentryprehook
2427     }%
2428     \renewcommand*{\gls.assign@desc}[1]{%
2429       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2430       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2431     }
2432     \gls@defglossaryentry{#1}{#2}%
2433   \egroup
2434 }%
2435 }

```

Only allowed in the preamble. (Otherwise a long description could cause problems when writing the entry definition to the temporary file.)

```

2436 \@onlypreamble{\longnewglossaryentry}

```

deglossaryentry As the above but only defines the entry if it doesn't already exist.

```

2437 \newcommand{\longprovideglossaryentry}[3]{%
2438   \ifglstryexists{#1}{}%
2439   {\longnewglossaryentry{#1}{#2}{#3}}%
2440 }
2441 \@onlypreamble{\longprovideglossaryentry}

```

defglossaryentry

```
\gls@defglossaryentry{<label>}{<key-val list>}
```

Defines a new entry without checking if it already exists.

```

2442 \newcommand{\gls@defglossaryentry}[2]{%

```

Prevent any further use of \GlsSetQuote:

```
2443 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2444 \edef\@glo@label{\glsdetoklabel{#1}}%
```

Provide a means for user defined keys to reference the label:

```
2445 \let\glslabel\@glo@label
```

Set up defaults. If the name or description keys are omitted, an error will be generated.

```
2446 \let\@glo@name\@gls@noname
```

```
2447 \let\@glo@desc\@gls@nodesc
```

```
2448 \let\@glo@descplural\@gls@default@value
```

```
2449 \let\@glo@type\@gls@default@value
```

```
2450 \let\@glo@symbol\@gls@default@value
```

```
2451 \let\@glo@symbolplural\@gls@default@value
```

```
2452 \let\@glo@text\@gls@default@value
```

```
2453 \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.)

```
2454 \let\@glo@first\@gls@default@value
```

```
2455 \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2456 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2457 \let\@glo@counter\@gls@default@value
```

```
2458 \def\@glo@see{}%
```

```
2459 \def\@glo@parent{}%
```

```
2460 \def\@glo@prefix{}%
```

Initialise nonnumberlist setting if we're in the document environment.

```
2461 \@gls@initnonumberlist
```

```
2462 \def\@glo@useri{}%
```

```
2463 \def\@glo@userii{}%
```

```
2464 \def\@glo@useriii{}%
```

```
2465 \def\@glo@useriv{}%
```

```
2466 \def\@glo@userv{}%
```

```
2467 \def\@glo@uservi{}%
```

```

2468 \def\@glo@short{}%
2469 \def\@glo@shortpl{}%
2470 \def\@glo@long{}%
2471 \def\@glo@longpl{}%

```

Add start hook in case another package wants to add extra keys.

```

2472 \@newglossaryentryprehook

```

Extract key-val information from third parameter:

```

2473 \setkeys{glossentry}{#2}%

```

Check there is a default glossary.

```

2474 \ifundef\glsdefaulttype
2475 {%
2476   \PackageError{glossaries}%
2477     {No default glossary type (have you used ‘nomain’ by mistake?)}%
2478     {If you use package option ‘nomain’ you must define
2479      a new glossary before you can define entries}%
2480 }%
2481 {}%

```

Assign type. This must be fully expandable

```

2482 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2483 \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.

```

2484 \ifcsundef{glolist@\@glo@type}%
2485 {%
2486   \PackageError{glossaries}%
2487     {Glossary type ‘\@glo@type’ has not been defined}%
2488     {You need to define a new glossary type, before making entries
2489      in it}%
2490 }%
2491 {}%

```

Check if it's an ignored glossary

```

2492 \ifignoredglossary\@glo@type
2493 {%

```

The description may be omitted for an entry in an ignored glossary.

```

2494   \ifx\@glo@desc\glsnodelsc
2495     \let\@glo@desc\empty
2496   \fi
2497 }%
2498 {%
2499 }%
2500 \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
2501 \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
2502   \@glolist@{\@glo@label},}%
2503 }%

```

Initialise level to 0.

```

2504 \gls@level=0\relax

```

Has this entry been assigned a parent?

```

2505 \ifx\@glo@parent\@empty

```

Doesn't have a parent. Set `\glo@<label>@parent` to empty.

```

2506 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2507 \else

```

Has a parent. Check to ensure this entry isn't its own parent.

```

2508 \ifdefequal\@glo@label\@glo@parent%
2509 {%
2510 \PackageError{glossaries}{Entry '@glo@label' can't be its own parent}{}%
2511 \def\@glo@parent{}%
2512 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2513 }%
2514 {%

```

Check the parent exists:

```

2515 \ifglentryexists{\@glo@parent}%
2516 {%

```

Parent exists. Set `\glo@<label>@parent`.

```

2517 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2518 \@glo@parent}%

```

Determine level.

```

2519 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2520 \advance\gls@level by 1\relax

```

If name hasn't been specified, use same as the parent name

```

2521 \ifx\@glo@name\@gls@name
2522 \expandafter\let\expandafter\@glo@name
2523 \csname glo@\@glo@parent @name\endcsname

```

If name and plural haven't been specified, use same as the parent

```

2524 \ifx\@glo@plural\@gls@default@value
2525 \expandafter\let\expandafter\@glo@plural
2526 \csname glo@\@glo@parent @plural\endcsname
2527 \fi
2528 \fi
2529 }%
2530 {%

```

Parent doesn't exist, so issue an error message and change this entry to have no parent

```

2531 \PackageError{glossaries}%
2532 {%
2533 Invalid parent '@glo@parent'
2534 for entry '@glo@label' - parent doesn't exist%
2535 }%
2536 {%
2537 Parent entries must be defined before their children%

```

```

2538     }%
2539     \def\@glo@parent{}%
2540     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2541     }%
2542     }%
2543 \fi

Set the level for this entry
2544 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

Define commands associated with this entry:
2545 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2546 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2547 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2548 \expandafter\gls@assign@field\expandafter
2549     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2550     {\@glo@label}{plural}{\@glo@plural}%
2551 \expandafter\gls@assign@field\expandafter
2552     {\csname glo@\@glo@label @text\endcsname}%
2553     {\@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.
2554 \ifx\@glo@first\@gls@default@value
2555     \expandafter\gls@assign@field\expandafter
2556         {\csname glo@\@glo@label @plural\endcsname}%
2557         {\@glo@label}{firstpl}{\@glo@firstplural}%
2558 \else
2559     \expandafter\gls@assign@field\expandafter
2560         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2561         {\@glo@label}{firstpl}{\@glo@firstplural}%
2562 \fi

2563 \ifcsundef{@glotype@\@glo@type @counter}%
2564 {%
2565     \def\@glo@defaultcounter{\glscounter}%
2566     }%
2567 {%
2568     \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
2569     }%
2570 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2571 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2572 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2573 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2574 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2575 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2576 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2577 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2578 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2579 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2580 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%

```



```

2581 \ifx\@glo@name\@glsnoname
2582   \@glsnoname
2583   \let\@glo@name\@gls@default@value
2584 \fi
2585 \gls@assign@field{}\@glo@label{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2586 \ifcsundef{glo@\@glo@label @numberlist}%
2587 {%
2588   \csxdef{glo@\@glo@label @numberlist}{%
2589     \noexpand\@gls@missingnumberlist{\@glo@label}}%
2590 }%
2591 {}%

```

Store nonumberlist setting if we're in the document environment.

```

2592 \@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.

```

2593 \def\@glo@@desc{\@glo@first}%
2594 \ifx\@glo@desc\@glo@@desc
2595   \let\@glo@desc\@glo@first
2596 \fi
2597 \ifx\@glo@desc\@gls@nodesc
2598   \@gls@nodesc
2599   \let\@glo@desc\@gls@default@value
2600 \fi
2601 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2602 \@gls@defsort{\@glo@type}{\@glo@label}%

2603 \def\@glo@@symbol{\@glo@text}%
2604 \ifx\@glo@symbol\@glo@@symbol
2605   \let\@glo@symbol\@glo@text
2606 \fi
2607 \gls@assign@field{relax}{\@glo@label}{symbol}{\@glo@symbol}%
2608 \expandafter
2609   \gls@assign@field\expandafter
2610   {\csname glo@\@glo@label @symbol\endcsname}
2611   {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

Define an associated boolean variable to determine whether this entry has been used yet (needs to be defined globally):

```

2612 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2613   \noexpand\global
2614     \noexpand\let\expandafter\noexpand
2615     \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2616 }%
2617 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2618   \noexpand\global

```

```

2619      \noexpand\let\expandafter\noexpand
2620      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2621  }%
2622  \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```
2623  \@glo@autosee
```

Determine and store main part of the entry's index format.

```

2624  \ifignoredglossary\@glo@type
2625  {%
2626    \csdef{glo@\@glo@label @index}{}%
2627  }
2628  {%
2629    \do@glo@storeentry{\@glo@label}%
2630  }%

```

Define entry counters if enabled:

```
2631  \@newglossaryentry@defcounters
```

Add end hook in case another package wants to add extra keys.

```

2632  \@newglossaryentryposthook
2633 }

```

\@glo@autosee Automatically implement \glssee.

```

2634 \newcommand*{\@glo@autosee}{%
2635   \ifdefvoid\@glo@see{}%
2636   {%
2637     \protected@edef\@do@glssee{%
2638       \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2639       \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
2640     \@do@glssee
2641   }%
2642   \@glo@autoseehook
2643 }%

```

glo@autoseehook

```
2644 \newcommand*{\@glo@autoseehook}{}
```

aryentryprehook Allow extra information to be added to glossary entries:

```
2645 \newcommand*{\@newglossaryentryprehook}{}
```

ryentryposthook Allow extra information to be added to glossary entries:

```
2646 \newcommand*{\@newglossaryentryposthook}{}
```

try@defcounters

```
2647 \newcommand*{\@newglossaryentry@defcounters}{}
```

`\glsmoveentry` Moves entry whose label is given by first argument to the glossary named in the second argument.

```

2648 \newcommand*{\glsmoveentry}[2]{%
2649   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2650   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2651   \def\glo@list{,%}
2652   \forglsentries[\glo@type]{\glo@label}%
2653   {%

2654     \ifdefequal\@glo@thislabel\glo@label
2655       {\eappto\glo@list{\glo@label,}}%
2656     }%
2657     \cslet\glo@list@\glo@type{\glo@list}%
2658     \csdef{glo@\@glo@thislabel @type}{#2}%
2659 }

```

`glossaryentryfield` Indicate what command should be used to display each entry in the glossary. (This enables the glossaries-accsupp package to use `\accsuppglossaryentryfield` instead.)

```

2660 \ifglxindy
2661   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2662 \else
2663   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2664 \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.)

```

2665 \ifglxindy
2666   \newcommand*{\@glossarysubentryfield}{%
2667     \string\subglossentry}
2668 \else
2669   \newcommand*{\@glossarysubentryfield}{%
2670     \string\subglossentry}
2671 \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.)

```

2672 \newcommand{\@glo@storeentry}[1]{%

```

Escape makeindex/xindy special characters in the label:

```

2673   \edef\@glo@esclabel{#1}%
2674   \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters

```

2675 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2676 \@gls@checkmkidxchars\@glo@sort
    Same again for the name string. Escape any special characters in the prefix
2677 \@gls@checkmkidxchars\@glo@prefix
    Get the parent, if one exists
2678 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
    Write the information to the glossary file.
2679 \ifglxsindy
    Store using xindy syntax.
2680 \ifx\@glo@parent\@empty
    Entry doesn't have a parent
2681 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2682 (\string"\@glo@sort\string" %
2683 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2684 }%
2685 \else
    Entry has a parent
2686 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2687 \csname glo@\@glo@parent @index\endcsname
2688 (\string"\@glo@sort\string" %
2689 \string"\@glo@prefix\@glossarysubentryfield
2690 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2691 }%
2692 \fi
2693 \else
    Store using makeindex syntax.
2694 \ifx\@glo@parent\@empty
    Sanitize \@glo@prefix
2695 \@onelevel@sanitize\@glo@prefix
    Entry doesn't have a parent
2696 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2697 \@glo@sort\@gls@actualchar\@glo@prefix
2698 \@glossaryentryfield{\@glo@esclabel}%
2699 }%
2700 \else
    Entry has a parent
2701 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2702 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2703 \@glo@sort\@gls@actualchar\@glo@prefix
2704 \@glossarysubentryfield
2705 {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2706 }%
2707 \fi
2708 \fi
2709 }

```

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`

```
2710 \AtBeginDocument{%
2711   \ifpackageloaded{amsmath}%
2712   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2713   }%
2714 }
2715 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2716   \ifmeasuring@
2717   \else
2718     #1%
2719   \fi
2720 }
2721 \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.

```
2722 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2723   \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2724 }
2725 \newcommand*\glspatchtabularx{%
2726   \ifdef\TX@trial
2727   {%
2728     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2729     \let\glspatchtabularx\relax
2730   }%
2731   {%
2732 }
```

`\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.

```
2733 \newcommand*{\glsreset}[1]{%
2734   \gls@ifnotmeasuring
2735   {%
2736     \glsdoifexists{#1}%
2737     {%
2738       \@glsreset{#1}%
2739     }%
2740   }%
2741 }
```

`\glslocalreset` As above, but with only a local effect:

```

2742 \newcommand*{\glslocalreset}[1]{%
2743   \gls@ifnotmeasuring
2744   {%
2745     \glsdoifexists{#1}%
2746     {%
2747       \@glslocalreset{#1}%
2748     }%
2749   }%
2750 }

```

`\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.

```

2751 \newcommand*{\glsunset}[1]{%
2752   \gls@ifnotmeasuring
2753   {%
2754     \glsdoifexists{#1}%
2755     {%
2756       \@glsunset{#1}%
2757     }%
2758   }%
2759 }

```

`\glslocalunset` As above, but with only a local effect:

```

2760 \newcommand*{\glslocalunset}[1]{%
2761   \gls@ifnotmeasuring
2762   {%
2763     \glsdoifexists{#1}%
2764     {%
2765       \@glslocalunset{#1}%
2766     }%
2767   }%
2768 }

```

`\@glslocalunset` Local unset. This defaults to just `\@glslocalunset` but is changed by `\glsenableentrycount`.

```

2769 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2770 \newcommand*{\@glslocalunset}[1]{%
2771   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2772 }

```

`\@glsunset` Global unset. This defaults to just `\@glsunset` but is changed by `\glsenableentrycount`.

```

2773 \newcommand*{\@glsunset}{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2774 \newcommand*{\@@glsunset}[1]{%
2775   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2776 }

```

`\@glslocalreset` Local reset. This defaults to just `\@@glslocalreset` but is changed by `\glsenableentrycount`.

```
2777 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`\@@glslocalreset` Local reset without checks.

```
2778 \newcommand*{\@@glslocalreset}[1]{%
2779   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2780 }
```

`\@glsreset` Global reset. This defaults to just `\@@glsreset` but is changed by `\glsenableentrycount`.

```
2781 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2782 \newcommand*{\@@glsreset}[1]{%
2783   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2784 }
```

Reset all entries for the named glossaries (supplied in a comma-separated list). Syntax:
`\glsresetall[⟨glossary-list⟩]`

`\glsresetall`

```
2785 \newcommand*{\glsresetall}[1][\@glo@types]{%
2786   \forallglsentries[#1]{\@glsentry}%
2787   {%
2788     \glsreset{\@glsentry}%
2789   }%
2790 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2791 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2792   \forallglsentries[#1]{\@glsentry}%
2793   {%
2794     \glslocalreset{\@glsentry}%
2795   }%
2796 }
```

Unset all entries for the named glossaries (supplied in a comma-separated list). Syntax:
`\glsunsetall[⟨glossary-list⟩]`

`\glsunsetall`

```
2797 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2798   \forallglsentries[#1]{\@glsentry}%
2799   {%
2800     \glsunset{\@glsentry}%
2801   }%
2802 }
```

As above, but with only a local effect:

lslocalunsetall

```
2803 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2804   \forallglsentries[#1]{\@glsentry}%
2805   {%
2806     \glslocalunset{\@glsentry}%
2807   }%
2808 }
```

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.

```
2809 \newcommand*{\@newglossaryentry@defcounters}{%
2810   \csdef{glo@\@glo@label @currcount}{0}%
2811   \csdef{glo@\@glo@label @prevcount}{0}%
2812 }
```

`nableentrycount` Enables tracking of how many times an entry has been marked as used.

```
2813 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2814   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2815   \renewcommand*{\gls@defdocnewglossaryentry}{%
2816     \renewcommand*{\newglossaryentry}[2]{%
2817       \PackageError{glossaries}{\string\newglossaryentry\space
2818       may only be used in the preamble when entry counting has
2819       been activated}{If you use \string\glsenableentrycount\space
2820       you must place all entry definitions in the preamble not in
2821       the document environment}%
2822     }%
2823   }%
```

Define commands `\glsentrycurrcount` and `\glsentryprevcount` to access these new fields. Default to zero if undefined.

```
2824   \newcommand*{\glsentrycurrcount}[1]{%
2825     \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2826     {0}{\@gls@entry@field{##1}{currcount}}%
2827   }%
2828   \newcommand*{\glsentryprevcount}[1]{%
```



```

2829 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2830 {0}{\@gls@entry@field{##1}{prevcount}}}%
2831 }%

```

Make the unset and reset functions also increment or reset the entry counter.

```

2832 \renewcommand*{\@glsunset}[1]{%
2833   \@glsunset{##1}%
2834   \@gls@increment@currcount{##1}%
2835 }%
2836 \renewcommand*{\@glslocalunset}[1]{%
2837   \@glslocalunset{##1}%
2838   \@gls@local@increment@currcount{##1}%
2839 }%
2840 \renewcommand*{\@glsreset}[1]{%
2841   \@glsreset{##1}%
2842   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2843 }%
2844 \renewcommand*{\@glslocalreset}[1]{%
2845   \@glslocalreset{##1}%
2846   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2847 }%

```

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.)

```

2848 \def\@cgl's@##1##2[##3]{%
2849   \ifnum\glsentryprevcount{##2}=1\relax
2850     \cgl'sformat{##2}{##3}%
2851     \glsunset{##2}%
2852   \else
2853     \@gls@{##1}{##2}[##3]%
2854   \fi
2855 }%

```

Similarly for the analogous commands. No case change plural:

```

2856 \def\@cgl'spl@##1##2[##3]{%
2857   \ifnum\glsentryprevcount{##2}=1\relax
2858     \cgl'splformat{##2}{##3}%
2859     \glsunset{##2}%
2860   \else
2861     \@cgl'spl@{##1}{##2}[##3]%
2862   \fi
2863 }%

```

First letter uppercase singular:

```

2864 \def\@cGls@##1##2[##3]{%
2865   \ifnum\glsentryprevcount{##2}=1\relax
2866     \cGlsformat{##2}{##3}%
2867     \glsunset{##2}%
2868   \else
2869     \@Gls@{##1}{##2}[##3]%
2870   \fi

```

2871 }%

First letter uppercase plural:

```
2872 \def\@cGlspl@##1##2[##3]{%
2873   \ifnum\glsentryprevcount{##2}=1\relax
2874     \cGlsplformat{##2}{##3}%
2875     \glsunset{##2}%
2876   \else
2877     \@Glspl@{##1}{##2}[##3]%
2878   \fi
2879 }%
```

Write information to aux file at the end of the document

```
2880 \AtEndDocument{\@gls@write@entrycounts}%
```

Fetch previous count information from aux file. (No check here to determine if the entry is still defined.)

```
2881 \renewcommand*{\@gls@entry@count}[2]{%
2882   \csxdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
2883 }%
```

\glsenableentrycount may only be used once and only in the preamble.

```
2884 \let\glsenableentrycount\relax
2885 }
2886 \@onlypreamble\glsenableentrycount
```

ement@currcount

```
2887 \newcommand*{\@gls@increment@currcount}[1]{%
2888   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2889     \number\numexpr\glsentrycurrcount{##1}+1}%
2890 }
```

ement@currcount

```
2891 \newcommand*{\@gls@local@increment@currcount}[1]{%
2892   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2893     \number\numexpr\glsentrycurrcount{##1}+1}%
2894 }
```

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.)

```
2895 \newcommand*{\@gls@write@entrycounts}{%
2896   \immediate\write\@auxout
2897   {\string\providecommand*\string\@gls@entry@count}[2]{}}%
2898 \forallglsentries{\@glsentry}{%
2899   \ifglsused{\@glsentry}%
2900   {\immediate\write\@auxout
2901     {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2902   }%
2903 }
```

```

2903 }%
2904 }

gls@entry@count Default behaviour is to ignore arguments. Activated by \glsenableentrycount.
2905 \newcommand*{\@gls@entry@count}[2]{%

\cgl's 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.)
2906 \newrobustcmd*{\cgl's}{\@gls@hyp@opt\@cgl's}

\@cgl's Defined the un-starred form. Need to determine if there is a final optional argument
2907 \newcommand*{\@cgl's}[2][{}]{%
2908 \new@ifnextchar[{\@cgl's@{#1}{#2}}{\@cgl's@{#1}{#2}[]}%
2909 }

\@cgl's@ Read in the final optional argument. This defaults to same behaviour as \gls but issues a
warning.
2910 \def\@cgl's@#1#2[#3]{%
2911 \GlossariesWarning{\string\cgl's\space is defaulting to
2912 \string\gls\space since you haven't enabled entry counting}%
2913 \@gls@{#1}{#2}[#3]%
2914 }

\cgl'sformat Format used by \cgl's if entry only used once on previous run. The first argument is the label,
the second argument is the insert text.
2915 \newcommand*{\cgl'sformat}[2]{%
2916 \ifgl'shaslong{#1}{\gl'sentrylong{#1}}{\gl'sentryfirst{#1}}#2%
2917 }

\cGls 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.)
2918 \newrobustcmd*{\cGls}{\@gls@hyp@opt\@cGls}

\@cGls Defined the un-starred form. Need to determine if there is a final optional argument
2919 \newcommand*{\@cGls}[2][{}]{%
2920 \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2}[]}%
2921 }

\@cGls@ Read in the final optional argument. This defaults to same behaviour as \Gls but issues a
warning.
2922 \def\@cGls@#1#2[#3]{%
2923 \GlossariesWarning{\string\cGls\space is defaulting to
2924 \string\Gls\space since you haven't enabled entry counting}%
2925 \@Gls@{#1}{#2}[#3]%
2926 }

```

`\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.

```
2927 \newcommand*{\cGlsformat}[2]{%
2928   \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2929 }
```

`\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.)

```
2930 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}
```

`\@cglsp1` Defined the un-starred form. Need to determine if there is a final optional argument

```
2931 \newcommand*{\@cglsp1}[2][{}]{%
2932   \new@ifnextchar[\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2}[{}]}%
2933 }
```

`\@cglsp1@` Read in the final optional argument. This defaults to same behaviour as `\glsp1` but issues a warning.

```
2934 \def\@cglsp1@#1#2[#3]{%
2935   \GlossariesWarning{\string\cglsp1\space is defaulting to
2936     \string\glsp1\space since you haven't enabled entry counting}%
2937   \@glsp1@{#1}{#2}[#3]%
2938 }
```

`\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.

```
2939 \newcommand*{\cglsp1format}[2]{%
2940   \ifglshaslong{#1}{\glsp1long{#1}}{\glsp1firstplural{#1}}#2%
2941 }
```

`\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.)

```
2942 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}
```

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

```
2943 \newcommand*{\@cGlspl}[2][{}]{%
2944   \new@ifnextchar[\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[{}]}%
2945 }
```

`\@cGlspl@` Read in the final optional argument. This defaults to same behaviour as `\Glspl` but issues a warning.

```
2946 \def\@cGlspl@#1#2[#3]{%
2947   \GlossariesWarning{\string\cGlspl\space is defaulting to
2948     \string\Glspl\space since you haven't enabled entry counting}%
2949   \@Glspl@{#1}{#2}[#3]%
2950 }
```

`\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.

```

2951 \newcommand*{\cGlsplformat}[2]{%
2952   \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2953 }

```

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`

```

2954 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2955   \let\@gls@default\glsdefaulttype
2956   \def\glsdefaulttype{#1}\input{#2}%
2957   \let\glsdefaulttype\@gls@default
2958 }

```

`\loadglsentries` can only be used in the preamble:

```

2959 \@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`

```

2960 \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 \TeX code that can be used in the preamble.

ensure backward compatibility, the default use the old `\glsdisplay` and `\glsdisplayfirst` style of commands

```
2961 \newcommand*{\glsentryfmt}{%
2962   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2963 }
```

Format that provides backwards compatibility:

```
2964 \newcommand*{\@gls@default@entryfmt}[2]{%
2965   \ifdefempty\glscustomtext
2966   {%
2967     \glsifplural
2968     {%
```

Plural form

```
2969     \glscapscase
2970     {%
```

Don't adjust case

```
2971     \ifglsused\glslabel
2972     {%
```

Subsequent use

```
2973         #2{\glsentryplural{\glslabel}}%
2974         {\glsentrydescplural{\glslabel}}%
2975         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2976     }%
2977     {%
```

First use

```
2978         #1{\glsentryfirstplural{\glslabel}}%
2979         {\glsentrydescplural{\glslabel}}%
2980         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2981     }%
2982     }%
2983     {%
```

Make first letter upper case

```
2984     \ifglsused\glslabel
2985     {%
```

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.)

```
2986     \ifbool{glscompatible-3.07}%
2987     {%
2988       \protected@edef\@glo@etext{%
2989         #2{\glsentryplural{\glslabel}}%
2990         {\glsentrydescplural{\glslabel}}%
2991         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2992       \xmakefirstuc\@glo@etext
2993     }%
```

```

2994      {%
2995      #2{\Glsentryplural{\glslabel}}%
2996      {\glsentrydescplural{\glslabel}}%
2997      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2998      }%
2999      }%
3000      {%

First use
3001      \ifbool{glscompatible-3.07}%
3002      {%
3003      \protected@edef\@glo@etext{%
3004      #1{\Glsentryfirstplural{\glslabel}}%
3005      {\glsentrydescplural{\glslabel}}%
3006      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3007      \xmakefirstuc\@glo@etext
3008      }%
3009      {%
3010      #1{\Glsentryfirstplural{\glslabel}}%
3011      {\glsentrydescplural{\glslabel}}%
3012      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
3013      }%
3014      }%
3015      }%
3016      {%

Make all upper case
3017      \ifglsused\glslabel
3018      {%

Subsequent use
3019      \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
3020      {\glsentrydescplural{\glslabel}}%
3021      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3022      }%
3023      {%

First use
3024      \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
3025      {\glsentrydescplural{\glslabel}}%
3026      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3027      }%
3028      }%
3029      }%
3030      {%

Singular form
3031      \glscapscase
3032      {%

Don't adjust case
3033      \ifglsused\glslabel

```

```

3034      {%
      Subsequent use
3035          #2{\glsentrytext{\glslabel}}%
3036          {\glsentrydesc{\glslabel}}%
3037          {\glsentrysymbol{\glslabel}}{\glsinsert}%
3038      }%
3039      {%

      First use
3040          #1{\glsentryfirst{\glslabel}}%
3041          {\glsentrydesc{\glslabel}}%
3042          {\glsentrysymbol{\glslabel}}{\glsinsert}%
3043      }%
3044  }%
3045  {%

      Make first letter upper case
3046      \ifglused\glslabel
3047      {%

      Subsequent use
3048          \ifbool{glscompatible-3.07}%
3049          {%
3050              \protected@edef\@glo@etext{%
3051                  #2{\glsentrytext{\glslabel}}%
3052                  {\glsentrydesc{\glslabel}}%
3053                  {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3054              \xmakefirstuc\@glo@etext
3055          }%
3056          {%
3057              #2{\Glsentrytext{\glslabel}}%
3058              {\glsentrydesc{\glslabel}}%
3059              {\glsentrysymbol{\glslabel}}{\glsinsert}%
3060          }%
3061      }%
3062      {%

      First use
3063          \ifbool{glscompatible-3.07}%
3064          {%
3065              \protected@edef\@glo@etext{%
3066                  #1{\glsentryfirst{\glslabel}}%
3067                  {\glsentrydesc{\glslabel}}%
3068                  {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3069              \xmakefirstuc\@glo@etext
3070          }%
3071          {%
3072              #1{\Glsentryfirst{\glslabel}}%
3073              {\glsentrydesc{\glslabel}}%
3074              {\glsentrysymbol{\glslabel}}{\glsinsert}%

```



```

3075         }%
3076     }%
3077 }%
3078 {%

    Make all upper case
3079     \ifglused\glslabel
3080     {%

        Subsequent use
3081         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
3082         {\glsentrydesc{\glslabel}}}%
3083         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
3084     }%
3085     {%

        First use
3086         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
3087         {\glsentrydesc{\glslabel}}}%
3088         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
3089     }%
3090 }%
3091 }%
3092 }%
3093 {%

    Custom text provided in \glsdisp
3094     \ifglused{\glslabel}%
3095     {%

        Subsequent use
3096         #2{\glscustomtext}%
3097         {\glsentrydesc{\glslabel}}}%
3098         {\glsentrysymbol{\glslabel}}{}%
3099     }%
3100     {%

        First use
3101         #1{\glscustomtext}%
3102         {\glsentrydesc{\glslabel}}}%
3103         {\glsentrysymbol{\glslabel}}{}%
3104     }%
3105 }%
3106 }

```

`\glsgenentryfmt` Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

```

3107 \newcommand*{\glsgenentryfmt}{%
3108     \ifdefempty\glscustomtext
3109     {%
3110         \glsifplural
3111         {%

```

Plural form

3112 \glscapscase
3113 {%

Don't adjust case

3114 \ifglused\glslabel
3115 {%

Subsequent use

3116 \glsentryplural{\glslabel}\glsinsert
3117 }%
3118 {%

First use

3119 \glsentryfirstplural{\glslabel}\glsinsert
3120 }%
3121 }%
3122 {%

Make first letter upper case

3123 \ifglused\glslabel
3124 {%

Subsequent use.

3125 \Glsentryplural{\glslabel}\glsinsert
3126 }%
3127 {%

First use

3128 \Glsentryfirstplural{\glslabel}\glsinsert
3129 }%
3130 }%
3131 {%

Make all upper case

3132 \ifglused\glslabel
3133 {%

Subsequent use

3134 \mfirstucMakeUppercase
3135 {\glsentryplural{\glslabel}\glsinsert}%
3136 }%
3137 {%

First use

3138 \mfirstucMakeUppercase
3139 {\glsentryfirstplural{\glslabel}\glsinsert}%
3140 }%
3141 }%
3142 }%
3143 {%

Singular form

```
3144 \glscapscase
3145 {%
```

Don't adjust case

```
3146 \ifglused\glslabel
3147 {%
```

Subsequent use

```
3148 \glentrytext{\glslabel}\glsinsert
3149 }%
3150 {%
```

First use

```
3151 \glentryfirst{\glslabel}\glsinsert
3152 }%
3153 }%
3154 {%
```

Make first letter upper case

```
3155 \ifglused\glslabel
3156 {%
```

Subsequent use

```
3157 \Glentrytext{\glslabel}\glsinsert
3158 }%
3159 {%
```

First use

```
3160 \Glentryfirst{\glslabel}\glsinsert
3161 }%
3162 }%
3163 {%
```

Make all upper case

```
3164 \ifglused\glslabel
3165 {%
```

Subsequent use

```
3166 \mfirstucMakeUppercase{\glentrytext{\glslabel}\glsinsert}%
3167 }%
3168 {%
```

First use

```
3169 \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glsinsert}%
3170 }%
3171 }%
3172 }%
3173 }%
3174 {%
```

Custom text provided in \glldisp. (The insert is most likely to be empty at this point.)

```
3175 \glscustomtext\glsinsert
```

```

3176 }%
3177 }

```

`\glsgenacfmt` Define a generic acronym format that uses the long and short keys (or their plurals) and `\acrfullformat`, `\firstacronymfont` and `\acronymfont`.

```

3178 \newcommand*{\glsgenacfmt}{%
3179   \ifdefempty\glscustomtext
3180     {%
3181       \ifglused\glslabel
3182         {%

```

Subsequent use:

```

3183     \glsifplural
3184     {%

```

Subsequent plural form:

```

3185     \glscapscase
3186     {%

```

Subsequent plural form, don't adjust case:

```

3187     \acronymfont{\glstryshortpl{\glslabel}}\glsinsert
3188     }%
3189     {%

```

Subsequent plural form, make first letter upper case:

```

3190     \acronymfont{\Glstryshortpl{\glslabel}}\glsinsert
3191     }%
3192     {%

```

Subsequent plural form, all caps:

```

3193     \mfirstucMakeUppercase
3194     {\acronymfont{\glstryshortpl{\glslabel}}\glsinsert}%
3195     }%
3196     }%
3197     {%

```

Subsequent singular form

```

3198     \glscapscase
3199     {%

```

Subsequent singular form, don't adjust case:

```

3200     \acronymfont{\glstryshort{\glslabel}}\glsinsert
3201     }%
3202     {%

```

Subsequent singular form, make first letter upper case:

```

3203     \acronymfont{\Glstryshort{\glslabel}}\glsinsert
3204     }%
3205     {%

```

Subsequent singular form, all caps:

```

3206     \mfirstucMakeUppercase
3207     {\acronymfont{\glstryshort{\glslabel}}\glsinsert}%

```

```

3208      }%
3209      }%
3210      }%
3211      {%

```

First use:

```

3212      \glsifplural
3213      {%

```

First use plural form:

```

3214      \glscapscase
3215      {%

```

First use plural form, don't adjust case:

```

3216      \genplacrformat{\glslabel}{\glsinsert}%
3217      }%
3218      {%

```

First use plural form, make first letter upper case:

```

3219      \Genplacrformat{\glslabel}{\glsinsert}%
3220      }%
3221      {%

```

First use plural form, all caps:

```

3222      \mfirstucMakeUppercase
3223      {\genplacrformat{\glslabel}{\glsinsert}}%
3224      }%
3225      }%
3226      {%

```

First use singular form

```

3227      \glscapscase
3228      {%

```

First use singular form, don't adjust case:

```

3229      \genacrformat{\glslabel}{\glsinsert}%
3230      }%
3231      {%

```

First use singular form, make first letter upper case:

```

3232      \Genacrformat{\glslabel}{\glsinsert}%
3233      }%
3234      {%

```

First use singular form, all caps:

```

3235      \mfirstucMakeUppercase
3236      {\genacrformat{\glslabel}{\glsinsert}}%
3237      }%
3238      }%
3239      }%
3240      }%
3241      {%

```

User supplied text.

```
3242 \glscustomtext
3243 }%
3244 }
```

genacrfullformat `\genacrfullformat{<label>}{<insert>}`

The full format used by `\glsgenacfmt` (singular).

```
3245 \newcommand*{\genacrfullformat}[2]{%
3246   \glentrylong{#1}#2\space
3247   (\protect\firstacronymfont{\glentryshort{#1}})%
3248 }
```

Genacrfullformat `\Genacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```
3249 \newcommand*{\Genacrfullformat}[2]{%
3250   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3251   \xmakefirstuc\gls@text
3252 }
```

nplacrfullformat `\genplacrfullformat{<label>}{<insert>}`

The full format used by `\glsgenacfmt` (plural).

```
3253 \newcommand*{\genplacrfullformat}[2]{%
3254   \glentrylongpl{#1}#2\space
3255   (\protect\firstacronymfont{\glentryshortpl{#1}})%
3256 }
```

nplacrfullformat `\Genplacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```
3257 \newcommand*{\Genplacrfullformat}[2]{%
3258   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3259   \xmakefirstuc\gls@text
3260 }
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
3261 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3262 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```

3263 \newcommand*{\defglsdisplay}[2][\glsdefaultttype]{%
3264   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3265   Use \string\defglsentryfmt\space instead}%
3266   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3267   \edef\@gls@doentrydef{%
3268     \noexpand\defglsentryfmt[#1]{%
3269       \noexpand\ifcsdef{gls@#1@displayfirst}%
3270       {%
3271         \noexpand\@@gls@default@entryfmt
3272         {\noexpand\csuse{gls@#1@displayfirst}}}%
3273         {\noexpand\csuse{gls@#1@display}}}%
3274       }%
3275       {%
3276         \noexpand\@@gls@default@entryfmt
3277         {\noexpand\glsdisplayfirst}%
3278         {\noexpand\csuse{gls@#1@display}}}%
3279       }%
3280     }%
3281   }%
3282   \@gls@doentrydef
3283 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```

3284 \newcommand*{\defglsdisplayfirst}[2][\glsdefaultttype]{%
3285   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3286   Use \string\defglsentryfmt\space instead}%
3287   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3288   \edef\@gls@doentrydef{%
3289     \noexpand\defglsentryfmt[#1]{%
3290       \noexpand\ifcsdef{gls@#1@display}%
3291       {%
3292         \noexpand\@@gls@default@entryfmt
3293         {\noexpand\csuse{gls@#1@displayfirst}}}%
3294         {\noexpand\csuse{gls@#1@display}}}%
3295       }%
3296       {%
3297         \noexpand\@@gls@default@entryfmt
3298         {\noexpand\csuse{gls@#1@displayfirst}}}%
3299         {\noexpand\glsdisplay}%
3300       }%
3301     }%
3302   }%
3303   \@gls@doentrydef
3304 }
```

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.

```
3305 \define@key{glslink}{counter}{%
3306   \ifcsundef{c@#1}%
3307   {%
3308     \PackageError{glossaries}%
3309     {There is no counter called '#1'}%
3310     {%
3311       The counter key should have the name of a valid counter
3312       as its value%
3313     }%
3314   }%
3315   {%
3316     \def\@gls@counter{#1}%
3317   }%
3318 }
```

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.

```
3319 \define@key{glslink}{format}{%
3320   \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.

```
3321 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3322 \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.

```
3323 \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.

```
3324 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3325 \newcommand*{\glsifhyper}[2]{%
3326 \glslinkvar{#1}{#2}{#1}%
3327 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3328 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3329 }
```

`\@gls@hyp@opt` Used by the commands such as `\glslink` to determine whether to modify the hyper option.

```
3330 \newcommand*{\@gls@hyp@opt}[1]{%
3331 \let\glslinkvar\@firstofthree
3332 \let\@gls@hyp@opt@cs#1\relax
3333 \@ifstar{\s@gls@hyp@opt}%
3334 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3335 }
```

`\s@gls@hyp@opt` Starred version

```
3336 \newcommand*{\s@gls@hyp@opt}[1] []{%
3337 \let\glslinkvar\@secondofthree
3338 \@gls@hyp@opt@cs[hyper=false,#1]}
```

`\p@gls@hyp@opt` Plus version

```
3339 \newcommand*{\p@gls@hyp@opt}[1] []{%
3340 \let\glslinkvar\@thirdofthree
3341 \@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`

```
3342 \newrobustcmd*{\glslink}{%  
3343 \@gls@hyp@opt\@gls@@link  
3344 }
```

`\@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.

```
3345 \newcommand*{\@gls@@link}[3][\@gls@link]{%  
3346 \glsdoifexistsordo{#2}%  
3347 {%  
3348 \let\do@gls@link@checkfirsthyper\relax  
3349 \@gls@link[#1]{#2}{#3}%  
3350 }%
```

Display the specified text. (The entry doesn't exist so there's nothing to link it to.)

```
3351 \glstextformat{#3}%  
3352 }%  
  
3353 \glspostlinkhook  
3354 }
```

`glspostlinkhook`

```
3355 \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`.

```
3356 \newcommand*{\@gls@link@checkfirsthyper}{%  
3357 \ifglsused{\glslabel}%  
3358 {%  
3359 }%  
3360 {%  
3361 \gls@checkisacronymlist\glstype  
3362 \ifglshyperfirst  
3363 \if@glsisacronymlist  
3364 \ifglsacrfootnote  
3365 \KV@glslink@hyperfalse  
3366 \fi  
3367 \fi  
3368 \else  
3369 \KV@glslink@hyperfalse  
3370 \fi  
3371 }%
```

Allow user to hook into this

```
3372 \glslinkcheckfirsthyperhook  
3373 }
```

`checkfirsthyperhook` Allow used to hook into the `\@gls@link@checkfirsthyper` macro
3374 `\newcommand*{\glslinkcheckfirsthyperhook}{}`

`linkpostsetkeys`
3375 `\newcommand*{\glslinkpostsetkeys}{}`

`\glsifhyperon` Check the value of the hyper key:
3376 `\newcommand{\glsifhyperon}[2]{\ifKV@gls@link@hyper#1\else#2\fi}`

`disablehyperinlist` Disable hyperlink if in the “nohyper” list.
3377 `\newcommand*{\do@glsdisablehyperinlist}{%`
3378 `\expandafter\DTLifinlist\expandafter{\gls@type}{\@gls@nohyperlist}%`
3379 `{\KV@gls@link@hyperfalse}{}}%`
3380 `}`

`\let@gls@link@opts` Hook to set default options for `\@gls@link`.
3381 `\newcommand*{\@gls@setdefault@gls@link@opts}{}`

`\@gls@link`
3382 `\def\@gls@link[#1]#2#3{%`
Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with tabularx).
3383 `\leavevmode`
3384 `\edef\glslabel{\glsdetoklabel{#2}}%`
Save options in `\@gls@link@opts` and label in `\@gls@link@label`
3385 `\def\@gls@link@opts{#1}%`
3386 `\let\@gls@link@label\glslabel`
3387 `\def\@glsnumberformat{glsnumberformat}%`
3388 `\edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%`
If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default
3389 `\edef\gls@type{\csname glo@\glslabel @type\endcsname}%`
Save original setting
3390 `\let\org@ifKV@gls@link@hyper\ifKV@gls@link@hyper`
Set defaults:
3391 `\@gls@setdefault@gls@link@opts`
Switch off hyper setting if the glossary type has been identified in nohyperlist.
3392 `\do@glsdisablehyperinlist`
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`.
3393 `\do@gls@link@checkfirsthyper`
3394 `\setkeys{gls@link}{#1}%`
Add a hook for the user to customise things after the keys have been set.
3395 `\glslinkpostsetkeys`

```

Store the entry's counter in \theglsentrycounter
3396 \gls@saveentrycounter

Define sort key if necessary:
3397 \gls@setsort{\glslabel}%

(De-tok'ing done by \@do@wrglossary)
3398 \@do@wrglossary{#2}%
3399 \ifKV@glslink@hyper
3400 \glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3401 \else

3402 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3403 \fi

Restore original setting
3404 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3405 }

\glolinkprefix
3406 \newcommand*{\glolinkprefix}{glo:}

glsentrycounter Set default value of entry counter
3407 \def\glsentrycounter{\glscounter}%

saveentrycounter Need to check if using equation counter in align environment:
3408 \newcommand*{\gls@saveentrycounter}{%
3409 \def\gls@Hcounter{}}%

Are we using equation counter?
3410 \ifthenelse{\equal{\gls@counter}{equation}}{%
3411 {

If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as
may be inside an inner environment.)
3412 \ifcsundef{xatlevel@}%
3413 {%
3414 \edef\theglsentrycounter{\expandafter\noexpand
3415 \csname the\gls@counter\endcsname}%
3416 }%
3417 {%
3418 \ifx\xatlevel@\@empty
3419 \edef\theglsentrycounter{\expandafter\noexpand
3420 \csname the\gls@counter\endcsname}%
3421 \else
3422 \savecounters@
3423 \advance\c@equation by 1\relax
3424 \edef\theglsentrycounter{\csname the\gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3425 \ifcsundef{theH\@gls@counter}%
3426 {%
3427 \def\@gls@Hcounter{\theglsentrycounter}%
3428 }%
3429 {%
3430 \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3431 }%
3432 \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3433 \restorecounters@
3434 \fi
3435 }%
3436 }%
3437 {%

```

Not using equation counter so no special measures:

```

3438 \edef\theglsentrycounter{\expandafter\noexpand
3439 \csname the\@gls@counter\endcsname}%
3440 }%

```

Check if hyperref version of this counter

```

3441 \ifx\@gls@Hcounter\@empty
3442 \ifcsundef{theH\@gls@counter}%
3443 {%
3444 \def\theHglentrycounter{\theglsentrycounter}%
3445 }%
3446 {%
3447 \protected@edef\theHglentrycounter{\expandafter\noexpand
3448 \csname theH\@gls@counter\endcsname}%
3449 }%
3450 \fi
3451 }

```

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.

```

3452 \def\@set@glo@numformat#1#2#3#4{%
3453 \expandafter\@glo@check@mkidxrangechar#3\@nil
3454 \protected@edef#1{%
3455 \@glo@prefix setentrycounter[#4]{#2}%
3456 \expandafter\string\csname\@glo@suffix\endcsname
3457 }%
3458 \@gls@checkmkidxchars#1%
3459 }

```

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.

```

3460 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3461 \if#1(\relax
3462   \def\@glo@prefix{()%
3463   \if\relax#2\relax
3464     \def\@glo@suffix{glsnumberformat}%
3465   \else
3466     \def\@glo@suffix{#2}%
3467   \fi
3468 \else
3469   \if#1)\relax
3470     \def\@glo@prefix{}}}%
3471   \if\relax#2\relax
3472     \def\@glo@suffix{glsnumberformat}%
3473   \else
3474     \def\@glo@suffix{#2}%
3475   \fi
3476 \else
3477   \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3478 \fi
3479 \fi}

```

`\@gls@escbsdq` Escape backslashes and double quote marks. The argument must be a control sequence.

```

3480 \newcommand*{\@gls@escbsdq}[1]{%
3481   \def\@gls@checkedmkidx{%
3482     \let\gls@xdystring=#1\relax
3483     \@onelevel@sanitize\gls@xdystring
3484     \edef\do@gls@xdycheckbackslash{%
3485       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3486       \@backslashchar\@backslashchar\noexpand\null}%
3487     \do@gls@xdycheckbackslash
3488     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3489     \def\@gls@checkedmkidx{%
3490       \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3491       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

Unsanitize\gls@numberpage,\gls@alphpage,\gls@Alphpage and\gls@romanpage (thanks to David Carlisle for the suggestion.)

```

3492   \@for\@gls@tmp:=\gls@protected@pagefmts\do
3493   {%
3494     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3495     \@onelevel@sanitize\@gls@sanitized@tmp
3496     \edef\gls@dosubst{%
3497       \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3498       {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3499     }%
3500     \gls@dosubst
3501   }%

```

Assign to required control sequence

```

3502   \let#1=\gls@xdystring

```

3503 }

Catch special characters (argument must be a control sequence):

checkmkidxchars

```
3504 \newcommand{\@gls@checkmkidxchars}[1]{%
3505   \ifglxsindy
3506     \@gls@escbsdq{#1}%
3507   \else
3508     \def\@gls@checkedmkidx{%
3509       \expandafter\@gls@checkquote#1\@nil""\null
3510       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3511       \def\@gls@checkedmkidx{%
3512         \expandafter\@gls@checkescquote#1\@nil\""\null
3513         \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3514         \def\@gls@checkedmkidx{%
3515           \expandafter\@gls@checkescactual#1\@nil"??\null
3516           \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3517           \def\@gls@checkedmkidx{%
3518             \expandafter\@gls@checkactual#1\@nil??\null
3519             \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3520             \def\@gls@checkedmkidx{%
3521               \expandafter\@gls@checkbar#1\@nil||\null
3522               \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3523               \def\@gls@checkedmkidx{%
3524                 \expandafter\@gls@checkescbar#1\@nil\\|\null
3525                 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3526                 \def\@gls@checkedmkidx{%
3527                   \expandafter\@gls@checklevel#1\@nil!!\null
3528                   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3529                 \fi
3530 }
```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```
3531 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
3532 \newtoks\@gls@tmpb
```

@gls@checkquote Replace " with "" since " is a makeindex special character.

```
3533 \def\@gls@checkquote#1"#2"#3\null{%
3534   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3535   \toks@={#1}%
3536   \ifx\@nil#2\null
3537   \ifx\@nil#3\null
3538     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3539   \def\@gls@checkquote{\relax}%
3540   \else
```

```

3541 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3542 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3543 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
3544 \fi
3545 \else
3546 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3547 \@gls@quotechar\@gls@quotechar}%
3548 \ifx\null#3\null
3549 \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
3550 \else
3551 \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
3552 \fi
3553 \fi
3554 \@@gls@checkquote
3555 }

```

s@checkescquote Do the same for \":

```

3556 \def\@gls@checkescquote#1\"#2\"#3\null{%
3557 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3558 \toks@={#1}%
3559 \ifx\null#2\null
3560 \ifx\null#3\null
3561 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3562 \def\@@gls@checkescquote{\relax}%
3563 \else
3564 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3565 \@gls@quotechar\string\"@gls@quotechar
3566 \@gls@quotechar\string\"@gls@quotechar}%
3567 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3568 \fi
3569 \else
3570 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3571 \@gls@quotechar\string\"@gls@quotechar}%
3572 \ifx\null#3\null
3573 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3574 \else
3575 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3576 \fi
3577 \fi
3578 \@@gls@checkescquote
3579 }

```

@checkescactual Similarly for \? (which is replaces @ as makeindex's special character):

```

3580 \def\@gls@checkescactual#1\?#2\?#3\null{%
3581 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3582 \toks@={#1}%
3583 \ifx\null#2\null
3584 \ifx\null#3\null
3585 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```



```

3586 \def\@gls@checkescactual{\relax}%
3587 \else
3588 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3589 \@gls@quotechar\string"\@gls@actualchar
3590 \@gls@quotechar\string"\@gls@actualchar}%
3591 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3592 \fi
3593 \else
3594 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3595 \@gls@quotechar\string"\@gls@actualchar}%
3596 \ifx\null#3\null
3597 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3598 \else
3599 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3600 \fi
3601 \fi
3602 \@gls@checkescactual
3603 }

```

`gls@checkescbar` Similarly for `\|`:

```

3604 \def\@gls@checkescbar#1\|#2\|#3\null{%
3605 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3606 \toks@={#1}%
3607 \ifx\null#2\null
3608 \ifx\null#3\null
3609 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3610 \def\@gls@checkescbar{\relax}%
3611 \else
3612 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3613 \@gls@quotechar\string"\@gls@encapchar
3614 \@gls@quotechar\string"\@gls@encapchar}%
3615 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3616 \fi
3617 \else
3618 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3619 \@gls@quotechar\string"\@gls@encapchar}%
3620 \ifx\null#3\null
3621 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3622 \else
3623 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3624 \fi
3625 \fi
3626 \@gls@checkescbar
3627 }

```

`s@checkesclevel` Similarly for `\!`:

```

3628 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3629 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3630 \toks@={#1}%

```

```

3631 \ifx\null#2\null
3632 \ifx\null#3\null
3633 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3634 \def\@gls@checkesclevel{\relax}%
3635 \else
3636 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3637 \@gls@quotechar\string"\@gls@levelchar
3638 \@gls@quotechar\string"\@gls@levelchar}%
3639 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3640 \fi
3641 \else
3642 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3643 \@gls@quotechar\string"\@gls@levelchar}%
3644 \ifx\null#3\null
3645 \def\@gls@checkesclevel{\@gls@checkesclevel#2!!\null}%
3646 \else
3647 \def\@gls@checkesclevel{\@gls@checkesclevel#2!#3\null}%
3648 \fi
3649 \fi
3650 \@gls@checkesclevel
3651 }

```

\@gls@checkbar and for |:

```

3652 \def\@gls@checkbar#1|#2|#3\null{%
3653 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3654 \toks@={#1}%
3655 \ifx\null#2\null
3656 \ifx\null#3\null
3657 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3658 \def\@gls@checkbar{\relax}%
3659 \else
3660 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3661 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3662 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3663 \fi
3664 \else
3665 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3666 \@gls@quotechar\@gls@encapchar}%
3667 \ifx\null#3\null
3668 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3669 \else
3670 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3671 \fi
3672 \fi
3673 \@gls@checkbar
3674 }

```

@gls@checklevel and for !:

```

3675 \def\@gls@checklevel#1!#2!#3\null{%

```

```

3676 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3677 \toks@={#1}%
3678 \ifx\null#2\null
3679   \ifx\null#3\null
3680     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3681     \def\@gls@checklevel{\relax}%
3682   \else
3683     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3684       \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3685     \def\@gls@checklevel{\@gls@checklevel#3\null}%
3686   \fi
3687 \else
3688   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3689     \@gls@quotechar\@gls@levelchar}%
3690   \ifx\null#3\null
3691     \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3692   \else
3693     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3694   \fi
3695 \fi
3696 \@gls@checklevel
3697 }

```

gls@checkactual and for ?:

```

3698 \def\@gls@checkactual#1?#2?#3\null{%
3699   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3700   \toks@={#1}%
3701   \ifx\null#2\null
3702     \ifx\null#3\null
3703       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3704       \def\@gls@checkactual{\relax}%
3705     \else
3706       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3707         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3708       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3709     \fi
3710   \else
3711     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3712       \@gls@quotechar\@gls@actualchar}%
3713     \ifx\null#3\null
3714       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3715     \else
3716       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3717     \fi
3718   \fi
3719   \@gls@checkactual
3720 }

```

s@xdycheckquote As before but for use with xindy

```

3721 \def\@gls@xdycheckquote#1"#2"#3\null{%
3722   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3723   \toks@={#1}%
3724   \ifx\null#2\null
3725     \ifx\null#3\null
3726       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3727       \def\@gls@xdycheckquote{\relax}%
3728     \else
3729       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3730         \string"\string"}%
3731       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3732     \fi
3733   \else
3734     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3735       \string"}%
3736     \ifx\null#3\null
3737       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3738     \else
3739       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3740     \fi
3741   \fi
3742   \@gls@xdycheckquote
3743 }

```

ycheckbackslash Need to escape all backslashes for xindy. Define command that will define \@gls@xdycheckbackslash

```

3744 \edef\def\@gls@xdycheckbackslash{%
3745   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3746     ##2\@backslashchar##3\noexpand\null{%
3747     \noexpand\@gls@tmpb=\noexpand\expandafter
3748       {\noexpand\@gls@checkedmkidx}%
3749     \noexpand\toks@={##1}%
3750     \noexpand\ifx\noexpand\null##2\noexpand\null
3751       \noexpand\ifx\noexpand\null##3\noexpand\null
3752         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3753           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3754         \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3755       \noexpand\else
3756         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3757           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3758           \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3759       \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3760         \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3761       \noexpand\fi
3762     \noexpand\else
3763       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3764         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3765         \@backslashchar\@backslashchar}%
3766     \noexpand\ifx\noexpand\null##3\noexpand\null
3767       \noexpand\def\noexpand\@gls@xdycheckbackslash{%

```

```

3768      \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3769      \@backslashchar\noexpand\null}%
3770 \noexpand\else
3771      \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3772      \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3773      ##3\noexpand\null}%
3774 \noexpand\fi
3775 \noexpand\fi
3776 \noexpand\@gls@xdycheckbackslash
3777 }%
3778 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3779 \def@gls@xdycheckbackslash

```

\glsdohypertarget

```

3780 \newlength\gls@tmplen
3781 \newcommand*\glsdohypertarget}[2]{%
3782   \@glsshowtarget{#1}%
3783   \settoheight{\gls@tmplen}{#2}%
3784   \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3785 }

```

\glsdohyperlink

```

3786 \newcommand*\glsdohyperlink}[2]{%
3787   \@glsshowtarget{#1}%
3788   \hyperlink{#1}{#2}%
3789 }

```

\glsdonohyperlink

```

3790 \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.

```

3791 \ifcsundef{hyperlink}%
3792 {%
3793   \let\@glslink\glsdonohyperlink
3794 }%
3795 {%
3796   \let\@glslink\glsdohyperlink
3797 }

```

\@glstarget If \hypertarget is not defined, \@glstarget ignores its first argument and just does the second argument, otherwise it is equivalent to \hypertarget.

```

3798 \ifcsundef{hypertarget}%
3799 {%
3800   \let\@glstarget\@secondoftwo
3801 }%

```

```

3802 {%
3803   \let\@glstarget\glsdohypertarget
3804 }

```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```

3805 \newcommand{\glsdisablehyper}{%
3806   \KV@glslink@hyperfalse
3807   \let\@glslink\glsdonohyperlink
3808   \let\@glstarget\@secondoftwo
3809 }

```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```

3810 \newcommand{\glsenablehyper}{%
3811   \KV@glslink@hypertrue
3812   \let\@glslink\glsdohyperlink
3813   \let\@glstarget\glsdohypertarget
3814 }

```

Provide some convenience commands if not already defined:

```

3815 \providecommand{\@firstofthree}[3]{#1}
3816 \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`

```
3817 \newrobustcmd*\gls{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```

3818 \newcommand*\@gls[2][ ]{%
3819   \new@ifnextchar[\@gls@{#1}{#2}]{\@gls@{#1}{#2}[ ]}%
3820 }

```

`\@gls@` Read in the final optional argument:

```
3821 \def\@gls@#1#2[#3]{%
3822   \glsdoifexists{#2}%
3823   {%
3824     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3825     \let\glsifplural\@secondoftwo
3826     \let\glsupcase\@firstofthree
3827     \let\glscustomtext\@empty
3828     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstyp`.

```
3829   \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.

```
3830   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3831   \ifKV@glslink@local
3832     \glslocalunset{#2}%
3833   \else
3834     \glsunset{#2}%
3835   \fi
3836 }%
```

```
3837 \glspostlinkhook
3838 }
```

`\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`

```
3839 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3840 \newcommand*{\@Gls}[2] []{%
3841   \new@ifnextchar[\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}%
3842 }
```

`\@Gls@` Read in the final optional argument:

```
3843 \def\@Gls@#1#2[#3]{%
3844   \glsdoifexists{#2}%
3845   {%
3846     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3847 \let\glsifplural\@secondoftwo
3848 \let\glsifcaps\@secondofthree
3849 \let\glsifcustomtext\@empty
3850 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \glo@text) Note that \gls@link sets \glstyp.

```

3851 \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.

```

3852 \gls@link{#1}{#2}{\glo@text}%

```

Indicate that this entry has now been used

```

3853 \ifKV@glslink@local
3854 \glslocalunset{#2}%
3855 \else
3856 \glsunset{#2}%
3857 \fi
3858 }%

```

```

3859 \glspostlinkhook
3860 }

```

\GLS behaves like \gls, but the link text is converted to uppercase:

\GLS

```

3861 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

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

```

3862 \newcommand*{\@GLS}[2] [] {%
3863 \new@ifnextchar[\@GLS@{#1}{#2}]{\@GLS@{#1}{#2} []}%
3864 }

```

\@GLS@ Read in the final optional argument:

```

3865 \def\@GLS@#1#2[#3] {%
3866 \glsdoifexists{#2}%
3867 {%
3868 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3869 \let\glsifplural\@secondoftwo
3870 \let\glsifcaps\@thirdofthree
3871 \let\glsifcustomtext\@empty
3872 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \glo@text). Note that \gls@link sets \glstyp.

```

3873 \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.

```

3874 \gls@link{#1}{#2}{\glo@text}%

```


Indicate that this entry has now been used

```
3875 \ifKV@glslink@local
3876 \glsllocalunset{#2}%
3877 \else
3878 \glsunset{#2}%
3879 \fi
3880 }%

3881 \glspostlinkhook
3882 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3883 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3884 \newcommand*{\@glspl}[2][\@gls@hyp@opt\@glspl]{%
3885 \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}[]}%
3886 }
```

`\@glspl@` Read in the final optional argument:

```
3887 \def\@glspl@#1#2[#3]{%
3888 \glsoifexists{#2}%
3889 {%
3890 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3891 \let\glsifplural\@firstoftwo
3892 \let\glscapscase\@firstofthree
3893 \let\glscustomtext\@empty
3894 \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3895 \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.

```
3896 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3897 \ifKV@glslink@local
3898 \glsllocalunset{#2}%
3899 \else
3900 \glsunset{#2}%
3901 \fi
3902 }%

3903 \glspostlinkhook
3904 }
```

`\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`

```
3905 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3906 \newcommand*{\@Glspl}[2] [] {%
3907   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2} []}%
3908 }
```

`\@Glspl@` Read in the final optional argument:

```
3909 \def\@Glspl@#1#2[#3] {%
3910   \glsdoifexists{#2}%
3911   {%
3912     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3913     \let\glsifplural\@firstoftwo
3914     \let\glsapscase\@secondofthree
3915     \let\glscustomtext\@empty
3916     \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`.

```
3917   \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.

```
3918   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3919   \ifKV@glslink@local
3920     \glslocalunset{#2}%
3921   \else
3922     \glsunset{#2}%
3923   \fi
3924   }%
```

```
3925   \glspostlinkhook
3926 }
```

`\GLSp1` behaves like `\glsp1` except that all the link text is converted to uppercase.

`\GLSp1`

```
3927 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3928 \newcommand*{\@GLSp1}[2] [] {%
3929   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
3930 }
```

`\@GLSp1` Read in the final optional argument:

```
3931 \def\@GLSp1@#1#2[#3]{%
3932   \glsdoifexists{#2}%
3933   {%
3934     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3935     \let\glsifplural\@firstoftwo
3936     \let\glscapscase\@thirdofthree
3937     \let\glscustomtext\@empty
3938     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3939   \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.

```
3940   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3941   \ifKV@glslink@local
3942     \glslocalunset{#2}%
3943   \else
3944     \glsunset{#2}%
3945   \fi
3946 }%
```

```
3947 \glspostlinkhook
3948 }
```

`\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:

```
3949 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}
```

Defined the un-starred form.

`\@glsdisp`

```
3950 \newcommand*{\@glsdisp}[3][ ]{%
3951   \glsdoifexists{#2}{%

3952     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3953     \let\glsifplural\@secondoftwo
3954     \let\glscapscase\@firstofthree
3955     \def\glscustomtext{#3}%
3956     \def\glsinsert{}%
```

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@gls@link@local
3960 \glslocalunset{#2}%
3961 \else
3962 \glsunset{#2}%
3963 \fi
3964 }%
```

```
3965 \glspostlinkhook
3966 }
```

`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.

```
3967 \newcommand*{\@gls@link@nocheckfirsthyper}{}
```

`@gls@field@link`

```
3968 \newcommand{\@gls@field@link}[3]{%
3969 \glsdoifexists{#2}%
3970 {%
3971 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3972 \@gls@link[#1]{#2}{#3}%
3973 }%
3974 \glspostlinkhook
3975 }
```

`\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`

```
3976 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}
```

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

```
3977 \newcommand*{\@glstext}[2][{}]{%
3978 \new@ifnextchar[\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[{}]}}
```

Read in the final optional argument:

```
3979 \def\@glstext@#1#2[#3]{%
3980 \@gls@field@link{#1}{#2}{\glsentrytext{#2}#3}%
3981 }
```

`\GLStext` behaves like `\glstext` except the text is converted to uppercase.

`\GLStext`

```
3982 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3983 \newcommand*{\@GLStext}[2] [] {%
```

```
3984   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3985 \def\@GLStext@#1#2[#3] {%
```

```
3986   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstrytext{#2}#3}}%
```

```
3987 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
3988 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3989 \newcommand*{\@Glstext}[2] [] {%
```

```
3990   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3991 \def\@Glstext@#1#2[#3] {%
```

```
3992   \@gls@field@link{#1}{#2}{\glstrytext{#2}#3}}%
```

```
3993 }
```

`\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`

```
3994 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3995 \newcommand*{\@glsfirst}[2] [] {%
```

```
3996   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3997 \def\@glsfirst@#1#2[#3] {%
```

```
3998   \@gls@field@link{#1}{#2}{\glstryfirst{#2}#3}}%
```

```
3999 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
4000 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
4001 \newcommand*{\@Glsfirst}[2] [] {%
```

```
4002   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4003 \def\@Glsfirst@#1#2[#3]{%
4004   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
4005 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
4006 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
4007 \newcommand*{\@GLSfirst}[2] [] {%
4008   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4009 \def\@GLSfirst@#1#2[#3]{%
4010   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryfirst{#2}#3}}%
4011 }
```

\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

```
4012 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
4013 \newcommand*{\@glsplural}[2] [] {%
4014   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4015 \def\@glsplural@#1#2[#3]{%
4016   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
4017 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
4018 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
4019 \newcommand*{\@Glsplural}[2] [] {%
4020   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4021 \def\@Glsplural@#1#2[#3]{%
4022   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
4023 }
```

\GLSplural behaves like \glsplural except that the text is converted to uppercase.

\GLSplural

```
4024 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
4025 \newcommand*{\@GLSplural}[2] [] {%  
4026   \new@ifnextchar [{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4027 \def\@GLSplural@#1#2[#3] {%  
4028   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}{#3}}}%  
4029 }
```

\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

```
4030 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
4031 \newcommand*{\@glsfirstplural}[2] [] {%  
4032   \new@ifnextchar [{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4033 \def\@glsfirstplural@#1#2[#3] {%  
4034   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}{#3}}%  
4035 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
4036 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
4037 \newcommand*{\@Glsfirstplural}[2] [] {%  
4038   \new@ifnextchar [{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4039 \def\@Glsfirstplural@#1#2[#3] {%  
4040   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}{#3}}%  
4041 }
```

\GLSfirstplural behaves like \glsfirstplural except that the link text is converted to uppercase.

\GLSfirstplural

```
4042 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
4043 \newcommand*{\@GLSfirstplural}[2] [] {%  
4044   \new@ifnextchar [{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4045 \def\@GLSfirstplural@#1#2[#3] {%  
4046   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}{#3}}}%  
4047 }
```

`\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`

```
4048 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
4049 \newcommand*{\@glsname}[2] [] {%
```

```
4050   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4051 \def\@glsname@#1#2[#3] {%
```

```
4052   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
```

```
4053 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
4054 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
4055 \newcommand*{\@Glsname}[2] [] {%
```

```
4056   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4057 \def\@Glsname@#1#2[#3] {%
```

```
4058   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
```

```
4059 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
4060 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4061 \newcommand*{\@GLSname}[2] [] {%
```

```
4062   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4063 \def\@GLSname@#1#2[#3] {%
```

```
4064   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
```

```
4065 }
```

`\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`

```
4066 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
4067 \newcommand*{\@glsdesc}[2] [] {%
```

```
4068   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} [] }}
```


Read in the final optional argument:

```
4069 \def\@glsdesc#1#2[#3]{%
4070   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
4071 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
4072 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4073 \newcommand*{\@Glsdesc}[2][{}]{%
4074   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4075 \def\@Glsdesc#1#2[#3]{%
4076   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
4077 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
4078 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4079 \newcommand*{\@GLSdesc}[2][{}]{%
4080   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4081 \def\@GLSdesc#1#2[#3]{%
4082   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
4083 }
```

\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

```
4084 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4085 \newcommand*{\@glsdescplural}[2][{}]{%
4086   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4087 \def\@glsdescplural#1#2[#3]{%
4088   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
4089 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
4090 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4091 \newcommand*{\@Glsdescplural}[2] [] {%  
4092   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4093 \def\@Glsdescplural@#1#2[#3] {%  
4094   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%  
4095 }
```

\GLSdescplural behaves like \glsdescplural except that the link text is converted to uppercase.

\GLSdescplural

```
4096 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4097 \newcommand*{\@GLSdescplural}[2] [] {%  
4098   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4099 \def\@GLSdescplural@#1#2[#3] {%  
4100   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%  
4101 }
```

\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

```
4102 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
4103 \newcommand*{\@glssymbol}[2] [] {%  
4104   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4105 \def\@glssymbol@#1#2[#3] {%  
4106   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%  
4107 }
```

\Glssymbol behaves like \glssymbol except that the first letter is converted to uppercase.

\Glssymbol

```
4108 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4109 \newcommand*{\@Glssymbol}[2] [] {%  
4110   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4111 \def\@Glssymbol@#1#2[#3] {%  
4112   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}%  
4113 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
4114 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4115 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
4116   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4117 \def\@GLSsymbol@#1#2[#3] {%
```

```
4118   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}}%
```

```
4119 }
```

`\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`

```
4120 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4121 \newcommand*{\@glssymbolplural}[2] [] {%
```

```
4122   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4123 \def\@glssymbolplural@#1#2[#3] {%
```

```
4124   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}}%
```

```
4125 }
```

`\Glsymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`Glsymbolplural`

```
4126 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4127 \newcommand*{\@Glsymbolplural}[2] [] {%
```

```
4128   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4129 \def\@Glsymbolplural@#1#2[#3] {%
```

```
4130   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}}%
```

```
4131 }
```

`\GLSsymbolplural` behaves like `\glssymbolplural` except that the link text is converted to uppercase.

`GLSsymbolplural`

```
4132 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4133 \newcommand*{\@GLSsymbolplural}[2] [] {%
```

```
4134   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4135 \def\@GLSsymbolplural@#1#2[#3]{%
4136   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
4137 }
```

`\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`

```
4138 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4139 \newcommand*{\@glsuseri}[2] [] {%
4140   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4141 \def\@glsuseri@#1#2[#3]{%
4142   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
4143 }
```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```
4144 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4145 \newcommand*{\@Glsuseri}[2] [] {%
4146   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4147 \def\@Glsuseri@#1#2[#3]{%
4148   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
4149 }
```

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
4150 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4151 \newcommand*{\@GLSuseri}[2] [] {%
4152   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4153 \def\@GLSuseri@#1#2[#3]{%
4154   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
4155 }
```

`\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`

```
4156 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
4157 \newcommand*{\@glsuserii}[2] [] {%  
4158   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4159 \def\@glsuserii@#1#2[#3] {%  
4160   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%  
4161 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
4162 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4163 \newcommand*{\@Glsuserii}[2] [] {%  
4164   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4165 \def\@Glsuserii@#1#2[#3] {%  
4166   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%  
4167 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
4168 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
4169 \newcommand*{\@GLSuserii}[2] [] {%  
4170   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4171 \def\@GLSuserii@#1#2[#3] {%  
4172   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%  
4173 }
```

\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

```
4174 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4175 \newcommand*{\@glsuseriii}[2] [] {%  
4176   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4177 \def\@glsuseriii@#1#2[#3] {%  
4178   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%  
4179 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
4180 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4181 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
4182   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
4183 \def\@Glsuseriii@#1#2[#3] {%
```

```
4184   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
4185 }
```

\Glsuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
4186 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4187 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
4188   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
4189 \def\@GLSuseriii@#1#2[#3] {%
```

```
4190   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
```

```
4191 }
```

\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

```
4192 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4193 \newcommand*{\@glsuseriv}[2] [] {%
```

```
4194   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]
```

Read in the final optional argument:

```
4195 \def\@glsuseriv@#1#2[#3] {%
```

```
4196   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
4197 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\GLSuseriv

```
4198 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4199 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
4200   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}]
```

Read in the final optional argument:

```
4201 \def\@GLSuseriv@#1#2[#3] {%
```

```
4202   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
4203 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
4204 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4205 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
4206   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4207 \def\@GLSuseriv@#1#2[#3] {%
```

```
4208   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
```

```
4209 }
```

\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

```
4210 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4211 \newcommand*{\@glsuserv}[2] [] {%
```

```
4212   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4213 \def\@glsuserv@#1#2[#3] {%
```

```
4214   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}}%
```

```
4215 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
4216 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4217 \newcommand*{\@Glsuserv}[2] [] {%
```

```
4218   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4219 \def\@Glsuserv@#1#2[#3] {%
```

```
4220   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}}%
```

```
4221 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4222 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4223 \newcommand*{\@GLSuserv}[2] [] {%
```

```
4224   \new@ifnextchar[{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4225 \def\@GLSuserv@#1#2[#3]{%
4226   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4227 }
```

\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

```
4228 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4229 \newcommand*{\@glsuservi}[2][ ]{%
4230   \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
4231 \def\@glsuservi@#1#2[#3]{%
4232   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
4233 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4234 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4235 \newcommand*{\@Glsuservi}[2][ ]{%
4236   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
4237 \def\@Glsuservi@#1#2[#3]{%
4238   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
4239 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4240 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4241 \newcommand*{\@GLSuservi}[2][ ]{%
4242   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
4243 \def\@GLSuservi@#1#2[#3]{%
4244   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
4245 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4246 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```


Define the un-starred form. Need to determine if there is a final optional argument

```
4247 \newcommand*{\ns@acrshort}[2] [] {%
4248   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} [] }%
4249 }
```

Read in the final optional argument:

```
4250 \def\@acrshort#1#2[#3] {%
4251   \glsdoifexists{#2}%
4252   {%
4253     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4254     \let\glsifplural\@secondoftwo
4255     \let\glscapscase\@firstofthree
4256     \let\glsinsert\@empty
4257     \def\glscustomtext{%
4258       \acronymfont{\glstryshort{#2}}#3%
4259     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4260   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4261   }%
4262   \glspostlinkhook
4263 }
```

\Acrshort

```
4264 \newrobustcmd*{\Acrshort}{\@gl@hyp@opt\ns@Acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4265 \newcommand*{\ns@Acrshort}[2] [] {%
4266   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} [] }%
4267 }
```

Read in the final optional argument:

```
4268 \def\@Acrshort#1#2[#3] {%
4269   \glsdoifexists{#2}%
4270   {%
4271     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4272     \def\glslabel{#2}%
4273     \let\glsifplural\@secondoftwo
4274     \let\glscapscase\@secondofthree
4275     \let\glsinsert\@empty
4276     \def\glscustomtext{%
4277       \acronymfont{\Glsentryshort{#2}}#3%
4278     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4279   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4280   }%
```

```

4281 \glspostlinkhook
4282 }

```

\ACRshort

```

4283 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\@ns@ACRshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4284 \newcommand*{\ns@ACRshort}[2][\%
4285 \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
4286 }

```

Read in the final optional argument:

```

4287 \def\@ACRshort#1#2[#3]{%
4288 \glsdoifexists{#2}%
4289 {%
4290 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4291 \def\glslabel{#2}%
4292 \let\glsifplural\@secondoftwo
4293 \let\glsapscase\@thirdofthree
4294 \let\glsinsert\@empty
4295 \def\glscustomtext{%
4296 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4297 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

4298 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4299 }%
4300 \glspostlinkhook
4301 }

```

Short plural:

\acrshortpl

```

4302 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\@ns@acrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4303 \newcommand*{\ns@acrshortpl}[2][\%
4304 \new@ifnextchar[\@acrshortpl{#1}{#2}]{\@acrshortpl{#1}{#2}[]}%
4305 }

```

Read in the final optional argument:

```

4306 \def\@acrshortpl#1#2[#3]{%
4307 \glsdoifexists{#2}%
4308 {%
4309 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4310 \def\glslabel{#2}%
4311 \let\glsifplural\@firstoftwo
4312 \let\glsupcase\@firstofthree
4313 \let\glsinsert\@empty
4314 \def\glscustomtext{%
4315 \acronymfont{\glsentryshortpl{#2}}#3%
4316 }%

Call \@gls@link Note that \@gls@link sets \glstyp.
4317 \@gls@link[#1]{#2}{\csname gls@\glstyp @entryfmt\endcsname}%
4318 }%

4319 \glspostlinkhook
4320 }

```

\Acrshortpl

```

4321 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4322 \newcommand*{\ns@Acrshortpl}[2] [] {%
4323 \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
4324 }

Read in the final optional argument:
4325 \def\@Acrshortpl#1#2[#3] {%
4326 \glsdoifexists{#2}%
4327 {%

4328 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4329 \def\glslabel{#2}%
4330 \let\glsifplural\@firstoftwo
4331 \let\glsupcase\@secondofthree
4332 \let\glsinsert\@empty
4333 \def\glscustomtext{%
4334 \acronymfont{\Glsentryshortpl{#2}}#3%
4335 }%

Call \@gls@link Note that \@gls@link sets \glstyp.
4336 \@gls@link[#1]{#2}{\csname gls@\glstyp @entryfmt\endcsname}%
4337 }%

4338 \glspostlinkhook
4339 }

```

\ACRshortpl

```

4340 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4341 \newcommand*{\ns@ACRshortpl}[2] [] {%
4342 \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
4343 }

```

Read in the final optional argument:

```
4344 \def\@ACRshortpl#1#2[#3]{%
4345   \glsdoifexists{#2}%
4346   {%
4347     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4348     \def\glslabel{#2}%
4349     \let\glsifplural\@firstoftwo
4350     \let\glscapscase\@thirdofthree
4351     \let\glsinsert\@empty
4352     \def\glscustomtext{%
4353       \mfirstucMakeUppercase{\acronymfont{\glentryshortpl{#2}}#3}%
4354     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4355   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4356   }%
4357   \glspostlinkhook
4358 }
```

\acrlong

```
4359 \newrobustcmd*{\acrlong}{\@gl@hyp@opt\@ns@acrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4360 \newcommand*{\@ns@acrlong}[2][ ]{%
4361   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[ ]}%
4362 }
```

Read in the final optional argument:

```
4363 \def\@acrlong#1#2[#3]{%
4364   \glsdoifexists{#2}%
4365   {%
4366     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4367     \def\glslabel{#2}%
4368     \let\glsifplural\@secondoftwo
4369     \let\glscapscase\@firstofthree
4370     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4371   \def\glscustomtext{%
4372     \glentrylong{#2}#3%
4373   }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4374   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4375   }%
```

```

4376 \glspostlinkhook
4377 }

```

\Acrlong

```

4378 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4379 \newcommand*{\ns@Acrlong}[2][ ]{%
4380   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[ ]}%
4381 }

```

Read in the final optional argument:

```

4382 \def\@Acrlong#1#2[#3]{%
4383   \glsdoifexists{#2}%
4384   {%
4385     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4386     \def\glslabel{#2}%
4387     \let\glsifplural\@secondoftwo
4388     \let\glsapscase\@secondofthree
4389     \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4390   \def\glscustomtext{%
4391     \Glsentrylong{#2}#3%
4392   }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4393   \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4394   }%
4395   \glspostlinkhook
4396 }

```

\ACRlong

```

4397 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\ns@ACRlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4398 \newcommand*{\ns@ACRlong}[2][ ]{%
4399   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[ ]}%
4400 }

```

Read in the final optional argument:

```

4401 \def\@ACRlong#1#2[#3]{%
4402   \glsdoifexists{#2}%
4403   {%
4404     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4405 \def\glslabel{#2}%
4406 \let\glsifplural\@secondoftwo
4407 \let\glsapscase\@thirdofthree
4408 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4409 \def\glscustomtext{%
4410 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4411 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4412 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4413 }%

```

```

4414 \glspostlinkhook
4415 }

```

Short plural:

\acrlongpl

```

4416 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4417 \newcommand*{\ns@acrlongpl}[2][\%
4418 \new@ifnextchar[\@acrlongpl{#1}{#2}]{\@acrlongpl{#1}{#2}[]}%
4419 }

```

Read in the final optional argument:

```

4420 \def\@acrlongpl#1#2[#3]{%
4421 \glsdoifexists{#2}%
4422 {%
4423 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4424 \def\glslabel{#2}%
4425 \let\glsifplural\@firstoftwo
4426 \let\glsapscase\@firstofthree
4427 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4428 \def\glscustomtext{%
4429 \glsentrylongpl{#2}#3%
4430 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4431 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4432 }%

```

```

4433 \glspostlinkhook
4434 }

```

\Acrlongpl

```
4435 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4436 \newcommand*{\ns@Acrlongpl}[2][\%  
4437 \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%  
4438 }
```

Read in the final optional argument:

```
4439 \def\@Acrlongpl#1#2[#3]{%  
4440 \glsdoifexists{#2}%  
4441 {%  
  
4442 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4443 \def\glslabel{#2}%  
4444 \let\glsifplural\@firstoftwo  
4445 \let\glsapscase\@secondofthree  
4446 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4447 \def\glscustomtext{%  
4448 \Glsentrylongpl{#2}#3%  
4449 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4450 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
4451 }%  
  
4452 \glspostlinkhook  
4453 }
```

\ACRlongpl

```
4454 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\ns@ACRlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4455 \newcommand*{\ns@ACRlongpl}[2][\%  
4456 \new@ifnextchar[\@ACRlongpl{#1}{#2}]{\@ACRlongpl{#1}{#2}[]}%  
4457 }
```

Read in the final optional argument:

```
4458 \def\@ACRlongpl#1#2[#3]{%  
4459 \glsdoifexists{#2}%  
4460 {%  
  
4461 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4462 \def\glslabel{#2}%  
4463 \let\glsifplural\@firstoftwo  
4464 \let\glsapscase\@thirdofthree  
4465 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4466 \def\glscustomtext{%
4467     \mfirstucMakeUppercase{\glsenrylongpl{#2}#3}%
4468 }%

Call \@gls@link. Note that \@gls@link sets \glstype.
4469 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4470 }%

4471 \glspostlinkhook
4472 }

```

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>}`

```

4473 \newcommand*{\@gls@entry@field}[2]{%
4474     \csname glo@\glsdetoklabel{#1}@#2\endcsname
4475 }

```

`\glsletentryfield` `\glsletentryfield{<cs>}{<label>}{<field>}`

```

4476 \newcommand*{\glsletentryfield}[3]{%
4477     \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
4478 }

```

`\Gls@entry@field` Generic first letter uppercase version.

`\@Gls@entry@field{<label>}{<field>}`

```

4479 \newcommand*{\@Gls@entry@field}[2]{%
4480     \glsdoifexistsordo{#1}%
4481     {%
4482         \letcs{\@glo@text}{glo@\glsdetoklabel{#1}@#2}%
4483         \ifdef\@glo@text
4484         {%
4485             \xmakefirstuc{\@glo@text}%
4486         }%
4487     }%
4488     ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4489     entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry

```



```

4490     label and the field name}%
4491   }%
4492 }%
4493 {%
4494   ??%
4495 }%
4496 }

```

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`

```

4497 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4498 \newrobustcmd*{\Glsentryname}[1]{%
4499   \@Gls@entryname{#1}%
4500 }

```

`\@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:

```

4501 \newcommand*{\@Gls@entryname}[1]{%
4502   \@Gls@entry@field{#1}{name}%
4503 }

```

`\ls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

4504 \newcommand*{\@Gls@acrentryname}[1]{%
4505   \ifglshaslong{#1}%
4506   {%
4507     \letcs\@glo@text{glo\@glsdetoklabel{#1}@name}%
4508     \expandafter\@gls@getbody\@glo@text{}\@nil
4509     \expandafter\ifx\@gls@body\glsentrylong\relax
4510     \expandafter\Glsentrylong\@gls@rest
4511   \else
4512     \expandafter\ifx\@gls@body\glsentryshort\relax
4513     \expandafter\Glsentryshort\@gls@rest
4514   \else
4515     \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.

```

4516     {%
4517       \let\glsentryshort\Glsentryshort
4518       \@glo@text
4519     }%
4520   \else

```

```

4521      \xmakefirstuc{\@glo@text}%
4522      \fi
4523      \fi
4524      \fi
4525  }%
4526  {%

```

Not an acronym

```

4527      \@Gls@entry@field{#1}{name}%
4528  }%
4529 }

```

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.

`\glentrydesc`

```

4530 \newcommand*{\glentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4531 \newrobustcmd*{\Glsentrydesc}[1]{%
4532   \@Gls@entry@field{#1}{desc}%
4533 }

```

Plural form:

`entrydescplural`

```

4534 \newcommand*{\glentrydescplural}[1]{%
4535   \@gls@entry@field{#1}{descplural}%
4536 }

```

`entrydescplural`

```

4537 \newrobustcmd*{\Glsentrydescplural}[1]{%
4538   \@Gls@entry@field{#1}{descplural}%
4539 }

```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

`\glentrytext`

```

4540 \newcommand*{\glentrytext}[1]{\@gls@entry@field{#1}{text}}

```

`\Glsentrytext`

```

4541 \newrobustcmd*{\Glsentrytext}[1]{%
4542   \@Gls@entry@field{#1}{text}%
4543 }

```

Get the plural form:

`\glentryplural`

```
4544 \newcommand*{\glentryplural}[1]{%
4545   \@gls@entry@field{#1}{plural}%
4546 }
```

`\Glsentryplural`

```
4547 \newrobustcmd*{\Glsentryplural}[1]{%
4548   \@Gls@entry@field{#1}{plural}%
4549 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glentrysymbol`

```
4550 \newcommand*{\glentrysymbol}[1]{%
4551   \@gls@entry@field{#1}{symbol}%
4552 }
```

`\Glsentrysymbol`

```
4553 \newrobustcmd*{\Glsentrysymbol}[1]{%
4554   \@Gls@entry@field{#1}{symbol}%
4555 }
```

Plural form:

`trysymbolplural`

```
4556 \newcommand*{\glentrysymbolplural}[1]{%
4557   \@gls@entry@field{#1}{symbolplural}%
4558 }
```

`trysymbolplural`

```
4559 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4560   \@Gls@entry@field{#1}{symbolplural}%
4561 }
```

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).

`\glentryfirst`

```
4562 \newcommand*{\glentryfirst}[1]{%
4563   \@gls@entry@field{#1}{first}%
4564 }
```

`\Glsentryfirst`

```
4565 \newrobustcmd*{\Glsentryfirst}[1]{%
4566   \@Gls@entry@field{#1}{first}%
4567 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

entryfirstplural

```
4568 \newcommand*{\glsentryfirstplural}[1]{%
4569   \@gls@entry@field{#1}{firstpl}%
4570 }
```

entryfirstplural

```
4571 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4572   \@Gls@entry@field{#1}{firstpl}%
4573 }
```

sentrytitlecase

```
4574 \newrobustcmd*{@glsentrytitlecase}[2]{%
4575   \glsfieldfetch{#1}{#2}{\@gls@value}%
4576   \xcapitalisewords{\@gls@value}%
4577 }
4578 \ifdef\texorpdfstring
4579 {
4580   \newcommand*{\glsentrytitlecase}[2]{%
4581     \texorpdfstring
4582       {\@glsentrytitlecase{#1}{#2}}%
4583       {\@gls@entry@field{#1}{#2}}%
4584   }
4585 }
4586 {
4587   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4588 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
4589 \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.

\glsentrysort

```
4590 \newcommand*{\glsentrysort}[1]{%
4591   \@gls@entry@field{#1}{sort}%
4592 }
```

\glsentryuseri Get the first user key (as specified by the user1 when the entry was defined). The argument is the label associated with the entry.

```
4593 \newcommand*{\glsentryuseri}[1]{%
4594   \@gls@entry@field{#1}{useri}%
4595 }
```

\Glsentryuseri

```
4596 \newrobustcmd*{\Glsentryuseri}[1]{%
```

```

4597 \@Gls@entry@field{#1}{useri}%
4598 }

```

`\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.

```

4599 \newcommand*{\glentryuserii}[1]{%
4600 \@Gls@entry@field{#1}{userii}%
4601 }

```

`\Glsentryuserii`

```

4602 \newrobustcmd*{\Glsentryuserii}[1]{%
4603 \@Gls@entry@field{#1}{userii}%
4604 }

```

`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.

```

4605 \newcommand*{\glentryuseriii}[1]{%
4606 \@Gls@entry@field{#1}{useriii}%
4607 }

```

`Glsentryuseriii`

```

4608 \newrobustcmd*{\Glsentryuseriii}[1]{%
4609 \@Gls@entry@field{#1}{useriii}%
4610 }

```

`\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.

```

4611 \newcommand*{\glentryuseriv}[1]{%
4612 \@Gls@entry@field{#1}{useriv}%
4613 }

```

`\Glsentryuseriv`

```

4614 \newrobustcmd*{\Glsentryuseriv}[1]{%
4615 \@Gls@entry@field{#1}{useriv}%
4616 }

```

`\glentryuserv` Get the fifth user key (as specified by the user5 when the entry was defined). The argument is the label associated with the entry.

```

4617 \newcommand*{\glentryuserv}[1]{%
4618 \@Gls@entry@field{#1}{userv}%
4619 }

```

`\Glsentryuserv`

```

4620 \newrobustcmd*{\Glsentryuserv}[1]{%
4621 \@Gls@entry@field{#1}{userv}%
4622 }

```

`\glentryuservi` Get the sixth user key (as specified by the user6 when the entry was defined). The argument is the label associated with the entry.

```
4623 \newcommand*{\glentryuservi}[1]{%
4624   \@gls@entry@field{#1}{uservi}%
4625 }
```

`\Glsentryuservi`

```
4626 \newrobustcmd*{\Glsentryuservi}[1]{%
4627   \@Gls@entry@field{#1}{uservi}%
4628 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4629 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4630 \newrobustcmd*{\Glsentryshort}[1]{%
4631   \@Gls@entry@field{#1}{short}%
4632 }
```

`glentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4633 \newcommand*{\glentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`Glsentryshortpl`

```
4634 \newrobustcmd*{\Glsentryshortpl}[1]{%
4635   \@Gls@entry@field{#1}{shortpl}%
4636 }
```

`\glentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4637 \newcommand*{\glentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4638 \newrobustcmd*{\Glsentrylong}[1]{%
4639   \@Gls@entry@field{#1}{long}%
4640 }
```

`\glentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4641 \newcommand*{\glentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4642 \newrobustcmd*{\Glsentrylongpl}[1]{%
4643   \@Gls@entry@field{#1}{longpl}%
4644 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4645 \newcommand*{\glsentryfull}[1]{%
4646   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4647 }
```

`\Glsentryfull`

```
4648 \newrobustcmd*{\Glsentryfull}[1]{%
4649   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4650 }
```

`\glsentryfullpl`

```
4651 \newcommand*{\glsentryfullpl}[1]{%
4652   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4653 }
```

`\Glsentryfullpl`

```
4654 \newrobustcmd*{\Glsentryfullpl}[1]{%
4655   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4656 }
```

`entrynumberlist` Displays the number list as is.

```
4657 \newcommand*{\glsentrynumberlist}[1]{%
4658   \glsdoifexists{#1}%
4659   {%
4660     \@gls@entry@field{#1}{numberlist}%
4661   }%
4662 }
```

`splaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4663 \@ifpackageloaded{hyperref} {%
4664   \newcommand*{\glsdisplaynumberlist}[1]{%
4665     \GlossariesWarning
4666     {%
4667       \string\glsdisplaynumberlist\space
4668       doesn't work with hyperref.^^JUsing
4669       \string\glsentrynumberlist\space instead%
4670     }%
4671     \glsentrynumberlist{#1}%
4672   }%
4673 }%
4674 {%
4675   \newcommand*{\glsdisplaynumberlist}[1]{%
4676     \glsdoifexists{#1}%
4677     {%
4678       \bgroup
```



```

4714 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4715 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

This key is only used by `\glsaddall`:

```

4716 \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`

```

4717 \newrobustcmd*{\glsadd}[2] [] {%

```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4718   \@gls@adjustmode
4719   \glsdoifexists{#2}%
4720   {%
4721     \def\@glsnumberformat{glsnumberformat}%
4722     \edef\@gls@counter{\csname glo@glstetoklabel{#2}@counter\endcsname}%
4723     \setkeys{glossadd}{#1}%

```

Store the entry's counter in `\theglsentrycounter`

```

4724     \@gls@saveentrycounter

```

Define sort key if necessary:

```

4725     \@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.

```

4726     \@@do@wrglossary{#2}%
4727   }%
4728 }

```

`@gls@adjustmode`

```

4729 \newcommand*{\@gls@adjustmode}{}
4730 \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`

```

4731 \newrobustcmd*{\glsaddall}[1] [] {%
4732   \edef\@glo@type{\@glo@types}%

```

```

4733 \setkeys{glossadd}{#1}%
4734 \forallglsentries[\@glo@type]{\@glo@entry}{%
4735   \glsadd[#1]{\@glo@entry}%
4736 }%
4737 }

```

```

\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.

```

4738 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4739   \forallglsentries[#1]{\@glo@entry}%
4740   {%
4741     \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4742   }%
4743 }

```

`\glsignore`

```

4744 \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 `\glsgroupname` 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.

```

4745 \edef\glsopenbrace{\expandafter\@gobble\string\{ }

```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```

4746 \edef\glsclosebrace{\expandafter\@gobble\string\} }

```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4747 \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.

```
4748 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4749 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4750 \edef\glstildechar{\string~}
```

`@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4751 \ifglsxindy
4752   \newcommand*{\@glsfirstletter}{A}
4753 \fi
```

`letterAfterDigits` Sets the first letter to come after the digits 0,...,9. The starred version sanitizes.

```
4754 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}{%
4755   \@ifstar\s@GlsSetXdyFirstLetterAfterDigits\@GlsSetXdyFirstLetterAfterDigits}
4756 \ifglsxindy
4757   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4758     \renewcommand*{\@glsfirstletter}{#1}}
4759   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}[1]{%
4760     \renewcommand*{\@glsfirstletter}{#1}%
4761     \@onelevel@sanitize\@glsfirstletter
4762   }
4763 \else
4764   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4765     \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4766   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}{%
4767     \@GlsSetXdyFirstLetterAfterDigits
4768   }
4769 \fi
```

`numbergrouporder` Specifies the order of the number group.

```
4770 \ifglsxindy
4771   \newcommand*{\@xdynumbergrouporder}{:before \string"\@glsfirstletter\string"}
4772 \fi
```

`numberGroupOrder` Sets the relative location of the number group. The starred version sanitizes.

```
4773 \newcommand*{\GlsSetXdyNumberGroupOrder}[1]{%
4774   \@ifstar\s@GlsSetXdyNumberGroupOrder\@GlsSetXdyNumberGroupOrder
4775 }
4776 \ifglsxindy
4777   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4778     \renewcommand*{\@xdynumbergrouporder}{#1}%
4779   }
4780 \fi
```

```

4779 }
4780 \newcommand*{\s@GlsSetXdyNumberGroupOrder}[1]{%
4781   \renewcommand*{\@xdynumbergrouporder}{#1}%
4782   \@onelevel@sanitize\@xdynumbergrouporder
4783 }
4784 \else
4785   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4786     \glsnocindywarning\GlsSetXdyNumberGroupOrder}
4787   \newcommand*{\s@GlsSetXdyNumberGroupOrder}{%
4788     \@GlsSetXdyNumberGroupOrder}
4789 \fi

```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```

4790 \newcommand*{\@glsminrange}{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.

```

4791 \ifglsxindy
4792   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4793     \renewcommand*{\@glsminrange}{#1}}
4794 \else
4795   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4796     \glsnocindywarning\GlsSetXdyMinRangeLength}
4797 \fi

```

`\writeist`

```

4798 \ifglsxindy

```

Code to use if xindy is required.

```

4799 \def\writeist{%

```

Define write register if not already defined

```

4800   \ifundef{\glswrite}{\newwrite\glswrite}{}%

```

Update attributes list

```

4801   \@gls@addpredefinedattributes

```

Open the file.

```

4802   \openout\glswrite=\istfilename

```

Write header comment at the start of the file

```

4803   \write\glswrite{;; xindy style file created by the glossaries
4804     package}%
4805   \write\glswrite{;; for document '\jobname' on
4806     \the\year-\the\month-\the\day}%

```

Specify the required styles

```

4807   \write\glswrite{^^J; required styles^^J}
4808   \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4809     \ifx\@xdystyle\@empty
4810       \else

```

```

4811         \protected@write\glswrite{}\{(require
4812             \string"\@xdystyle.xdy\string")}\}%
4813     \fi
4814 }%

```

List the allowed attributes (possible values used by the format key)

```

4815     \write\glswrite{^^J%
4816         ; list of allowed attributes (number formats)^^J}%
4817     \write\glswrite{(define-attributes ((\@xdyattributes)))}%

```

Define any additional alphabets

```

4818     \write\glswrite{^^J; user defined alphabets^^J}%
4819     \write\glswrite{\@xdyuseralphabets}%

```

Define location classes.

```

4820     \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.

```

4821     \@for\@gls@classI:=\@gls@xdy@locationlist\do{%

```

Case where $\langle Hprefix \rangle$ is empty:

```

4822         \protected@write\glswrite{}\{(define-location-class
4823             \string"\@gls@classI\string"^^J\space\space\space
4824             (
4825                 :sep "{"
4826                 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4827                 :sep "}"
4828             )
4829             ^^J\space\space\space
4830             :min-range-length \@glsminrange^^J%
4831         )
4832     }%

```

Nested iteration over all classes:

```

4833     {%
4834         \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4835             \protected@write\glswrite{}\{(define-location-class
4836                 \string"\@gls@classII-\@gls@classI\string"
4837                 ^^J\space\space\space
4838                 (
4839                     :sep "{"
4840                     \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4841                     :sep "{"
4842                     \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4843                     :sep "}"
4844                 )
4845                 ^^J\space\space\space
4846                 :min-range-length \@glsminrange^^J%
4847             )
4848         }%
4849     }%

```

```

4850     }%
4851 }%

```

User defined location classes (needs checking for new location format).

```

4852 \write\glswrite{^^J; user defined location classes}%
4853 \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.)

```

4854 \write\glswrite{^^J; define cross-reference class^^J}%
4855 \write\glswrite{(define-crossref-class \string"see\string"
4856 :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.)

```

4857 \write\glswrite{(markup-crossref-list
4858 :class \string"see\string"^^J\space\space\space
4859 :open \string"\string\glsseeformat\string"
4860 :close \string"{}\string")}%

```

Provide hook to write extra material here (used by glossaries-extra to define a seealso class).

```

4861 \@xdycrossrefhook

```

List the order to sort the classes.

```

4862 \write\glswrite{^^J; define the order of the location classes}%
4863 \write\glswrite{(define-location-class-order
4864 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4865 \write\glswrite{^^J; define the glossary markup^^J}%
4866 \write\glswrite{(markup-index^^J\space\space\space
4867 :open \string"\string
4868 \glossarysection[\string\glossarytoctitle]{\string
4869 \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)

```

4870 \@for\@this@ctr:=\@xdycounters\do{%
4871   {%
4872     \@for\@this@attr:=\@xdyattributelist\do{%
4873       \protected@write\glswrite{}\string\providecommand*%
4874       \expandafter\string
4875       \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4876       {%
4877         \string\setentrycounter
4878         [\expandafter\@gobble\string\#1]{\@this@ctr}%
4879         \expandafter\string

```

```

4880             \csname\@this@attr\endcsname
4881             {\expandafter\@gobble\string\#2}%
4882         }%
4883     }%
4884 }%
4885 }%
4886 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4887 \write\glswrite{%
4888     \string\begin
4889     {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4890     \space\space:close \string"\glpercentchar\glstildechar n\string
4891     \end{theglossary}\string\glossarypostamble
4892     \glstildechar n\string" ^^J\space\space\space
4893     :tree)}}%

```

Specify what to put between letter groups

```

4894 \write\glswrite{(markup-letter-group-list
4895     :sep \string"\string\glsgroupskip\glstildechar n\string"))}%

```

Specify what to put between entries

```

4896 \write\glswrite{(markup-indexentry
4897     :open \string"\string\relax \string\glresetentrylist
4898     \glstildechar n\string)}}%

```

Specify how to format entries

```

4899 \write\glswrite{(markup-locclass-list :open
4900     \string"\glsoopenbrace\string\glossaryentrynumbers
4901     \glsoopenbrace\string\relax\space \string"^^J\space\space\space
4902     :sep \string", \string"
4903     :close \string"\glsclosebrace\glsclosebrace\string"))}%

```

Specify how to separate location numbers

```

4904 \write\glswrite{(markup-locref-list
4905     :sep \string"\string\delimN\space\string"))}%

```

Specify how to indicate location ranges

```

4906 \write\glswrite{(markup-range
4907     :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.

```

4908 \@onelevel@sanitize\gls@suffiF
4909 \@onelevel@sanitize\gls@suffiFF
4910 \ifx\gls@suffiF\@empty
4911 \else
4912     \write\glswrite{(markup-range
4913         :close "\gls@suffiF" :length 1 :ignore-end)}}%
4914 \fi
4915 \ifx\gls@suffiFF\@empty

```

```

4916     \else
4917         \write\glswrite{(markup-range
4918             :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4919     \fi

Specify how to format locations.
4920     \write\glswrite{^^J; define format to use for locations^^J}%
4921     \write\glswrite{\@xdylocref}%

Specify how to separate letter groups.
4922     \write\glswrite{^^J; define letter group list format^^J}%
4923     \write\glswrite{(markup-letter-group-list
4924         :sep \string"\string\glsgroupskip\glstildechar n\string")}}%

Define letter group headings.
4925     \write\glswrite{^^J; letter group headings^^J}%
4926     \write\glswrite{(markup-letter-group
4927         :open-head \string"\string\glsgroupheading
4928         \glsopenbrace\string"^^J\space\space\space
4929         :close-head \string"\glsclosebrace\string")}}%

Define additional letter groups.
4930     \write\glswrite{^^J; additional letter groups^^J}%
4931     \write\glswrite{\@xdylettergroups}%

Define additional sort rules
4932     \write\glswrite{^^J; additional sort rules^^J}%
4933     \write\glswrite{\@xdysortrules}%

Hook for any additional information:
4934     \@gls@writeisthook

Close the style file
4935     \closeout\glswrite

Suppress any further calls.
4936     \let\writeist\relax
4937 }
4938 \else

Code to use if makeindex is required.
4939 \edef\@gls@actualchar{\string?}
4940 \edef\@gls@encapchar{\string|}
4941 \edef\@gls@levelchar{\string!}
4942 \edef\@gls@quotechar{\string"}%
4943 \let\GlsSetQuote\gls@nosetquote
4944 \def\writeist{\relax
4945     \ifundef{\glswrite}{\newwrite\glswrite}{\relax
4946     \openout\glswrite=\istfilename
4947     \write\glswrite{\glspercentchar\space makeindex style file
4948         created by the glossaries package}
4949     \write\glswrite{\glspercentchar\space for document
4950         '\jobname' on \the\year-\the\month-\the\day}

```



```

4951 \write\glswrite{actual '@gls@actualchar'}
4952 \write\glswrite{encap '@gls@encapchar'}
4953 \write\glswrite{level '@gls@levelchar'}
4954 \write\glswrite{quote '@gls@quotechar'}
4955 \write\glswrite{keyword \string\string\glossaryentry\string}
4956 \write\glswrite{preamble \string\string\glossarysection[\string
4957 \glossarytoctitle]{\string\glossarytitle}\string
4958 \glossarypreamble\string\n\string\begin{theglossary}\string
4959 \glossaryheader\string\n\string}
4960 \write\glswrite{postamble \string\string%\string\n\string
4961 \end{theglossary}\string\glossarypostamble\string\n
4962 \string}
4963 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
4964 \string}
4965 \write\glswrite{item_0 \string\string%\string\n\string}
4966 \write\glswrite{item_1 \string\string%\string\n\string}
4967 \write\glswrite{item_2 \string\string%\string\n\string}
4968 \write\glswrite{item_01 \string\string%\string\n\string}
4969 \write\glswrite{item_x1
4970 \string\string\relax \string\glsresetentrylist\string\n
4971 \string}
4972 \write\glswrite{item_12 \string\string%\string\n\string}
4973 \write\glswrite{item_x2
4974 \string\string\relax \string\glsresetentrylist\string\n
4975 \string}

4976 \write\glswrite{delim_0 \string\string\{\string
4977 \glossaryentrynumbers\string\{\string\relax \string}
4978 \write\glswrite{delim_1 \string\string\{\string
4979 \glossaryentrynumbers\string\{\string\relax \string}
4980 \write\glswrite{delim_2 \string\string\{\string
4981 \glossaryentrynumbers\string\{\string\relax \string}
4982 \write\glswrite{delim_t \string\string\}\string\}\string}
4983 \write\glswrite{delim_n \string\string\delimN \string}
4984 \write\glswrite{delim_r \string\string\delimR \string}
4985 \write\glswrite{headings_flag 1}
4986 \write\glswrite{heading_prefix
4987 \string\string\glsgroupheading\string\{\string}
4988 \write\glswrite{heading_suffix
4989 \string\string\}\string\relax
4990 \string\glsresetentrylist \string}
4991 \write\glswrite{symhead_positive \string\glsymbols\string}
4992 \write\glswrite{numhead_positive \string\glsnumbers\string}
4993 \write\glswrite{page_compositor \string\glscompositor\string}
4994 \@gls@escbsdq\gls@suffixF
4995 \@gls@escbsdq\gls@suffixFF
4996 \ifx\gls@suffixF\@empty
4997 \else
4998 \write\glswrite{suffix_2p \string\gls@suffixF\string}
4999 \fi

```

```

5000 \ifx\gls@suffixFF\@empty
5001 \else
5002 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
5003 \fi

```

Hook for any additional information:

```
5004 \@gls@writeisthook
```

Close the file and disable \writeist.

```

5005 \closeout\glswrite
5006 \let\writeist\relax
5007 }
5008 \fi

```

SetWriteIstHook Allow user to append information to the style file.

```

5009 \newcommand*{\GlsSetWriteIstHook}[1]{\renewcommand*{\@gls@writeisthook}{#1}}
5010 \@onlypremakeg\GlsSetWriteIstHook

```

ls@writeisthook

```
5011 \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.

```

5012 \ifglxsindy
5013 \newcommand*{\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
5014 \newcommand*{\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
5015 \else
5016 \newcommand*{\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}%

```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```

5017 \@ifpackageloaded{tracklang}%
5018 {%
5019 \IfTrackedLanguage{german}%
5020 {%
5021 \def\@gls@extramakeindexopts{-g}%
5022 }%
5023 }%
5024 }%
5025 {}%

```

Need to redefine \@gls@checkquote

```

5026 \edef\@gls@docheckquotedef{%
5027 \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
5028 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5029 \noexpand\toks@={####1}%
5030 \noexpand\ifx\noexpand\null####2\noexpand\null
5031 \noexpand\ifx\noexpand\null####3\noexpand\null
5032 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5033 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5034 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%

```

```

5035 \noexpand\else
5036 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5037 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5038 \noexpand\@gls@quotetchar\noexpand\@gls@quotetchar
5039 \noexpand\@gls@quotetchar\noexpand\@gls@quotetchar}%
5040 \noexpand\def\noexpand\@gls@checkquote{%
5041 \noexpand\@gls@checkquote####3\noexpand\null}%
5042 \noexpand\fi
5043 \noexpand\else
5044 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5045 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5046 \noexpand\@gls@quotetchar\noexpand\@gls@quotetchar}%
5047 \noexpand\ifx\noexpand\null####3\noexpand\null
5048 \noexpand\def\noexpand\@gls@checkquote{%
5049 \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
5050 \noexpand\else
5051 \noexpand\def\noexpand\@gls@checkquote{%
5052 \noexpand\@gls@checkquote####2#1####3\noexpand\null}%
5053 \noexpand\fi
5054 \noexpand\fi
5055 \noexpand\@gls@checkquote
5056 }%
5057 }%
5058 \@gls@docheckquotedef
5059 \edef\@gls@docheckquotedef{%
5060 \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
5061 \noexpand\def\noexpand\@gls@checkedmkidx{%
5062 \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
5063 #1#1\noexpand\null
5064 \noexpand\expandafter\noexpand\@gls@updatechecked
5065 \noexpand\@gls@checkedmkidx{####1}%
5066 \noexpand\def\noexpand\@gls@checkedmkidx{%
5067 \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
5068 \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
5069 \noexpand\null
5070 \noexpand\expandafter\noexpand\@gls@updatechecked
5071 \noexpand\@gls@checkedmkidx{####1}%
5072 \noexpand\def\noexpand\@gls@checkedmkidx{%
5073 \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
5074 \noexpand\?\noexpand\?\noexpand\null
5075 \noexpand\expandafter\noexpand\@gls@updatechecked
5076 \noexpand\@gls@checkedmkidx{####1}%
5077 \noexpand\def\noexpand\@gls@checkedmkidx{%
5078 \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
5079 \noexpand\?\noexpand\?\noexpand\null
5080 \noexpand\expandafter\noexpand\@gls@updatechecked
5081 \noexpand\@gls@checkedmkidx{####1}%
5082 \noexpand\def\noexpand\@gls@checkedmkidx{%
5083 \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil

```

```

5084         \noexpand|\noexpand|\noexpand\null
5085     \noexpand\expandafter\noexpand\@gls@updatechecked
5086         \noexpand\@gls@checkedmkidx{####1}%
5087     \noexpand\def\noexpand\@gls@checkedmkidx{%
5088     \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
5089         \noexpand\\ \noexpand\\ \noexpand\null
5090     \noexpand\expandafter\noexpand\@gls@updatechecked
5091         \noexpand\@gls@checkedmkidx{####1}%
5092     \noexpand\def\noexpand\@gls@checkedmkidx{%
5093     \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
5094         \noexpand!\noexpand!\noexpand\null
5095     \noexpand\expandafter\noexpand\@gls@updatechecked
5096         \noexpand\@gls@checkedmkidx{####1}%
5097     }%
5098 }%
5099 \@gls@docheckquotedef
5100 \edef\@gls@docheckquotedef{%
5101     \noexpand\def\noexpand\@gls@checkescquote####1%
5102     \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
5103     ####3\noexpand\null{%
5104     \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5105     \noexpand\toks@={####1}%
5106     \noexpand\ifx\noexpand\null####2\noexpand\null
5107     \noexpand\ifx\noexpand\null####3\noexpand\null
5108     \noexpand\edef\noexpand\@gls@checkedmkidx{%
5109         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5110     \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
5111     \noexpand\else
5112     \noexpand\edef\noexpand\@gls@checkedmkidx{%
5113         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5114         \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5115             \csname#1\endcsname}\noexpand\@gls@quotechar
5116         \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5117             \csname#1\endcsname}\noexpand\@gls@quotechar}%
5118     \noexpand\def\noexpand\@gls@checkescquote{%
5119         \noexpand\@gls@checkescquote####3\noexpand\null}%
5120     \noexpand\fi
5121     \noexpand\else
5122     \noexpand\edef\noexpand\@gls@checkedmkidx{%
5123         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5124         \noexpand\@gls@quotechar\noexpand\string
5125         \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
5126     \noexpand\ifx\noexpand\null####3\noexpand\null
5127     \noexpand\def\noexpand\@gls@checkescquote{%
5128         \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5129         \expandonce{\csname#1\endcsname}\noexpand\null}%
5130     \noexpand\else
5131     \noexpand\def\noexpand\@gls@checkescquote{%
5132         \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%

```

```

5133         ###3\noexpand\null}%
5134     \noexpand\fi
5135     \noexpand\fi
5136     \noexpand\@@gls@checkescquote
5137 }%
5138 }%
5139 \@gls@docheckquotedef
5140 }
5141 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
5142 {\string\GlsSetQuote\space not permitted here}%
5143 {Move \string\GlsSetQuote\space earlier in the preamble, as
5144  soon as possible after glossaries.sty has been loaded}}
5145 \fi

```

ramakeindexopts

```

5146 \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`

```

5147 \newcommand{\noist}{%
    Update attributes list
5148 \@gls@addpredefinedattributes
5149 \let\writeist\relax
5150 }

```

`\@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. You either need to have a `\makeglossary` 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`.

`\@makeglossary`

```

5151 \newcommand*{\@makeglossary}[1]{%
5152 \ifglossaryexists{#1}%
5153 {%

```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```

5154 \ifglssavewrites
5155 \expandafter\newtoks\csname glo@#1@filetok\endcsname
5156 \else
5157 \expandafter\newwrite\csname glo@#1@file\endcsname

```

```

5158      \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
5159      \fi
5160      \@gls@renewglossary
5161      \writeist
5162  }%
5163  {%
5164      \PackageError{glossaries}%
5165      {Glossary type ‘#1’ not defined}%
5166      {New glossaries must be defined before using \string\makeglossaries}%
5167  }%
5168 }

```

`\@glsopenfile` Open write file associated with the given glossary.

```

5169 \newcommand*{\@glsopenfile}[2]{%
5170   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
5171   \PackageInfo{glossaries}{Writing glossary file
5172     \jobname.\csname @glotype@#2@out\endcsname}%
5173 }

```

`\@closegls`

```

5174 \newcommand*{\@closegls}[1]{%
5175   \closeout\csname glo@#1@file\endcsname
5176 }

```

`\@gls@automake`

```

5177 \ifglxindy
5178 \newcommand*{\@gls@automake}[1]{%
5179   \ifglossaryexists{#1}
5180   {%
5181     \@closegls{#1}%
5182     \ifdefstring{\glsorder}{letter}%
5183     {\def\@gls@order{-M ord/letorder }}%
5184     {\let\@gls@order\@empty}%
5185     \ifcsundef{@xdy@#1@language}%
5186     {\let\@gls@langmod\@xdy@main@language}%
5187     {\letcs\@gls@langmod{@xdy@#1@language}}%
5188     \edef\@gls@dothiswrite{\noexpand\write18{xindy
5189       -I xindy
5190       \@gls@order
5191       -L \@gls@langmod\space
5192       -M \@gls@istfilebase\space
5193       -C \@gls@codepage\space
5194       -t \jobname.\csuse{@glotype@#1@log}
5195       -o \jobname.\csuse{@glotype@#1@in}
5196       \jobname.\csuse{@glotype@#1@out}}}%
5197     }%
5198     \@gls@dothiswrite
5199   }%
5200   {%

```

```

5201 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5202 }%
5203 }
5204 \else
5205 \newcommand*{\@gls@automake}[1]{%
5206 \ifglossaryexists{#1}
5207 {%
5208 \closegls{#1}%
5209 \ifdefstring{\glsorder}{letter}%
5210 {\def\@gls@order{-l }}%
5211 {\let\@gls@order\@empty}%
5212 \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5213 -s \istfilename\space
5214 -t \jobname.\csuse{@glotype@#1@log}
5215 -o \jobname.\csuse{@glotype@#1@in}
5216 \jobname.\csuse{@glotype@#1@out}}}%
5217 }%
5218 \@gls@dothiswrite
5219 }%
5220 {%
5221 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5222 }%
5223 }
5224 \fi

```

omake@immediate

```

5225 \ifglxsindy
5226 \newcommand*{\@gls@automake@immediate}[1]{%
5227 \ifglossaryexists{#1}
5228 {%
5229 \IfFileExists{\jobname.\csuse{@glotype@#1@out}}{%
5230 {%
5231 \ifdefstring{\glsorder}{letter}%
5232 {\def\@gls@order{-M ord/letorder }}%
5233 {\let\@gls@order\@empty}%
5234 \ifcsundef{@xdy@#1@language}%
5235 {\let\@gls@langmod\@xdy@main@language}%
5236 {\letcs\@gls@langmod{@xdy@#1@language}}%
5237 \edef\@gls@dothiswrite{\noexpand\immediate\noexpand\write18{xindy
5238 -I xindy
5239 \@gls@order
5240 -L \@gls@langmod\space
5241 -M \gls@istfilebase\space
5242 -C \gls@codepage\space
5243 -t \jobname.\csuse{@glotype@#1@log}
5244 -o \jobname.\csuse{@glotype@#1@in}
5245 \jobname.\csuse{@glotype@#1@out}}}%
5246 }%
5247 \@gls@dothiswrite

```

```

5248 }%
5249 {\GlossariesWarning{can't automake '#1': \jobname.\csuse{@glotype@#1@out}
5250 doesn't exist. Rerun may be required}}%
5251 }%
5252 {%
5253 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5254 }%
5255 }
5256 \else
5257 \newcommand*{\@gls@automake@immediate}[1]{%
5258 \ifglossaryexists{#1}
5259 {%
5260 \IfFileExists{\jobname.\csuse{@glotype@#1@out}}{%
5261 {%
5262 \ifdefstring{\glsorder}{letter}%
5263 {\def\@gls@order{-l }}%
5264 {\let\@gls@order\@empty}%
5265 \edef\@gls@dothiswrite{\noexpand\immediate\noexpand\write18{makeindex \@gls@order
5266 -s \istfilename\space
5267 -t \jobname.\csuse{@glotype@#1@log}
5268 -o \jobname.\csuse{@glotype@#1@in}
5269 \jobname.\csuse{@glotype@#1@out}}}%
5270 }%
5271 \@gls@dothiswrite
5272 }%
5273 {\GlossariesWarning{can't automake '#1': \jobname.\csuse{@glotype@#1@out}
5274 doesn't exist. Rerun may be required}}%
5275 }%
5276 {%
5277 \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5278 }%
5279 }
5280 \fi

```

omakeglossaries Issue warning that \makeglossaries hasn't been used.

```
5281 \newcommand*{\@warn@nomakeglossaries}{}%
```

Only use this if warning if \printglossary has been used without \makeglossaries

```
5282 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}%

```

omake@immediate

```

5283 \newcommand{\@gls@@automake@immediate}{%
5284 \ifnum\gls@automake@nr=2\relax
5285 \@for\@gls@type:=\@glo@types\do{%
5286 \ifdefempty{\@gls@type}{}%
5287 {\@gls@automake@immediate{\@gls@type}}%
5288 }%
5289 \glsautomakefalse
5290 \renewcommand*{\@gls@doautomake}{}%
5291 \fi

```


5292 }

\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

5293 \newcommand*{\makeglossaries}{%

If automake=immediate setting is on, use the shell escape now.

5294 \@gls@@automake@immediate

Define the write used for style file also used for all other output files if savewrites=true.

5295 \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.

5296 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}}

5297 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}}

If \@gls@extramakeindexopts has been defined, write it:

5298 \ifundef\@gls@extramakeindexopts

5299 {}%

5300 {%

5301 \protected@write\@auxout{}{\string\providecommand

5302 \string\@gls@extramakeindexopts[1]{}}

5303 \protected@write\@auxout{}{\string\@gls@extramakeindexopts

5304 {\@gls@extramakeindexopts}}%

5305 }%

Write the name of the style file to the aux file (needed by makeglossaries)

5306 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%

5307 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

Iterate through each glossary type and activate it.

5308 \@for\@glo@type:=\@glo@types\do{%

5309 \ifthenelse{\equal{\@glo@type}{}}{}}{%

5310 \@makeglossary{\@glo@type}}%

5311 }%

New glossaries must be created before \makeglossaries so disable \newglossary.

5312 \renewcommand*\newglossary[4][]{%

5313 \PackageError{glossaries}{New glossaries

5314 must be created before \string\makeglossaries}{You need

5315 to move \string\makeglossaries\space after all your

5316 \string\newglossary\space commands}}%

Any subsequence 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).

5317 \let\@makeglossary\relax

5318 \let\makeglossaries\relax

Disable all commands that have no effect after \makeglossaries

```

5319 \let\@disable@onlypremakeg

```

Allow see key:

```

5320 \let\gls@checkseeallowed\relax

```

Suppress warning about no \makeglossaries

```

5321 \let\warn@nomakeglossaries\relax

```

Activate warning about missing \printglossary

```

5322 \def\warn@noprintglossary{%
5323   \ifdefstring{\@glo@types}{,}%
5324   {%
5325     \GlossariesWarningNoLine{No glossaries have been defined}%
5326   }%
5327   {%
5328     \GlossariesWarningNoLine{No \string\printglossary\space
5329       or \string\printglossaries\space
5330       found. ^^J(Remove \string\makeglossaries\space if you
5331       don't want any glossaries.) ^^JThis document will not
5332       have a glossary}%
5333   }%
5334 }%

```

Declare list parser for \glsdisplaynumberlist

```

5335 \ifglssavenumberlist
5336   \edef\@gls@ddeflistparser{\noexpand\DeclareListParser
5337     {\noexpand\glsnumlistparser}{\delimN}}%
5338   \@gls@ddeflistparser
5339 \fi

```

Prevent user from also using \makenoidxglossaries

```

5340 \let\makenoidxglossaries\@no@makeglossaries

```

Prohibit sort key in printgloss family:

```

5341 \renewcommand*{\@printgloss@setsort}{%
5342   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5343 }%

```

Check the automake setting:

```

5344 \ifglssautomake
5345   \renewcommand*{\@gls@doautomake}{%
5346     \@for\@gls@type:=\@glo@types\do{%
5347       \ifdefempty{\@gls@type}{}%
5348       {\@gls@automake{\@gls@type}}%
5349     }%
5350   }%
5351 \fi

```

Check the sort setting:

```

5352 \@glo@check@sortallowed\makeglossaries
5353 }

```

Must occur in the preamble:

```
5354 \@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.

```
5355 \AtEndDocument{%
5356   \warn@nomakeglossaries
5357   \warn@noprintglossary
5358 }
```

`noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
5359 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5360   \renewcommand{\@gls@noref@warn}[1]{%
5361     \GlossariesWarning{Empty glossary for
5362       \string\printnoidxglossary[type={##1}].
5363     Rerun may be required (or you may have forgotten to use
5364     commands like \string\gls)}}%
5365 }
```

Don't escape makeindex/xindy characters:

```
5366 \let\@gls@checkmkidxchars\@gobble
```

Don't escape locations:

```
5367 \glscsclocationsfalse
```

Write glossary information to aux instead of glossary files

```
5368 \let\@do@@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5369 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5370 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5371 \renewcommand{\@do@seeglossary}[2]{%
5372   \edef\@gls@label{\glsdetoklabel{##1}}%
5373   \protected@write\@auxout{}{%
5374     \string\@gls@reference
5375     {\csname glo@\@gls@label @type\endcsname}%
5376     {\@gls@label}%
5377     {%
5378       \string\glsseeformat##2}%
5379     }%
5380   }%
5381 }
```

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.

```

5382 \AtBeginDocument
5383 {%
5384   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5385 }%

  Change warning about no glossaries
5386 \def\warn@noprintglossary{%
5387   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5388     or \string\printnoidxglossaries ^^J
5389     found. (Remove \string\makenoidxglossaries\space if you
5390     don't want any glossaries.)^^JThis document will not have a glossary}%
5391 }%

  Suppress warning about no \makeglossaries
5392 \let\warn@nomakeglossaries\relax

  Prevent user from also using \makeglossaries
5393 \let\makeglossaries\@no@makeglossaries

  Allow sort key in printgloss family:
5394 \renewcommand*{\@printgloss@setsort}{%
5395   \let\@glo@assign@sortkey\@glo@assign@sortkey

  Initialise default sort order:
5396   \def\@glo@sorttype{\@glo@default@sorttype}%
5397 }%

  All entries must be defined in the preamble:
5398 \renewcommand*\new@glossaryentry[2]{%
5399   \PackageError{glossaries}{Glossary entries must be
5400     defined in the preamble^^Jwhen you use
5401     \string\makenoidxglossaries}%
5402   {Either move your definitions to the preamble or use
5403     \string\makeglossaries}%
5404 }%

  Redefine \glsentrynumberlist
5405 \renewcommand*\glsentrynumberlist[1]{%
5406   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5407   \ifdef\@gls@loclist
5408   {%
5409     \glsnoidxloclist{\@gls@loclist}%
5410   }%
5411   {%
5412     ??\glsdoifexists{##1}%
5413     {%
5414       \GlossariesWarning{Missing location list for '##1'. Either
5415         a rerun is required or you haven't referenced the entry}%
5416     }%
5417   }%

```

```

5418 }%
    Redefine \glsdisplaynumberlist
5419 \renewcommand*{\glsdisplaynumberlist}[1]{%
5420   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5421   \ifdef\@gls@loclist
5422   {%
5423     \def\@gls@noidxloclist@sep{%
5424       \def\@gls@noidxloclist@sep{%
5425         \def\@gls@noidxloclist@sep{%
5426           \glsnumlistsep
5427         }%
5428         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5429       }%
5430     }%
5431     \def\@gls@noidxloclist@finalsep{}%
5432     \def\@gls@noidxloclist@prev{}%
5433     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5434     \@gls@noidxloclist@finalsep
5435     \@gls@noidxloclist@prev
5436   }%
5437   {%
5438     ??\glsdoifexists{##1}%
5439     {%
5440       \GlossariesWarning{Missing location list for ‘##1’. Either
5441         a rerun is required or you haven’t referenced the entry}%
5442     }%
5443   }%
5444 }%

```

Provide a generic way of iterating through the number list:

```

5445 \renewcommand*{\glsnumberlistloop}[3]{%
5446   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5447   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5448   \let\@gls@org@glsseeformat\glsseeformat
5449   \let\glsnoidxdisplayloc##2\relax
5450   \let\glsseeformat##3\relax
5451   \ifdef\@gls@loclist
5452   {%
5453     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5454   }%
5455   {%
5456     ??\glsdoifexists{##1}%
5457     {%
5458       \GlossariesWarning{Missing location list for ‘##1’. Either
5459         a rerun is required or you haven’t referenced the entry}%
5460     }%
5461   }%
5462   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5463   \let\glsseeformat\@gls@org@glsseeformat

```

```

5464 }%

    Modify sanitize sort function
5465 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5466 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5467 \@gls@noidx@setsanitizesort

    Check sort option allowed.
5468 \@gls@check@sortallowed\makenoidxglossaries
5469 }

    Preamble-only command:
5470 \@onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

5471 \newcommand*{\glsnumberlistloop}[2]{%
5472   \PackageError{glossaries}{\string\glsnumberlistloop\space
5473     only works with \string\makenoidxglossaries}{}%
5474 }

```

`\listloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc {<prefix>}{<counter>}{<format>}{<n>}`)

```

5475 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5476   #1%
5477 }

```

`\makeglossaries` Can't use both `\makeglossaries` and `\makenoidxglossaries`

```

5478 \newcommand*{\@no@makeglossaries}{%
5479   \PackageError{glossaries}{You can't use both
5480     \string\makeglossaries\space and \string\makenoidxglossaries}%
5481     {Either use one or other (or none) of those commands but not both
5482     together.}%
5483 }

```

`\@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```

5484 \newcommand{\@gls@noref@warn}[1]{%
5485   \GlossariesWarning{\string\makenoidxglossaries\space
5486     is required to make \string\printnoidxglossary[type={#1}] work}%
5487 }

```

1.14 Writing information to associated files

`\s@noidxglossary` Write the glossary information to the aux file (for the ‘noidx’ method):

```

5488 \newcommand*{\@gls@noidxglossary}{%
5489   \protected@write\@auxout{}{%
5490     \string\@gls@reference

```

```

5491     {\csname glo@\@gls@label @type\endcsname}%
5492     {\@gls@label}%
5493     {\string\glsnoidxdisplayloc
5494     {\@glo@counterprefix}%
5495     {\@gls@counter}%
5496     {\@glsnumberformat}%
5497     {\@glslocref}%
5498     }%
5499 }%
5500 }

```

`\istfile` Deprecated.

```

5501 \providecommand\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

5502 \AtEndDocument{%
5503   \glswritefiles
5504 }

```

`\@glswritefiles` Only write the files if `savewrites=true`.

```

5505 \newcommand*{\@glswritefiles}{%

```

Iterate through all the glossaries.

```

5506   \forallglossaries{\@glo@type}{%

```

Check for empty glossaries (patch provided by Patrick Häcker)

```

5507     \ifcsundef{glo@\@glo@type @filetok}%
5508     {%
5509       \def\gls@tmp{}%
5510     }%
5511     {%
5512       \edef\gls@tmp{\expandafter\the
5513         \csname glo@\@glo@type @filetok\endcsname}%
5514     }%
5515     \ifx\gls@tmp\@empty
5516       \ifx\@glo@type\glsdefaulttype
5517         \GlossariesWarningNoLine{Glossary '@@glo@type' has no
5518           entries.^^JRemember to use package option 'nomain' if
5519 you
5520           don't want to^^Juse the main glossary}%
5521       \else
5522         \GlossariesWarningNoLine{Glossary '@@glo@type' has no
5523           entries}%
5524       \fi
5525     \else
5526       \@glsopenfile{\glswrite}{\@glo@type}%
5527       \immediate\write\glswrite{%
5528         \expandafter\the
5529         \csname glo@\@glo@type @filetok\endcsname}%
5530       \immediate\closeout\glswrite

```

```

5531     \fi
5532 }%
5533 }

```

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 `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```

5534 \newcommand*{\gls@glossary}[1]{%
5535   \@gls@glossary{#1}%
5536 }

```

`\@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 `\glsperscentchar` would be needed with the sort sanitization switched off.) This means that it's safe to simply ignore the second argument.

```

5537 \newcommand*{\@gls@glossary}[2]{%
5538   \if@gls@debug
5539     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5540   \fi
5541 }

```

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.

`s@renewglossary`


```

5542 \newcommand{\@gls@renewglossary}{%
5543   \gdef\@gls@glossary##1{\@bsphack\beginingroup\gls@wrglossary{##1}}%
5544   \let\@gls@renewglossary\empty
5545 }

```

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`

```

5546 \newcommand*{\gls@wrglossary}[2]{%
5547   \ifglssavewrites
5548     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5549     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5550     \expandafter{\@gls@tmp~J}%
5551   \else
5552     \ifcsdef{glo@#1@file}%
5553     {%
5554       \expandafter\protected@write\csname glo@#1@file\endcsname{%
5555         \gls@disablepagerefexpansion}{#2}%
5556     }%
5557     {%
5558       \ifignoredglossary{#1}{}%
5559       {%
5560         \GlossariesWarning{No file defined for glossary ‘#1’}%
5561       }%
5562     }%
5563   \fi
5564   \endgroup\@esphack
5565 }

```

`\@do@wrglossary`

```

5566 \newcommand*{\@do@wrglossary}[1]{%
5567   \glswriteentry{#1}{\@do@wrglossary{#1}}%
5568 }

```

`\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.

```

5569 \newcommand*{\glswriteentry}[2]{%
5570   \ifglindexonlyfirst
5571     \ifglused{#1}{}{#2}%
5572   \else
5573     #2%
5574   \fi
5575 }

```

`tected@pagefmts` List of page formats to be protected against expansion.

```

5576 \newcommand{\gls@protected@pagefmts}{\gls@numberpage,\gls@alphpage,%
5577 \gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage}

```

agerefexpansion

```

5578 \newcommand*{\gls@disablepagerefexpansion}{%
5579   \@for\@gls@this:=\gls@protected@pagefmts\do
5580   {%
5581     \expandafter\let\@gls@this\relax
5582   }%
5583 }

```

\gls@alphpage

```

5584 \newcommand*{\gls@alphpage}{\@alph\c@page}

```

\gls@Alphpage

```

5585 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

```

\gls@numberpage

```

5586 \newcommand*{\gls@numberpage}{\number\c@page}

```

\gls@arabicpage

```

5587 \newcommand*{\gls@arabicpage}{\@arabic\c@page}

```

\gls@romanpage

```

5588 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

```

\gls@Romanpage

```

5589 \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 ($\langle csname \rangle \c@page$ must be valid).

```

5590 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5591   \eappto\gls@protected@pagefmts{\,\expandonce{\csname gls#1page\endcsname}}%
5592   \csedef{gls#1page}{\,\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5593   \eappto\@wrglossarynumberhook{%
5594     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5595     \expandonce{\csname#1\endcsname}%
5596     \noexpand\def\expandonce{\csname#1\endcsname}{%
5597       \noexpand\@wrglossary@pageformat
5598       \expandonce{\csname gls#1page\endcsname}%
5599       \expandonce{\csname org@gls#1\endcsname}%
5600     }%
5601   }%
5602 }

```

ssarynumberhook Hook used by \@@do@wrglossary
5603 \newcommand*\@wrglossarynumberhook{}

sary@pageformat
5604 \newcommand{\@wrglossary@pageformat}[3]{%
5605 \ifx#3\c@page #1\else #2#3\fi
5606 }

@@do@wrglossary Write the glossary entry in the appropriate format.
5607 \newcommand*\@@do@wrglossary[1]{%
5608 \ifglsecllocations
5609 \@@do@esc@wrglossary{#1}%
5610 \else
5611 \@@do@noesc@wrglossary{#1}%
5612 \fi
5613 }

noesc@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.

5614 \newcommand*\@@do@noesc@wrglossary[1]{%

Don't fully expand yet.

5615 \expandafter\def\expandafter\@glsloc\ref\expandafter{\the\glsentrycounter}%
5616 \expandafter\def\expandafter\@glsHloc\ref\expandafter{\the\Hglentrycounter}%

Find the prefix if \@glsHloc\ref and \@glsloc\ref aren't the same.

5617 \ifx\@glsHloc\ref\@glsloc\ref
5618 \def\@gls@counterprefix{}%
5619 \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 \the\glsentrycounter and \the\Hglentrycounter at this point, even if it hasn't been updated yet. The page number is not expected to occur in the prefix.)

5620 \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5621 {\@glsloc\ref}{\@glsHloc\ref}}%
5622 }%
5623 \@do@gls@getcounterprefix
5624 \fi

De-tok label if required.

5625 \edef\@gls@label{\@glsdetoklabel{#1}}%

Write the information to file:

5626 \@@do@wrglossary
5627 }

owprimitivemods Conditional to determine whether or not \@@do@esc@wrglossary should be allowed to temporarily redefine \the and \number.

```
5628 \newif\ifglswrallowprimitivemods
```

```
5629 \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).

```
5630 \newcommand*{\@@do@esc@wrglossary}[1]{% please read documented code!
5631 \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions (scoped):

```
5632 \let\gls@orgthe\the
5633 \let\gls@orgnumber\number
5634 \let\gls@orgarabic\@arabic
5635 \let\gls@orgromannumeral\romannumeral
5636 \let\gls@orgalph\@alph
5637 \let\gls@orgAlph\@Alph
5638 \let\gls@orgRoman\@Roman
```

Redefine:

```
5639 \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.

```
5640 \def\gls@the##1{%
5641 \ifx##1\c@page \gls@numberpage\else\gls@orgthe##1\fi}%
5642 \def\the{\expandafter\gls@the}%
5643 \def\gls@number##1{%
5644 \ifx##1\c@page \gls@numberpage\else\gls@orgnumber##1\fi}%
5645 \def\number{\expandafter\gls@number}%
5646 \fi
5647 \def\@arabic##1{%
5648 \ifx##1\c@page \gls@arabicpage\else\gls@orgarabic##1\fi}%
5649 \def\romannumeral##1{%
5650 \ifx##1\c@page \gls@romanpage\else\gls@orgromannumeral##1\fi}%
5651 \def\@Roman##1{%
5652 \ifx##1\c@page \gls@Romanpage\else\gls@orgRoman##1\fi}%
5653 \def\@alph##1{%
5654 \ifx##1\c@page \gls@alphpage\else\gls@orgalph##1\fi}%
5655 \def\@Alph##1{%
5656 \ifx##1\c@page \gls@Alphpage\else\gls@orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5657 \@wrglossarynumberhook
```

Prevent expansion:

```
5658 \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```

5659 \protected@xdef\@glslocref{\theglsentrycounter}%
5660 \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.

```

5661 \@gls@checkmkidxchars\@glslocref

```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

5662 \expandafter\ifx\theglsentrycounter\theglsentrycounter\relax
5663 \def\@glo@counterprefix{}%
5664 \else
5665 \protected@edef\@glsHlocref{\theglsentrycounter}%
5666 \@gls@checkmkidxchars\@glsHlocref
5667 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5668 {\@glslocref}{\@glsHlocref}%
5669 }%
5670 \@do@gls@getcounterprefix
5671 \fi

```

De-tok label if required

```

5672 \edef\@gls@label{\glsdetoklabel{#1}}%

```

Write the information to file:

```

5673 \@do@wrglossary
5674 }

```

@do@wrglossary

```

5675 \newcommand*{\@do@wrglossary}{%

```

Determine whether to use `xindy` or `makeindex` syntax

```

5676 \ifglsxindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5677 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5678 \def\@glo@range{}%
5679 \expandafter\if\@glo@prefix(\relax
5680 \def\@glo@range{:open-range}%
5681 \else
5682 \expandafter\if\@glo@prefix)\relax
5683 \def\@glo@range{:close-range}%
5684 \fi
5685 \fi

```

Write to the glossary file using `xindy` syntax.

```

5686 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5687 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)

5688 :locref \string"\@glo@counterprefix}{\@glslocref}\string" %
5689 :attr \string"\@gls@counter\@glo@suffix\string"
5690 \@glo@range
5691 )
5692 }%
5693 \else

```

Convert the format information into the format required for makeindex

```
5694 \set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5695 {\@glo@counterprefix}%
```

Write to the glossary file using makeindex syntax.

```
5696 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5697 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5698 \@gls@encapchar\@glo@numfmt}{\@gls@locref}}%
5699 \fi
5700 }
```

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.)

```
5701 \newcommand*\@gls@getcounterprefix[2]{%
5702 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5703 \ifx\@gls@thisloc\@gls@thisHloc
5704 \def\@glo@counterprefix{}%
5705 \else
5706 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5707 \def\@glo@tmp{##2}%
5708 \ifx\@glo@tmp\@empty
5709 \def\@glo@counterprefix{}%
5710 \else
5711 \def\@glo@counterprefix{##1}%
5712 \fi
5713 }%
5714 \@gls@get@counterprefix#2.#1\end@getprefix
```

Warn if no prefix can be formed.

```
5715 \ifx\@glo@counterprefix\@empty
5716 \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5717 prefixing^^Jlocation ‘#1’. You need to modify the
5718 definition of \string\theH\@gls@counter^^Jotherwise you
5719 will get the warning: “name{\@gls@counter.#1}’ has been^^J
5720 referenced but does not exist”}%
5721 \fi
5722 \fi
5723 }
```

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.

```
5724 \newcommand{\@do@seeglossary}[2]{%
```

```

5725 \def\@gls@xref{#2}%
5726 \@onelevel@sanitize\@gls@xref
5727 \@gls@checkmkidxchars\@gls@xref
5728 \ifglxsindy
5729   \gls@glossary{\csname glo@#1@type\endcsname}{%
5730     (indexentry
5731       :tkey (\csname glo@#1@index\endcsname)
5732       :xref (\string"\@gls@xref\string")
5733       :attr \string"see\string"
5734     )
5735   }%
5736 \else
5737   \gls@glossary{\csname glo@#1@type\endcsname}{%
5738     \string\glossaryentry{\csname glo@#1@index\endcsname
5739     \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5740 \fi
5741 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5742 \def\@gls@fixbraces#1#2#3\@nil{%
5743   \ifx#2[\relax
5744     \@gls@fixbraces#1#2#3\@end@fixbraces
5745   \else
5746     \def#1{{#2#3}}%
5747   \fi
5748 }

```

`@@gls@fixbraces`

```

5749 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5750   \def#1{[#2]{#3}}%
5751 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

5752 \DeclareRobustCommand*\glssee[3][\seename]{%
5753   \@do@seeglossary{#2}{#1}{#3}}
5754 \newcommand*\@glssee[3][\seename]{%
5755   \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.

```

5756 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5757   \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

5758 \DeclareRobustCommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

5759   \let\@gls@dolast\relax

```


Don't display separator on the first iteration of the loop

```
5760 \let\@gls@donext\relax
```

Iterate through the labels

```
5761 \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
5762 \ifx\@xfor@nextelement\@nnil
5763 \@gls@dolast
5764 \else
5765 \@gls@donext
5766 \fi
```

Display the entry for this label. (Expanding label as it's a temporary control sequence that's used elsewhere.)

```
5767 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5768 \let\@gls@dolast\glsseelastsep
5769 \let\@gls@donext\glsseesep
5770 }%
5771 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5772 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
5773 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5774 \DeclareRobustCommand*\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.)

```
5775 \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.

```
5776 \newcommand*\gls@save@numberlist[1]{%
5777 \ifglssavenumberlist
5778 \toks@{#1}%
5779 \edef\@do@writeaux@info{%
```

```

5780      \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5781    }%
5782    \@onelevel@sanitize\@do@writeaux@info
5783    \protected@write\@auxout{}{\@do@writeaux@info}%
5784  \fi
5785 }

```

`\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.)

```

5786 \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.

```

5787 \ifcsundef{printglossary}{}%
5788 {%

```

If `\printglossary` is already defined, issue a warning and undefine it.

```

5789   \@gls@warnonglossdefined
5790   \undef\printglossary
5791 }

```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```

5792 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5793   \@printglossary{#1}{\@print@glossary}%
5794 }

```

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`

```

5795 \newcommand*{\printglossaries}{}%
5796   \forall@glossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5797 }

```

`\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.

```

5798 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5799   \@printglossary{#1}{\@print@noidx@glossary}%
5800 }

```

noidxglossaries Analogous to \printglossaries

```

5801 \newcommand*\printnoidxglossaries{%
5802   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5803 }

```

ntgloss@setsort Initialise to do nothing.

```

5804 \newcommand*\@printgloss@setsort{}

```

preglossaryhook

```

5805 \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.

```

5806 \newcommand{\@printglossary}[2]{%
  Set up defaults.
5807   \def\@glo@type{\glsdefaulttype}%
5808   \def\glossarytitle{\csname @glo@type@\@glo@type @title\endcsname}%
5809   \def\glossarytoctitle{\glossarytitle}%
5810   \let\org@glossarytitle\glossarytitle
5811   \def\@glossarystyle{%
5812     \ifx\@glossary@default@style\relax
5813       \GlossariesWarning{No default glossary style provided \MessageBreak
5814         for the glossary '\@glo@type'. \MessageBreak
5815         Using deprecated fallback. \MessageBreak
5816         To fix this set the style with \MessageBreak
5817         \string\setglossarystyle\space or use the \MessageBreak
5818         style key=value option}%
5819     \fi
5820   }%
5821   \def\gls@dotoc@title{\glssettoc@title{\@glo@type}}%
  Store current value of \glossaryentrynumbers. (This may be changed via the optional argument)
5822   \let\@org@glossaryentrynumbers\glossaryentrynumbers
  Localise the effects of the optional argument
5823   \bgroup
  Activate or deactivate sort key:
5824   \@printgloss@setsort
  Determine settings specified in the optional argument.
5825   \setkeys{printgloss}{#1}%
  Does the glossary exist?
5826   \ifglossaryexists{\@glo@type}%
5827   {%

```

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)

```
5828 \ifx\glossarytitle\org@glossarytitle
5829 \else
5830 \expandafter\let\csname @glo@type @title\endcsname
5831 \glossarytitle
5832 \fi
```

Allow a high-level user command to indicate the current glossary

```
5833 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5834 \let\org@glossaryentrynumbers\glossaryentrynumbers
5835 \let\glsnonextpages\glsnonextpages
```

Enable individual number list to be activated:

```
5836 \let\glsnextpages\glsnextpages
```

Enable suppression of description terminators.

```
5837 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5838 \gls@dotocitle
```

Set the glossary style

```
5839 \@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):

```
5840 \let\gls@org@glossaryentryfield\glossentry
5841 \let\gls@org@glossarysubentryfield\subglossentry
5842 \renewcommand{\glossentry}[1]{%
5843 \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5844 \gls@org@glossaryentryfield{##1}%
5845 }%
5846 \renewcommand{\subglossentry}[2]{%
5847 \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5848 \gls@org@glossarysubentryfield{##1}{##2}%
5849 }%
```

```
5850 \@gls@preglossaryhook
```

Now do the handler macro that deals with the actual glossary:

```
5851 #2%
5852 }%
5853 {\GlossariesWarning{Glossary '@glo@type' doesn't exist}}%
```

End the current scope

```
5854 \egroup
```

Reset \glossaryentrynumbers

```
5855 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no \printglossary

```
5856 \global\let\warn@noprntglossary\relax
5857 }
```

@print@glossary Internal workings of \printglossary dealing with reading the external file.

```
5858 \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.)

```
5859 \makeatletter
```

Input the glossary file, if it exists.

```
5860 \@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.

```
5861 \IfFileExists{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
5862 {}%
5863 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5864 \ifglxindy
5865 \ifcsundef{@xdy@\@glo@type @language}%
5866 {%
5867 \edef\@do@auxoutstuff{%
5868 \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.

```
5869 \noexpand\immediate\noexpand\write\@auxout{%
5870 \string\providecommand\string\@xdylanguage[2]{}}%
5871 \noexpand\immediate\noexpand\write\@auxout{%
5872 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5873 }%
5874 }%
5875 }%
5876 {%
5877 \edef\@do@auxoutstuff{%
5878 \noexpand\AtEndDocument{%
5879 \noexpand\immediate\noexpand\write\@auxout{%
5880 \string\providecommand\string\@xdylanguage[2]{}}%
5881 \noexpand\immediate\noexpand\write\@auxout{%
5882 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5883 @language\endcsname}}%
5884 }%
5885 }%
5886 }%
5887 \@do@auxoutstuff
```

```

5888 \edef\@do@auxoutstuff{%
5889 \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.

```

5890 \noexpand\immediate\noexpand\write\@auxout{%
5891 \string\providecommand\string\@gls@codepage[2]{}}%
5892 \noexpand\immediate\noexpand\write\@auxout{%
5893 \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5894 }%
5895 }%
5896 \@do@auxoutstuff
5897 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

5898 \renewcommand*{\@warn@nomakeglossaries}{%
5899 \GlossariesWarningNoLine{\string\makeglossaries\space
5900 hasn't been used,^^Jthe glossaries will not be updated}%
5901 }%
5902 }

```

The sort macros all have the syntax:

$$\backslash@glo@sortmacro@<order>\{<type>\}$$

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 $\backslash@gls@sref@<type>$. The actual sorting is done by $\backslash@glo@sortentries\{<handler>\}\{<type>\}$).

$\backslash@glo@sortentries$

```

5903 \newcommand*{\@glo@sortentries}[2]{%
5904 \glosortentrieswarning
5905 \def\@glo@sortinglist{}%
5906 \def\@glo@sortinghandler{#1}%
5907 \edef\@glo@type{#2}%
5908 \forlistcsloop{\@glo@do@sortentries}{\@gls@sref@#2}%
5909 \csdef{\@gls@sref@#2}{}%
5910 \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5911 \xifinlistcs{\@this@label}{\@gls@sref@#2}%
5912 {}%
5913 {%
5914 \listcsxadd{\@gls@sref@#2}{\@this@label}%
5915 }%
5916 \ifcsdef{\@glo@sortingchildren@\@this@label}%
5917 {%
5918 \@glo@addchildren{#2}{\@this@label}%
5919 }%
5920 {}%

```

```

5921 }%
5922 }

```

```
@glo@addchildren \@glo@addchildren{<type>}{<parent>}
```

```
5923 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```

5924 \bgroup
5925 \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
5926 \@for\@this@childlabel:=\@glo@childlist\do
5927 {%

```

Check this label hasn't already been added.

```

5928 \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
5929 }%
5930 {%
5931 \listcsxadd{\@glsref@#1}{\@this@childlabel}%
5932 }%

```

Does this child have children?

```

5933 \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
5934 {%
5935 \@glo@addchildren{#1}{\@this@childlabel}%
5936 }%
5937 {%
5938 }%
5939 }%
5940 \egroup
5941 }

```

```
@do@sortentries
```

```

5942 \newcommand*{\@glo@do@sortentries}[1]{%
5943 \ifglshasparent{#1}%
5944 {%

```

This entry has a parent, so add it to the child list

```

5945 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{#1}@parent}}%
5946 \ifcsundef{\@glo@sortingchildren@\@glo@parent}%
5947 {%
5948 \csdef{\@glo@sortingchildren@\@glo@parent}{}%
5949 }%
5950 }%
5951 \expandafter\@glo@sortedinsert
5952 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```

5953 \xifinlistcs{\@glo@parent}{\@glsref@\@glo@type}%
5954 {%

```

Yes, it has so do nothing.

```
5955 }%
5956 {%
```

No, it hasn't so add it now.

```
5957 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5958 }%
5959 }%
5960 {%
5961 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5962 }%
5963 }
```

`glo@sortedinsert` `\@glo@sortedinsert{<list>}{<entry label>}`

Insert into list.

```
5964 \newcommand*{\@glo@sortedinsert}[2]{%
5965 \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5966 }%
```

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`

```
5967 \newcommand*{\@glo@sorthandler@word}[2]{%
5968 \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5969 \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5970 \edef\glo@do@compare{%
5971 \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5972 {\expandonce\@gls@sort@B}%
5973 {\expandonce\@gls@sort@A}%
5974 }%
5975 \glo@do@compare
5976 }
```

`thandler@letter`

```
5977 \newcommand*{\@glo@sorthandler@letter}[2]{%
5978 \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5979 \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5980 \edef\glo@do@compare{%
5981 \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5982 {\expandonce\@gls@sort@B}%
5983 {\expandonce\@gls@sort@A}%
5984 }%
5985 \glo@do@compare
5986 }
```


sorthandler@case Case-sensitive sort.

```
5987 \newcommand*{\@glo@sorthandler@case}[2]{%
5988   \letcs\@gls@sort@A{\glo\glsdetoklabel{#1}@sort}%
5989   \letcs\@gls@sort@B{\glo\glsdetoklabel{#2}@sort}%
5990   \edef\glo@do@compare{%
5991     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5992     {\expandonce\@gls@sort@B}%
5993     {\expandonce\@gls@sort@A}%
5994   }%
5995   \glo@do@compare
5996 }
```

sorthandler@nocase Case-insensitive sort.

```
5997 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5998   \letcs\@gls@sort@A{\glo\glsdetoklabel{#1}@sort}%
5999   \letcs\@gls@sort@B{\glo\glsdetoklabel{#2}@sort}%
6000   \edef\glo@do@compare{%
6001     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
6002     {\expandonce\@gls@sort@B}%
6003     {\expandonce\@gls@sort@A}%
6004   }%
6005   \glo@do@compare
6006 }
```

@sortmacro@word Sort macro for ‘word’

```
6007 \newcommand*{\@glo@sortmacro@word}[1]{%
6008   \ifdefstring{\@glo@default@sorttype}{standard}%
6009   {%
6010     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
6011   }%
6012   {%
6013     \PackageError{glossaries}{Conflicting sort options:^^J
6014       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6015       \string\printnoidxglossary[sort=word]}{}%
6016   }%
6017 }
```

@sortmacro@letter Sort macro for ‘letter’

```
6018 \newcommand*{\@glo@sortmacro@letter}[1]{%
6019   \ifdefstring{\@glo@default@sorttype}{standard}%
6020   {%
6021     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
6022   }%
6023   {%
6024     \PackageError{glossaries}{Conflicting sort options:^^J
6025       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6026       \string\printnoidxglossary[sort=letter]}{}%
6027   }%
6028 }
```

```

tmacro@standard  Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)
6029 \newcommand*{\@glo@sortmacro@standard}[1]{%
6030   \ifdefstring{\@glo@default@sorttype}{standard}%
6031   {%
6032     \ifcsdef{\@glo@sorthandler@\glsorder}%
6033     {%
6034       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
6035     }%
6036     {%
6037       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
6038     }%
6039   }%
6040   {%
6041     \PackageError{glossaries}{Conflicting sort options:^^J
6042       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6043       \string\printnoidxglossary[sort=standard]}{}%
6044   }%
6045 }

@sortmacro@case  Sort macro for ‘case’
6046 \newcommand*{\@glo@sortmacro@case}[1]{%
6047   \ifdefstring{\@glo@default@sorttype}{standard}%
6048   {%
6049     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
6050   }%
6051   {%
6052     \PackageError{glossaries}{Conflicting sort options:^^J
6053       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6054       \string\printnoidxglossary[sort=case]}{}%
6055   }%
6056 }

ortmacro@nocase  Sort macro for ‘nocase’
6057 \newcommand*{\@glo@sortmacro@nocase}[1]{%
6058   \ifdefstring{\@glo@default@sorttype}{standard}%
6059   {%
6060     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
6061   }%
6062   {%
6063     \PackageError{glossaries}{Conflicting sort options:^^J
6064       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
6065       \string\printnoidxglossary[sort=nocase]}{}%
6066   }%
6067 }

o@sortmacro@def  Sort macro for ‘def’. The order of definition is given in \glolist@<type>.
6068 \newcommand*{\@glo@sortmacro@def}[1]{%
6069   \def\@glo@sortinglist{%
6070     \forglsentries[#1]{\@gls@thislabel}%

```

```

6071  {%
6072    \xifinlistcs{\@gls@thislabel}{\@glsref@#1}%
6073    {%
6074      \listeadadd{\@glo@sortinglist}{\@gls@thislabel}%
6075    }%
6076    {%
      Hasn't been referenced.
6077    }%
6078  }%
6079  \cslet{\@glsref@#1}{\@glo@sortinglist}%
6080 }

```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```

6081 \newcommand*{\@glo@sortmacro@def@do}[1]{%
6082   \ifinlistcs{#1}{\@glsref@{\@glo@type}}%
6083   {}%
6084   {%
6085     \listcsadd{\@glsref@{\@glo@type}}{#1}%
6086   }%
6087   \ifcsdef{\@glo@sortingchildren@#1}%
6088   {%
6089     \@glo@addchildren{\@glo@type}{#1}%
6090   }%
6091   {}%
6092 }

```

o@sortmacro@use Sort macro for 'use'. (No sorting is required, as the entries are already in order of use, so do nothing.)

```

6093 \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.

```

6094 \newcommand*{\@print@noidx@glossary}{%
6095   \ifcsdef{\@glsref@{\@glo@type}}%
6096   {%

```

Sort the entries:

```

6097   \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
6098   {%
6099     \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
6100   }%
6101   {%
6102     \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{}%
6103   }%

```

Do the glossary heading and preamble

```
6104 \glossarysection[\glossarytoctitle]{\glossarytitle}%  
6105 \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.

```
6106 \def\@gls@currentlettergroup{%  
6107 \begin{theglossary}%  
6108 \glossaryheader  
6109 \glsresetentrylist
```

Iterate through the entries.

```
6110 \forlistcsloop{\@gls@noidx@do}{\@glsref@\@glo@type}%
```

Finally end the glossary and do the postamble:

```
6111 \end{theglossary}%  
6112 \glossarypostamble  
6113 }%  
6114 {%  
6115 \@gls@noref@warn{\@glo@type}%  
6116 }%  
6117 }
```

\glo@grabfirst

```
6118 \def\glo@grabfirst#1#2\@nil{%  
6119 \def\@gls@firsttok{#1}%  
6120 \ifdefempty\@gls@firsttok  
6121 {%  
6122 \def\@glo@thislettergrp{0}%  
6123 }%  
6124 {%
```

Sanitize it:

```
6125 \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
6126 \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil  
6127 }%  
6128 }
```

\@glo@grabfirst

```
6129 \def\@glo@grabfirst#1#2\@nil{%  
6130 \ifdefempty\@glo@thislettergrp  
6131 {%  
6132 \def\@glo@thislettergrp{glssymbols}%  
6133 }%  
6134 {%  
6135 \count@=\uccode'#1\relax
```

```

6136 \ifnum\count@=0\relax
6137 \def\@glo@thislettergrp{glssymbols}%
6138 \else
6139 \ifdefstring\@glo@sorttype{case}%
6140 {%
6141 \count@=#1\relax
6142 }%
6143 {%
6144 }%
6145 \edef\@glo@thislettergrp{\the\count@}%
6146 \fi
6147 }%
6148 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label.
This only allows one sublevel.

```

6149 \newcommand{\@gls@noidx@do}[1]{%
  Get this entry's location list
6150 \global\letcs{\@gls@loclist}{glo@glstdetoklabel{#1}@loclist}%
  Does this entry have a parent?
6151 \ifglshasparent{#1}%
6152 {%
  Has a parent.
6153 \gls@level=\csuse{glo@glstdetoklabel{#1}@level}\relax
6154 \ifdefvoid{\@gls@loclist}
6155 {%
6156 \subglossentry{\gls@level}{#1}{}%
6157 }%
6158 {%
6159 \subglossentry{\gls@level}{#1}%
6160 {%
6161 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
6162 }%
6163 }%
6164 }%
6165 {%

```

Doesn't have a parent Get this entry's sort key

```

6166 \letcs{\@gls@sort}{glo@glstdetoklabel{#1}@sort}%
  Fetch the first letter:
6167 \expandafter\glo@grabfirst\@gls@sort{}\@nil
6168 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
6169 {}%
6170 {%

```

Do the group header:

```

6171 \ifdefempty{\@gls@currentlettergroup}{}%
6172 {%

```

The group skip may start a new scope, so make a global assignment.

```

6173      \global\let\@glo@thislettergrp\@glo@thislettergrp
6174      \glsgroupskip
6175      }%
6176      \glsgroupheading{\@glo@thislettergrp}%
6177      }%

6178      \global\let\@gls@currentlettergroup\@glo@thislettergrp

```

Do this entry:

```

6179      \ifdefvoid{\@gls@loclist}
6180      {%
6181          \glossentry{#1}{}%
6182      }%
6183      {%
6184          \glossentry{#1}%
6185          {%
6186              \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
6187          }%
6188      }%
6189      }%
6190 }

```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```

6191 \newcommand*{\glsnoidxloclist}[1]{%
6192   \def\@gls@noidxloclist@sep{}%
6193   \def\@gls@noidxloclist@prev{}%
6194   \forlistloop{\glsnoidxloclisthandler}{#1}%
6195 }

```

`xloclisthandler` Handler for location list iterator.

```

6196 \newcommand*{\glsnoidxloclisthandler}[1]{%
6197   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6198   {%

```

Same as previous location so skip.

```

6199   }%
6200   {%
6201       \@gls@noidxloclist@sep
6202       #1%
6203       \def\@gls@noidxloclist@sep{\delimN}%
6204       \def\@gls@noidxloclist@prev{#1}%
6205   }%
6206 }

```

`yloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```

6207 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
6208   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6209   {%
      Same as previous location so skip.
6210   }%
6211   {%
6212     \@gls@noidxloclist@sep
6213     \@gls@noidxloclist@prev
6214     \def\@gls@noidxloclist@prev{#1}%
6215   }%
6216 }

```

`\glsnoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```

6217 \newcommand*\glsnoidxdisplayloc[4]{%
6218   \setentrycounter[#1]{#2}%
6219   \csuse{#3}{#4}%
6220 }

```

`\@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.

```

6221 \newcommand*{\@gls@reference}[3]{%

```

Add to label list

```

6222   \glsdoifexistsorwarn{#2}%
6223   {%
6224     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
6225     \ifinlistcs{#2}{@glsref@#1}%
6226     {}%
6227     {\listcsgadd{@glsref@#1}{#2}}%

```

Add to location list

```

6228   \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
6229   {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}{}%
6230   {}%
6231   \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
6232 }%
6233 }

```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```

6234 \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`.

```
6235 \define@key{printgloss}{title}{%
6236   \def\glossarytitle{#1}%
6237   \let\gls@dotoc\title\relax
6238 }
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
6239 \define@key{printgloss}{toctitle}{%
6240   \def\glossarytoctitle{#1}%
6241   \let\gls@dotoc\title\relax
6242 }
```

The style key sets the glossary style (but only for the given glossary).

```
6243 \define@key{printgloss}{style}{%
6244   \ifcsundef{@glsstyle@#1}%
6245   {%
6246     \PackageError{glossaries}%
6247     {Glossary style ‘#1’ undefined}{}%
6248   }%
6249   {%
6250     \def\@glossarystyle{\setglossentrycompatibility
6251       \csname @glsstyle@#1\endcsname}%
6252   }%
6253 }
```

The numberedsection key determines if this glossary should be in a numbered section.

```
6254 \define@choicekey{printgloss}{numberedsection}%
6255   [\gls@numberedsection@val\gls@numberedsection@nr]%
6256   {false,nolabel,autolabel,nameref}[nolabel]%
6257   {%
6258     \ifcase\gls@numberedsection@nr\relax
6259       \renewcommand*{\@glossarysecstar}{*}%
6260       \renewcommand*{\@glossaryseclabel}{}%
6261     \or
6262       \renewcommand*{\@glossarysecstar}{}%
6263       \renewcommand*{\@glossaryseclabel}{}%
6264     \or
6265       \renewcommand*{\@glossarysecstar}{}%
6266       \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
6267     \or
6268       \renewcommand*{\@glossarysecstar}{*}%
6269       \renewcommand*{\@glossaryseclabel}{%
6270         \protected@edef\@currentlabelname{\glossarytoctitle}%
6271         \label{\glsautoprefix\@glo@type}}%
6272     \fi
6273 }
```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.


```

6274 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
6275   \csuse{glsnogroupskip#1}%
6276 }

```

The nopostdot key has the same effect as the package option of the same name.

```

6277 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
6278   \csuse{glsnopostdot#1}%
6279 }

```

nterLabelPrefix Make it easier to redefine the label prefix.

```

6280 \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.

```

6281 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
6282   \csuse{glsentrycounter#1}%
6283   \@gls@define@glossaryentrycounter
6284 }

```

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.

```

6285 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6286   \csuse{glssubentrycounter#1}%
6287   \@gls@define@glossarysubentrycounter
6288 }

```

The nonumberlist key determines if this glossary should have a number list.

```

6289 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
6290   \ifglslnonumberlist
6291     \def\glossaryentrynumbers##1{%
6292       \else
6293         \def\glossaryentrynumbers##1{##1}%
6294     \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

6295 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

@assign@sortkey Issue error if used with \printglossary

```

6296 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6297   \PackageError{glossaries}{‘sort’ key not permitted with
6298     \string\printglossary}%
6299   {The ‘sort’ key may only be used with \string\printnoidxglossary}%
6300 }

```

@assign@sortkey For use with \printnoidxglossary

```

6301 \newcommand*{\@glo@assign@sortkey}[1]{%
6302   \def\@glo@sorttype{#1}%
6303 }

```

`\@glsnonetpages` 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 `\glsnonetpages` is place 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.

```

6304 \newcommand*{\@glsnonetpages}{%
6305   \gdef\glossaryentrynumbers##1{%
6306     \glsresetentrylist
6307   }%
6308 }

```

`\@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 place 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.

```

6309 \newcommand*{\@glsnextpages}{%
6310   \gdef\glossaryentrynumbers##1{%
6311     ##1\glsresetentrylist}}

```

`sresetentrylist` Resets `\glossaryentrynumbers`

```

6312 \newcommand*{\glsresetentrylist}{%
6313   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonetpages` Outside of `\printglossary` this does nothing.

```

6314 \newcommand*{\glsnonetpages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

6315 \newcommand*{\glsnextpages}{}

```

Process `entrycounter` and then `subentrycounter` options (this ensures the sub-counter can pick up the main counter as the master if required):

```

6316 \@gls@define@glossaryentrycounter
6317 \@gls@define@glossarysubentrycounter

```

`subentrycounter` Resets the `glossarysubentry` counter.

```

6318 \newcommand*{\glsresetsubentrycounter}{%
6319   \ifglssubentrycounter
6320     \setcounter{glossarysubentry}{0}%
6321   \fi
6322 }

```

subentrycounter Resets the glossaryentry counter.

```
6323 \newcommand*{\glsresetentrycounter}{%
6324   \ifglentrycounter
6325     \setcounter{glossaryentry}{0}%
6326   \fi
6327 }
```

\glsstepentry Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```
6328 \newcommand*{\glsstepentry}[1]{%
6329   \ifglentrycounter
6330     \refstepcounter{glossaryentry}%
6331     \label{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6332   \fi
6333 }
```

glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```
6334 \newcommand*{\glsstepsubentry}[1]{%
6335   \ifglssubentrycounter
6336     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6337     \refstepcounter{glossarysubentry}%
6338     \label{\GlsEntryCounterLabelPrefix\currentglssubentry}%
6339   \fi
6340 }
```

\glsrefentry Reference the entry or sub-entry counter if in use, otherwise just do \gls.

```
6341 \newcommand*{\glsrefentry}[1]{%
6342   \ifglentrycounter
6343     \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6344   \else
6345     \ifglssubentrycounter
6346       \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6347     \else
6348       \gls{#1}%
6349     \fi
6350   \fi
6351 }
```

trycounterlabel Defines how to display the glossaryentry counter.

```
6352 \newcommand*{\glentrycounterlabel}{%
6353   \ifglentrycounter
6354     \theglossaryentry.\space
6355   \fi
6356 }
```

trycounterlabel Defines how to display the glossarysubentry counter.

```
6357 \newcommand*{\glssubentrycounterlabel}{%
```

```

6358 \ifglssubentrycounter
6359   \theglossarysubentry)\space
6360 \fi
6361 }

```

`\glentryitem` Step and display glossaryentry counter, if appropriate.

```

6362 \newcommand*{\glentryitem}[1]{%
6363   \ifglssubentrycounter
6364     \glstepentry{#1}\glentrycounterlabel
6365   \else
6366     \glresetsubentrycounter
6367   \fi
6368 }

```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

6369 \newcommand*{\glssubentryitem}[1]{%
6370   \ifglssubentrycounter
6371     \glstepsubentry{#1}\glssubentrycounterlabel
6372   \fi
6373 }

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

6374 \ifcsundef{theglossary}%
6375 {%
6376   \newenvironment{theglossary}{}{}%
6377 }%
6378 {%
6379   \@gls@warnontheglossdefined
6380   \renewenvironment{theglossary}{}{}%
6381 }

```

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`

```

6382 \newcommand*{\glossaryheader}{}

```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

6383 \newcommand*{\glstarget}[2]{\@glstarget{\glolinkprefix#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}`

```

6384 \providecommand*\compatibleglossentry}[2]{%
6385   \toks@{#2}%
6386   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6387     {\noexpand\glsnamefont
6388       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6389     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6390     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}}%
6391   {\the\toks@}%
6392   }%
6393   \@do@glossentry
6394 }

```

`\glossentryname`

```

6395 \newcommand*\glossentryname}[1]{%
6396   \glsdoifexistsorwarn{#1}%
6397   {%
6398     \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
6399     \expandafter\glsnamefont\expandafter{\glo@name}%
6400   }%
6401 }

```

`\Glossentryname`

```

6402 \newcommand*\Glossentryname}[1]{%
6403   \glsdoifexistsorwarn{#1}%
6404   {%
6405     \glsnamefont{\Glsentryname{#1}}%
6406   }%
6407 }

```

`\glossentrydesc`

```

6408 \newcommand*\glossentrydesc}[1]{%
6409   \glsdoifexistsorwarn{#1}%
6410   {%
6411     \glsentrydesc{#1}%
6412   }%
6413 }

```

`\Glossentrydesc`

```

6414 \newcommand*\Glossentrydesc}[1]{%
6415   \glsdoifexistsorwarn{#1}%

```

```

6416 {%
6417   \Glsentrydesc{#1}%
6418 }%
6419 }

```

lossentrysymbol

```

6420 \newcommand*{\glossentrysymbol}[1]{%
6421   \glsdoifexistsorwarn{#1}%
6422   {%
6423     \glsentrysymbol{#1}%
6424   }%
6425 }

```

lossentrysymbol

```

6426 \newcommand*{\Glossentrysymbol}[1]{%
6427   \glsdoifexistsorwarn{#1}%
6428   {%
6429     \Glsentrysymbol{#1}%
6430   }%
6431 }

```

blesubglossentry

```
\subglossentry{<level>}{<label>}{<page-list>}
```

```

6432 \providecommand*{\compatiblesubglossentry}[3]{%
6433   \toks@{#3}%
6434   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6435     {#2}%
6436     {\noexpand\glsnamefont
6437       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6438     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6439     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6440     {\the\toks@}%
6441   }%
6442   \@do@subglossentry
6443 }

```

rycompatibility

```

6444 \newcommand*{\setglossentrycompatibility}{%
6445   \let\glossentry\compatibleglossentry
6446   \let\subglossentry\compatiblesubglossentry
6447 }
6448 \setglossentrycompatibility

```

ossaryentryfield

```
\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}
```

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```
6449 \newcommand{\glossaryentryfield}[5]{%
6450   \GlossariesWarning
6451   {Deprecated use of \string\glossaryentryfield.^^J
6452    I recommend you change to \string\glossentry.^^J
6453    If you've just upgraded, try removing your gls auxiliary
6454    files^^J and recompile}%
6455   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}
```

arysubentryfield

```
\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.)

```
6456 \newcommand*{\glossarysubentryfield}[6]{%
6457   \GlossariesWarning
6458   {Deprecated use of \string\glossarysubentryfield.^^J
6459    I recommend you change to \string\subglossentry.^^J
6460    If you've just upgraded, try removing your gls auxiliary
6461    files^^J and recompile}%
6462   \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

```
6463 \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

```
6464 \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`

```
6465 \newcommand*{\glsgetgrouptitle}[1]{%
6466   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6467   \@gls@grptitle
6468 }
```

`\@gls@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
6469 \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.

```
6470 \dtl@ifsingle{#1}%
6471 {%
6472   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6473 }%
6474 {%
6475   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6476               or test{\ifstrequal{#1}{glsnumbers}}}%
6477   {%
6478     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6479   }%
6480   {%
6481     \def#2{#1}%
6482   }%
6483 }%
6484 }
```

`\@x@getgrouptitle` Version for the no-indexing app option:


```

6485 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6486   \DTLifint{#1}%
6487   {\edef#2{\char#1\relax}}%
6488   {%
6489     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6490   }%
6491 }

```

`\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`.

`\glsgetgrouplabel`

```

6492 \newcommand*{\glsgetgrouplabel}[1]{%
6493   \ifthenelse{equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6494   \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`

```

6495 \newcommand*{\setentrycounter}[2][ ]{%
6496   \def\@glo@counterprefix{#1}%
6497   \ifx\@glo@counterprefix\empty
6498     \def\@glo@counterprefix{.}%
6499   \else
6500     \def\@glo@counterprefix{.#1.}%
6501   \fi
6502   \def\glsentrycounter{#2}%
6503 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

6504 \newcommand*{\setglossarystyle}[1]{%
6505   \ifcsundef{@glsstyle@#1}%
6506   {%
6507     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6508   }%
6509   {%
6510     \csname @glsstyle@#1\endcsname
6511   }%

```

Set the default style if it's not already set.

```

6512 \ifx\@glossary@default@style\relax
6513   \protected@edef\@glossary@default@style{#1}%

```

```

6514 \fi
6515 }

```

`\glossarystyle`

```

6516 \newcommand*{\glossarystyle}[1]{%
6517   \ifcsundef{@glsstyle@#1}%
6518   {%
6519     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6520   }%
6521   {%
6522     \GlossariesWarning
6523     {Deprecated command \string\glossarystyle.^^J
6524      I recommend you switch to \string\setglossarystyle\space unless
6525      you want to maintain backward compatibility}%
6526     \setglossentrycompatibility
6527     \csname @glsstyle@#1\endcsname
6528   }%
6529   \ifcsdef{@glscompstyle@#1}%
6530   {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6531   {}%
6532 }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6532 \ifx\@glossary@default@style\relax
6533   \protected@edef\@glossary@default@style{#1}%
6534 \fi
6535 }

```

`\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.

```

6536 \newcommand{\newglossarystyle}[2]{%
6537   \ifcsundef{@glsstyle@#1}%
6538   {%
6539     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6540   }%
6541   {%
6542     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6543   }%
6544 }

```

`\newglossarystyle` Code for this macro supplied by Marco Daniel.

```
6545 \newcommand{\renewglossarystyle}[2]{%
6546   \ifcsundef{@glsstyle@#1}%
6547   {%
6548     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6549   }%
6550   {%
6551     \csdef{@glsstyle@#1}{#2}%
6552   }%
6553 }
```

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`

```
6554 \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 `\glshypernumber`. 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`

```
6555 \ifcsundef{hyperlink}%
6556 {%
6557   \def\glshypernumber#1{#1}%
6558 }%
6559 {%
6560   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6561 }
```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6562 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6563   \ifx\#1\%
6564   \else
6565     \@delimR#1\delimR\delimR\%
6566   \fi
```

```

6567 \ifx\|#2\%
6568 \else
6569 #2%
6570 \fi
6571 \ifx\|#3\%
6572 \else
6573 \@glshypernumber#3\@nil
6574 \fi
6575 }

```

`\@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`

```

6576 \def\@delimR#1\delimR #2\delimR #3\{%
6577 \ifx\|#2\%
6578 \@delimN{#1}%
6579 \else
6580 \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6581 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

6582 \def\@delimN#1{\@delimN#1\delimN \delimN\}
6583 \def\@delimN#1\delimN #2\delimN#3\{%
6584 \ifx\|#3\%
6585 \@gls@numberlink{#1}%
6586 \else
6587 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6588 \fi
6589 }

```

The following code is modified from `hyperref's \HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

6590 \def\@gls@numberlink#1{%
6591 \begingroup
6592 \toks@={}%
6593 \@gls@removespaces#1 \@nil
6594 \endgroup}

6595 \def\@gls@removespaces#1 #2\@nil{%
6596 \toks@=\expandafter{\the\toks@#1}%
6597 \ifx\|#2\%
6598 \edef\x{\the\toks@}%
6599 \ifx\x\empty
6600 \else

6601 \hyperlink{\glstrycounter\@gls@counterprefix\the\toks@}%
6602 {\the\toks@}%

```

```

6603 \fi
6604 \else
6605 \@gls@ReturnAfterFi{%
6606 \@gls@removespaces#2\@nil
6607 }%
6608 \fi
6609 }
6610 \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`

```
6611 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}
```

`\hypersf`

```
6612 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}
```

`\hypertt`

```
6613 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}
```

`\hyperbf`

```
6614 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}
```

`\hypermd`

```
6615 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}
```

`\hyperit`

```
6616 \newcommand*\hyperit[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
6617 \newcommand*\hypersl[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6618 \newcommand*\hyperup[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6619 \newcommand*\hypersc[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6620 \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.

```
6621 \newcommand{\oldacronym}[4][\gls@label]{%
6622   \def\gls@label{#2}%
6623   \newacronym[#4]{#1}{#2}{#3}%
6624   \ifcsundef{xspace}%
6625   {%
6626     \expandafter\edef\csname#1\endcsname{%
6627       \noexpand@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6628   }%
6629 }%
6630 {%
6631   \expandafter\edef\csname#1\endcsname{%
6632     \noexpand@ifstar{\noexpand\Gls{#1}\noexpand\xspace}%
6633     \noexpand\gls{#1}\noexpand\xspace}%
6634   }%
6635 }%
6636 }
```

`\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`

```
6637 \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 needed to cancel it out.

```
6638 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6639 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6640 \newcommand*{\glsshortkey}{short}
```

`\sshortpluralkey`

```
6641 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6642 \newcommand*{\glslongkey}{long}
```

`\lslongpluralkey`

```
6643 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6644 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\@ns@acrfull}
```

```
6645 \newcommand*{\ns@acrfull}[2][]{\%
```

```
6646 \new@ifnextchar[{\@acrfull{#1}{#2}}{\%
```

```
6647 {\@acrfull{#1}{#2}[]}%
```

```
6648 }
```

`\@acrfull` Low-level macro:

```
6649 \def\@acrfull#1#2[#3]{\%
```

Make it easier for acronym styles to change this:

```
6650 \acrfullfmt{#1}{#2}{#3}%
```

```
6651 }
```

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.
6652 \newcommand*{\acrfullfmt}[3]{%
6653   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6654 }

\acrlinkfullformat Format for full links like \acrfull. Syntax: \acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}
6655 \newcommand{\acrlinkfullformat}[5]{%
6656   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
6657 }

\acrfullformat Default full form is <long> (<short>).
6658 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}

\glsspace Robust space to ensure it's written to the .glsdefs file.
6659 \newrobustcmd{\glsspace}{\space}

Default format for full acronym

\Acrfull
6660 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}

6661 \newcommand*\ns@Acrfull[2][]{%
6662   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6663     {\@Acrfull{#1}{#2}[]}%
6664 }

Low-level macro:
6665 \def\@Acrfull#1#2[#3]{%

Make it easier for acronym styles to change this:
6666   \Acrfullfmt{#1}{#2}{#3}%
6667 }

\Acrfullfmt First letter upper case full format.
6668 \newcommand*{\Acrfullfmt}[3]{%
6669   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6670 }

\ACRfull
6671 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}

6672 \newcommand*\ns@ACRfull[2][]{%
6673   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6674     {\@ACRfull{#1}{#2}[]}%
6675 }

Low-level macro:
6676 \def\@ACRfull#1#2[#3]{%

```


Make it easier for acronym styles to change this:

```
6677 \ACRfullfmt{#1}{#2}{#3}%  
6678 }
```

\ACRfullfmt All upper case full format.

```
6679 \newcommand*{\ACRfullfmt}[3]{%  
6680 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6681 }
```

Plural:

\acrfullpl

```
6682 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}  
  
6683 \newcommand*\ns@acrfullpl[2][{}]{%  
6684 \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6685 {\@acrfullpl{#1}{#2}[{}]}%  
6686 }
```

Low-level macro:

```
6687 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6688 \acrfullplfmt{#1}{#2}{#3}%  
6689 }
```

\acrfullplfmt No case change plural full format.

```
6690 \newcommand*{\acrfullplfmt}[3]{%  
6691 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6692 }
```

\Acrfullpl

```
6693 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}  
  
6694 \newcommand*\ns@Acrfullpl[2][{}]{%  
6695 \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6696 {\@Acrfullpl{#1}{#2}[{}]}%  
6697 }
```

Low-level macro:

```
6698 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6699 \Acrfullplfmt{#1}{#2}{#3}%  
6700 }
```

\Acrfullplfmt First letter upper case plural full format.

```
6701 \newcommand*{\Acrfullplfmt}[3]{%  
6702 \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6703 }
```

`\ACRfullpl`

```
6704 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\@ns@ACRfullpl}
```

```
6705 \newcommand*\ns@ACRfullpl[2][\@ACRfullpl]{\@ACRfullpl{#1}{#2}}%
6706 \new@ifnextchar[\@ACRfullpl]{\@ACRfullpl{#1}{#2}}%
6707 \@ACRfullpl{#1}{#2}[]}%
6708 }
```

Low-level macro:

```
6709 \def\@ACRfullpl#1#2[#3]{%
  Make it easier for acronym styles to change this:
6710 \ACRfullplfmt{#1}{#2}{#3}%
6711 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6712 \newcommand*\ACRfullplfmt[3]{%
6713 \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%
6714 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6715 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6716 \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.

```
6717 \newcommand*\acrnameformat[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6718 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6719 \newtoks\glslabeltok
```

`\glsshorttok`

```
6720 \newtoks\glsshorttok
```

`\gslongtok`

```
6721 \newtoks\gslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
6722 \newcommand*\newacronymhook{}
```

GenericNewAcronym New improved version of setting the acronym style.

```
6723 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of \Glsentryname to workaround expansion issues that cause a problem for \makefirstuc

```
6724 \let\@Gls@entryname\@Gls@acentryname
```

Change the way acronyms are defined:

```
6725 \renewcommand{\newacronym}[4][\{%
6726   \ifdefempty{\@glsacronymlists}%
6727   {%
6728     \def\@glo@type{\acronymtype}%
6729     \setkeys{glossentry}{##1}%
6730     \DeclareAcronymList{\@glo@type}%
6731   }%
6732 }%
6733 \glskeylisttok{##1}%
6734 \glslabeltok{##2}%
6735 \glsshorttok{##3}%
6736 \glslongtok{##4}%
6737 \newacronymhook
6738 \protected@edef\@do@newglossaryentry{%
6739   \noexpand\newglossaryentry{\the\glslabeltok}%
6740   {%
6741     type=\acronymtype,%
6742     name={\expandonce{\acronymentry{##2}}},%
6743     sort={\acronymssort{\the\glsshorttok}{\the\glslongtok}},%
6744     text={\the\glsshorttok},%
6745     short={\the\glsshorttok},%
6746     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6747     long={\the\glslongtok},%
6748     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6749     \GenericAcronymFields,%
6750     \the\glskeylisttok
6751   }%
6752 }%
6753 \@do@newglossaryentry
6754 }%
```

Make sure that \acrfull etc reflects the new style:

```
6755 \renewcommand*{\acrfullfmt}[3]{%
6756   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6757 \renewcommand*{\Acrfullfmt}[3]{%
6758   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6759 \renewcommand*{\ACRfullfmt}[3]{%
6760   \glslink[##1]{##2}{%
6761     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6762 \renewcommand*{\acrfullplfmt}[3]{%
6763   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6764 \renewcommand*{\Acrfullplfmt}[3]{%
```

```

6765 \glslink{##1}{##2}{\Genplacrformat{##2}{##3}}}%
6766 \renewcommand*{\ACRfullplfmt}[3]{%
6767 \glslink{##1}{##2}{%
6768 \mfirstucMakeUppercase{\genplacrformat{##2}{##3}}}}%

```

Make sure that `\glsentryfull` etc reflects the new style:

```

6769 \renewcommand*{\glsentryfull}[1]{\genacrformat{##1}{}}%
6770 \renewcommand*{\Glsentryfull}[1]{\Genacrformat{##1}{}}%
6771 \renewcommand*{\glsentryfullpl}[1]{\genplacrformat{##1}{}}%
6772 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrformat{##1}{}}%
6773 }

```

`\icAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6774 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6775 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6776 \newcommand*{\acronymsort}[2]{#1}

```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```

6777 \newcommand*{\setacronymstyle}[1]{%
6778 \ifcsundef{@glsacr@dispstyle@#1}
6779 {%
6780 \PackageError{glossaries}{Undefined acronym style ‘#1’}{%
6781 }%
6782 {%
6783 \ifdefempty{\@glsacronymlists}%
6784 {%
6785 \DeclareAcronymList{\acronymtype}%
6786 }%
6787 }%
6788 \SetGenericNewAcronym
6789 \GlsUseAcrStyleDefs{#1}%
6790 \@for\@gls@type:=\@glsacronymlists\do{%
6791 \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6792 }%
6793 }%
6794 }

```

<code>\newacronymstyle</code>	<code>\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}</code>
-------------------------------	---

Defines a new acronym style called *<style name>*.

```

6795 \newcommand*{\newacronymstyle}[3]{%
6796   \ifcsdef{@glsacr@dispstyle@#1}%
6797   {%
6798     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6799   }%
6800   {%
6801     \csdef{@glsacr@dispstyle@#1}{#2}%
6802     \csdef{@glsacr@styledefs@#1}{#3}%
6803   }%
6804 }
```

`\renewacronymstyle` Redefines the given acronym style.

```

6805 \newcommand*{\renewacronymstyle}[3]{%
6806   \ifcsdef{@glsacr@dispstyle@#1}%
6807   {%
6808     \csdef{@glsacr@dispstyle@#1}{#2}%
6809     \csdef{@glsacr@styledefs@#1}{#3}%
6810   }%
6811   {%
6812     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6813   }%
6814 }
```

`\rEntryDispStyle`

```

6815 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

`\UseAcrStyleDefs`

```

6816 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

`long-short` *<long>* (*<short>*) acronym style.

```

6817 \newacronymstyle{long-short}%
6818 {%
```

Check for long form in case this is a mixed glossary.

```

6819   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6820 }%
6821 {%
6822   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6823   \renewcommand*{\genacrfullformat}[2]{%
6824     \glsentrylong{##1}##2\space
6825     (\protect\firstacronymfont{\glsentryshort{##1}})%
6826   }%
6827   \renewcommand*{\Genacrfullformat}[2]{%
```

```

6828 \Glsentrylong{##1}##2\space
6829 (\protect\firstacronymfont{\glsentryshort{##1}})%
6830 }%
6831 \renewcommand*{\genplacrfullformat}[2]{%
6832 \glsentrylongpl{##1}##2\space
6833 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6834 }%
6835 \renewcommand*{\Genplacrfullformat}[2]{%
6836 \Glsentrylongpl{##1}##2\space
6837 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6838 }%
6839 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6840 \renewcommand*{\acronymsort}[2]{##1}%
6841 \renewcommand*{\acronymfont}[1]{##1}%
6842 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6843 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6844 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6845 \newacronymstyle{long-sp-short}%
6846 {%
6847   \ifglshaslong{\glslabel}{\glsacfont}{\glsacfont}%
6848 }%
6849 {%
6850   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6851   \renewcommand*{\genacrfullformat}[2]{%
6852     \glsentrylong{##1}##2\glsacspace{##1}%
6853     (\protect\firstacronymfont{\glsentryshort{##1}})%
6854   }%
6855   \renewcommand*{\Genacrfullformat}[2]{%
6856     \Glsentrylong{##1}##2\glsacspace{##1}%
6857     (\protect\firstacronymfont{\glsentryshort{##1}})%
6858   }%
6859   \renewcommand*{\genplacrfullformat}[2]{%
6860     \glsentrylongpl{##1}##2\glsacspace{##1}%
6861     (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6862   }%
6863   \renewcommand*{\Genplacrfullformat}[2]{%
6864     \Glsentrylongpl{##1}##2\glsacspace{##1}%
6865     (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6866   }%
6867   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6868   \renewcommand*{\acronymsort}[2]{##1}%
6869   \renewcommand*{\acronymfont}[1]{##1}%
6870   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6871   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6872 }

```

`\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.

```
6873 \newcommand*{\glsacspace}[1]{%
6874   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
6875   \ifdim\dimen@<3em~\else\space\fi
6876 }
```

`short-long` *(short)* (*long*) acronym style.

```
6877 \newacronymstyle{short-long}%
6878 {%
```

Check for long form in case this is a mixed glossary.

```
6879   \ifglshaslong{\glslabel}{\glsacspace}{\glsacspace}%
6880 }%
6881 {%
6882   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6883   \renewcommand*{\genacrfullformat}[2]{%
6884     \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6885     (\glsentrylong{##1})%
6886   }%
6887   \renewcommand*{\Genacrfullformat}[2]{%
6888     \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6889     (\glsentrylong{##1})%
6890   }%
6891   \renewcommand*{\genplacrfullformat}[2]{%
6892     \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6893     (\glsentrylongpl{##1})%
6894   }%
6895   \renewcommand*{\Genplacrfullformat}[2]{%
6896     \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6897     (\glsentrylongpl{##1})%
6898   }%
6899   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6900   \renewcommand*{\acronymsort}[2]{##1}%
6901   \renewcommand*{\acronymfont}[1]{##1}%
6902   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6903   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6904 }
```

`long-sc-short` *(long)* (*textsc{short}*) acronym style.

```
6905 \newacronymstyle{long-sc-short}%
6906 {%
6907   \GlsUseAcrEntryDispStyle{long-short}%
6908 }%
6909 {%
6910   \GlsUseAcrStyleDefs{long-short}%
6911   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
6912   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6913 }
```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6914 \newacronymstyle{long-sm-short}%
6915 {%
6916   \GlsUseAcrEntryDisplayStyle{long-short}%
6917 }%
6918 {%
6919   \GlsUseAcrStyleDefs{long-short}%
6920   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6921   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6922 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6923 \newacronymstyle{sc-short-long}%
6924 {%
6925   \GlsUseAcrEntryDisplayStyle{short-long}%
6926 }%
6927 {%
6928   \GlsUseAcrStyleDefs{short-long}%
6929   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6930   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6931 }

```

sm-short-long *<short>* (\textsmaller{<long>}) acronym style.

```

6932 \newacronymstyle{sm-short-long}%
6933 {%
6934   \GlsUseAcrEntryDisplayStyle{short-long}%
6935 }%
6936 {%
6937   \GlsUseAcrStyleDefs{short-long}%
6938   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6939   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6940 }

```

long-short-desc *<long>* ({<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6941 \newacronymstyle{long-short-desc}%
6942 {%
6943   \GlsUseAcrEntryDisplayStyle{long-short}%
6944 }%
6945 {%
6946   \GlsUseAcrStyleDefs{long-short}%
6947   \renewcommand*{\GenericAcronymFields}{}%
6948   \renewcommand*{\acronymsort}[2]{##2}%
6949   \renewcommand*{\acronymentry}[1]{%
6950     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6951 }

```

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.


```

6952 \newacronymstyle{long-sp-short-desc}%
6953 {%
6954   \GlsUseAcrEntryDisplayStyle{long-sp-short}%
6955 }%
6956 {%
6957   \GlsUseAcrStyleDefs{long-sp-short}%
6958   \renewcommand*{\GenericAcronymFields}{}%
6959   \renewcommand*{\acronymsort}[2]{##2}%
6960   \renewcommand*{\acronymentry}[1]{%
6961     \glentrylong{##1}\glspacespace{##1}(\acronymfont{\glentryshort{##1}})}%
6962 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6963 \newacronymstyle{long-sc-short-desc}%
6964 {%
6965   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
6966 }%
6967 {%
6968   \GlsUseAcrStyleDefs{long-sc-short}%
6969   \renewcommand*{\GenericAcronymFields}{}%
6970   \renewcommand*{\acronymsort}[2]{##2}%
6971   \renewcommand*{\acronymentry}[1]{%
6972     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6973 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6974 \newacronymstyle{long-sm-short-desc}%
6975 {%
6976   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
6977 }%
6978 {%
6979   \GlsUseAcrStyleDefs{long-sm-short}%
6980   \renewcommand*{\GenericAcronymFields}{}%
6981   \renewcommand*{\acronymsort}[2]{##2}%
6982   \renewcommand*{\acronymentry}[1]{%
6983     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6984 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

6985 \newacronymstyle{short-long-desc}%
6986 {%
6987   \GlsUseAcrEntryDisplayStyle{short-long}%
6988 }%
6989 {%
6990   \GlsUseAcrStyleDefs{short-long}%
6991   \renewcommand*{\GenericAcronymFields}{}%

```

```

6992 \renewcommand*{\acronymsort}[2]{##2}%
6993 \renewcommand*{\acronymentry}[1]{%
6994     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6995 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6996 \newacronymstyle{sc-short-long-desc}%
6997 {%
6998     \GlsUseAcrEntryDispStyle{sc-short-long}%
6999 }%
7000 {%
7001     \GlsUseAcrStyleDefs{sc-short-long}%
7002     \renewcommand*{\GenericAcronymFields}{}%
7003     \renewcommand*{\acronymsort}[2]{##2}%
7004     \renewcommand*{\acronymentry}[1]{%
7005         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7006 }

```

short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

7007 \newacronymstyle{sm-short-long-desc}%
7008 {%
7009     \GlsUseAcrEntryDispStyle{sm-short-long}%
7010 }%
7011 {%
7012     \GlsUseAcrStyleDefs{sm-short-long}%
7013     \renewcommand*{\GenericAcronymFields}{}%
7014     \renewcommand*{\acronymsort}[2]{##2}%
7015     \renewcommand*{\acronymentry}[1]{%
7016         \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7017 }

```

dua *<long>* only acronym style.

```

7018 \newacronymstyle{dua}%
7019 {%

```

Check for long form in case this is a mixed glossary.

```

7020 \ifdefempty\glscustomtext
7021 {%
7022     \ifglshaslong{\glslabel}%
7023     {%
7024         \glcifplural
7025         {%

```

Plural form:

```

7026         \glscapscase
7027         {%

```

Plural form, don't adjust case:

```
7028      \glentrylongpl{\glslabel}\glsinsert
7029      }%
7030      {%
```

Plural form, make first letter upper case:

```
7031      \Glentrylongpl{\glslabel}\glsinsert
7032      }%
7033      {%
```

Plural form, all caps:

```
7034      \mfirstucMakeUppercase
7035      {\glentrylongpl{\glslabel}\glsinsert}%
7036      }%
7037      }%
7038      {%
```

Singular form

```
7039      \glscapscase
7040      {%
```

Singular form, don't adjust case:

```
7041      \glentrylong{\glslabel}\glsinsert
7042      }%
7043      {%
```

Subsequent singular form, make first letter upper case:

```
7044      \Glentrylong{\glslabel}\glsinsert
7045      }%
7046      {%
```

Subsequent singular form, all caps:

```
7047      \mfirstucMakeUppercase
7048      {\glentrylong{\glslabel}\glsinsert}%
7049      }%
7050      }%
7051      }%
7052      {%
```

Not an acronym:

```
7053      \glsgenentryfmt
7054      }%
7055      }%
7056      {\glscustomtext\glsinsert}%
7057      }%
7058      {%
7059      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

7060      \renewcommand*{\acrfullfmt}[3]{%
7061      \glslink{##1}{##2}{\glentrylong{##2}##3\space
7062      (\acronymfont{\glentryshort{##2}})}}%
7063      \renewcommand*{\Acrfullfmt}[3]{%
```

```

7064 \glslink{##1}{##2}{\Glsentrylong{##2}##3\space
7065 (\acronymfont{\glsentryshort{##2}})}}}%
7066 \renewcommand*{\ACRfullfmt}[3]{%
7067 \glslink{##1}{##2}{%
7068 \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
7069 (\acronymfont{\glsentryshort{##2}})}}}%

7070 \renewcommand*{\acrfullplfmt}[3]{%
7071 \glslink{##1}{##2}{\glsentrylongpl{##2}##3\space
7072 (\acronymfont{\glsentryshortpl{##2}})}}}%

7073 \renewcommand*{\Acrfullplfmt}[3]{%
7074 \glslink{##1}{##2}{\Glsentrylongpl{##2}##3\space
7075 (\acronymfont{\glsentryshortpl{##2}})}}}%
7076 \renewcommand*{\ACRfullplfmt}[3]{%
7077 \glslink{##1}{##2}{%
7078 \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
7079 (\acronymfont{\glsentryshortpl{##2}})}}}%
7080 \renewcommand*{\glsentryfull}[1]{%
7081 \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
7082 }%
7083 \renewcommand*{\Glsentryfull}[1]{%
7084 \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
7085 }%
7086 \renewcommand*{\glsentryfullpl}[1]{%
7087 \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
7088 }%
7089 \renewcommand*{\Glsentryfullpl}[1]{%
7090 \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
7091 }%
7092 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
7093 \renewcommand*{\acronymsort}[2]{##1}%
7094 \renewcommand*{\acronymfont}[1]{##1}%
7095 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7096 }

```

dua-desc <long> only acronym style with user-supplied description.

```

7097 \newacronymstyle{dua-desc}%
7098 {%
7099 \GlsUseAcrEntryDispStyle{dua}%
7100 }%
7101 {%
7102 \GlsUseAcrStyleDefs{dua}%
7103 \renewcommand*{\GenericAcronymFields}{}%

7104 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
7105 \renewcommand*{\acronymsort}[2]{##2}%
7106 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```
7107 \newacronymstyle{footnote}%  
7108 {%
```

Check for long form in case this is a mixed glossary.

```
7109 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%  
7110 }%  
7111 {%  
7112 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
7113 \glshyperfirstfalse  
7114 \renewcommand*{\genacrfullformat}[2]{%  
7115 \protect\firstacronymfont{\glsentryshort{##1}}##2%  
7116 \protect\footnote{\glsentrylong{##1}}%  
7117 }%  
7118 \renewcommand*{\Genacrfullformat}[2]{%  
7119 \firstacronymfont{\Glsentryshort{##1}}##2%  
7120 \protect\footnote{\glsentrylong{##1}}%  
7121 }%  
7122 \renewcommand*{\genplacrfullformat}[2]{%  
7123 \protect\firstacronymfont{\glsentryshortpl{##1}}##2%  
7124 \protect\footnote{\glsentrylongpl{##1}}%  
7125 }%  
7126 \renewcommand*{\Genplacrfullformat}[2]{%  
7127 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%  
7128 \protect\footnote{\glsentrylongpl{##1}}%  
7129 }%  
7130 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%  
7131 \renewcommand*{\acronymsort}[2]{##1}%  
7132 \renewcommand*{\acronymfont}[1]{##1}%  
7133 \renewcommand*{\acrpluralsuffix}{\glssacrpluralsuffix}%
```

Don't use footnotes for \acrfull:

```
7134 \renewcommand*{\acrfullfmt}[3]{%  
7135 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space  
7136 (\glsentrylong{##2})}%  
7137 \renewcommand*{\Acrfullfmt}[3]{%  
7138 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space  
7139 (\glsentrylong{##2})}%  
7140 \renewcommand*{\ACRfullfmt}[3]{%  
7141 \glslink[##1]{##2}{%  
7142 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space  
7143 (\glsentrylong{##2})}}}%  
7144 \renewcommand*{\acrfullplfmt}[3]{%  
7145 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space  
7146 (\glsentrylongpl{##2})}%  
7147 \renewcommand*{\Acrfullplfmt}[3]{%  
7148 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space  
7149 (\glsentrylongpl{##2})}%
```

```

7150 \renewcommand*{\ACRfullplfmt}[3]{%
7151   \glslink{##1}{##2}{%
7152     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
7153     (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

7154 \renewcommand*{\glsentryfull}[1]{%
7155   \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
7156 \renewcommand*{\Glsentryfull}[1]{%
7157   \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
7158 \renewcommand*{\glsentryfullpl}[1]{%
7159   \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7160 \renewcommand*{\Glsentryfullpl}[1]{%
7161   \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7162 }

```

footnote-sc \textsc{\<short>}\footnote{\<long>} acronym style.

```

7163 \newacronymstyle{footnote-sc}%
7164 {%
7165   \GlsUseAcrEntryDisplayStyle{footnote}%
7166 }%
7167 {%
7168   \GlsUseAcrStyleDefs{footnote}%
7169   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
7170   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
7171   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7172 }%

```

footnote-sm \textsmaller{\<short>}\footnote{\<long>} acronym style.

```

7173 \newacronymstyle{footnote-sm}%
7174 {%
7175   \GlsUseAcrEntryDisplayStyle{footnote}%
7176 }%
7177 {%
7178   \GlsUseAcrStyleDefs{footnote}%
7179   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
7180   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
7181   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7182 }%

```

footnote-desc \<short>\footnote{\<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7183 \newacronymstyle{footnote-desc}%
7184 {%
7185   \GlsUseAcrEntryDisplayStyle{footnote}%
7186 }%
7187 {%
7188   \GlsUseAcrStyleDefs{footnote}%
7189   \renewcommand*{\GenericAcronymFields}{}%

```

```

7190 \renewcommand*{\acronymsort}[2]{##2}%
7191 \renewcommand*{\acronymentry}[1]{%
7192   \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7193 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7194 \newacronymstyle{footnote-sc-desc}%
7195 {%
7196   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
7197 }%
7198 {%
7199   \GlsUseAcrStyleDefs{footnote-sc}%
7200   \renewcommand*{\GenericAcronymFields}{}%
7201   \renewcommand*{\acronymsort}[2]{##2}%
7202   \renewcommand*{\acronymentry}[1]{%
7203     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7204 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7205 \newacronymstyle{footnote-sm-desc}%
7206 {%
7207   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
7208 }%
7209 {%
7210   \GlsUseAcrStyleDefs{footnote-sm}%
7211   \renewcommand*{\GenericAcronymFields}{}%
7212   \renewcommand*{\acronymsort}[2]{##2}%
7213   \renewcommand*{\acronymentry}[1]{%
7214     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
7215 }

```

AcronymSynonyms

```

7216 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

\acs

```

7217 \let\acs\acrshort

```

First letter uppercase short form

\Acs

```

7218 \let\Acs\Acrshort

```

Plural short form

\acsp

```

7219 \let\acsp\acrshortpl

```

First letter uppercase plural short form

\Acsp

7220 \let\Acsp\Acrshorttpl

Long form

\acl

7221 \let\acl\acrlong

Plural long form

\aclp

7222 \let\aclp\acrlongpl

First letter upper case long form

\Acl

7223 \let\Acl\Acrlong

First letter upper case plural long form

\Aclp

7224 \let\Aclp\Acrlongpl

Full form

\acf

7225 \let\acf\acrfull

Plural full form

\acfp

7226 \let\acfp\acrfullpl

First letter upper case full form

\Acf

7227 \let\Acf\Acrfull

First letter upper case plural full form

\Acfp

7228 \let\Acfp\Acrfullpl

Standard form

\ac

7229 \let\ac\gls

First upper case standard form

\Ac

7230 \let\Ac\Gls

Standard plural form

`\acp`

```
7231 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
7232 \let\Acp\Glspl
```

```
7233 }
```

Define synonyms if required

```
7234 \ifglsacrshortcuts
```

```
7235 \DefineAcronymSynonyms
```

```
7236 \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.

```
7237 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
```

```
7238 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
7239 }
```

`\glsNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
7240 \newcommand*{\DefaultNewAcronymDef}{%
```

```
7241 \edef\@do@newglossaryentry{%
```

```
7242 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
7243 {%
```

```
7244 type=\acronymtype,%
```

```
7245 name={\the\glsshorttok},%
```

```
7246 sort={\the\glsshorttok},%
```

```
7247 text={\the\glsshorttok},%
```

```
7248 first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
```

```
7249 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
```

```
7250 firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
```

```
7251 {\noexpand\expandonce\noexpand\@glo@shortpl}},%
```

```
7252 short={\the\glsshorttok},%
```

```
7253 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
```

```
7254 long={\the\glslongtok},%
```

```
7255 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
```

```
7256 description={\the\glslongtok},%
```

```
7257 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
7258 \the\glskeylisttok
```

```
7259 }%
```

```
7260 }%
```

```
7261 \let\@org@gls@assign@firstpl\gls@assign@firstpl
```

```

7262 \let\@org@gl@s@assign@plural\gl@s@assign@plural
7263 \let\@org@gl@s@assign@descplural\gl@s@assign@descplural
7264 \def\gl@s@assign@firstpl##1##2{%
7265   \@@gl@s@expand@field{##1}{firstpl}{##2}%
7266 }%
7267 \def\gl@s@assign@plural##1##2{%
7268   \@@gl@s@expand@field{##1}{plural}{##2}%
7269 }%
7270 \def\gl@s@assign@descplural##1##2{%
7271   \@@gl@s@expand@field{##1}{descplural}{##2}%
7272 }%
7273 \do@newglossaryentry
7274 \let\gl@s@assign@firstpl\@org@gl@s@assign@firstpl
7275 \let\gl@s@assign@plural\@org@gl@s@assign@plural
7276 \let\gl@s@assign@symbolplural\@org@gl@s@assign@symbolplural
7277 }

```

ultAcronymStyle Set up the default acronym style:

```
7278 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```

7279   \@for\@gl@s@type:=\@gl@sacronymlists\do{%
7280     \SetDefaultAcronymDisplayStyle{\@gl@s@type}%
7281   }%

```

Set up the definition of `\newacronym`:

```
7282 \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).

```

7283   \ifx\@gl@sacronymlists\@empty
7284     \def\@glo@type{\acronymtype}%
7285     \setkeys{glossentry}{##1}%
7286     \DeclareAcronymList{\@glo@type}%
7287     \SetDefaultAcronymDisplayStyle{\@glo@type}%
7288   \fi
7289   \gl@keylisttok{##1}%
7290   \gl@labeltok{##2}%
7291   \gl@shorttok{##3}%
7292   \gl@longtok{##4}%
7293   \newacronymhook
7294   \DefaultNewAcronymDef
7295 }%
7296 \renewcommand*{\acrpluralsuffix}{\gl@sacrpluralsuffix}%
7297 }

```

\acrfootnote Used by the footnote acronym styles.

```
7298 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

acrlinkfootnote

```

7299 \newcommand*{\acrlinkfootnote}[3]{%
7300   \footnote{\glslink[#1]{#2}{#3}}%
7301 }

```

acrnolinkfootnote

```

7302 \newcommand*{\acrnolinkfootnote}[3]{%
7303   \footnote{#3}%
7304 }

```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```

7305 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7306   \def\glsentryfmt[#1]{%

7307     \ifdefempty\glscustomtext
7308     {%
7309       \ifglssused{\glslabel}%
7310       {%
7311         \acronymfont{\glsentryfmt}%
7312       }%
7313       {%
7314         \firstacronymfont{\glsentryfmt}%
7315         \ifglshassymbol{\glslabel}%
7316         {%
7317           \expandafter\protect\expandafter\acrfootnote\expandafter
7318             {\@gls@link@opts}{\@gls@link@label}%
7319           {%
7320             \glsifplural
7321               {\glsentrysymbolplural{\glslabel}}%
7322               {\glsentrysymbol{\glslabel}}%
7323             }%
7324           }%
7325         }%
7326       }%
7327     {\glscustomtext\glsinsert}%
7328   }%
7329 }

```

acronymNewAcronymDef

```

7330 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7331   \edef\@do@newglossaryentry{%
7332     \noexpand\newglossaryentry{\the\glslabeltok}%
7333     {%
7334       type=\acronymtype,%
7335       name={\noexpand\acronymfont{\the\glsshorttok}},%
7336       sort={\the\glsshorttok},%
7337       first={\the\glsshorttok},%
7338       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7339       text={\the\glsshorttok},%

```

```

7340 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7341 short={\the\glsshorttok},%
7342 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7343 long={\the\glslongtok},%
7344 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7345 symbol={\the\glslongtok},%
7346 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7347 \the\glskeylisttok
7348 }%
7349 }%
7350 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7351 \let\@org@gls@assign@plural\gls@assign@plural
7352 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7353 \def\gls@assign@firstpl##1##2{%
7354   \@@gls@expand@field{##1}{firstpl}{##2}%
7355 }%
7356 \def\gls@assign@plural##1##2{%
7357   \@@gls@expand@field{##1}{plural}{##2}%
7358 }%
7359 \def\gls@assign@symbolplural##1##2{%
7360   \@@gls@expand@field{##1}{symbolplural}{##2}%
7361 }%
7362 \do@newglossaryentry
7363 \let\gls@assign@plural\@org@gls@assign@plural
7364 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7365 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7366 }

```

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.

```

7367 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7368   \renewcommand{\newacronym}[4][\]{%
7369     \ifx\@glsacronymlists\@empty
7370       \def\@glo@type{\acronymtype}%
7371       \setkeys{glossentry}{##1}%
7372       \DeclareAcronymList{\@glo@type}%
7373       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7374     \fi
7375     \glskeylisttok{##1}%
7376     \glslabeltok{##2}%
7377     \glsshorttok{##3}%
7378     \glslongtok{##4}%
7379     \newacronymhook
7380     \DescriptionFootnoteNewAcronymDef
7381   }%

```

If footnote package option is specified, set the first use to append the long form (stored in

symbol) as a footnote.

```
7382 \for\@gls@type:=\@glsacronymlists\do{%
7383   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7384 }%
```

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.

```
7385 \ifglsacrsmallcaps
7386   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7387   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7388 \else
7389   \ifglsacrsmaller
7390     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7391   \fi
7392 \fi
```

Check for package option clash

```
7393 \ifglsacrdua
7394   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7395     can’t both be set}{}%
7396 \fi
7397 }%
```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```
7398 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7399   \defglsentryfmt[##1]{\glsentryfmt}%
7400 }
```

UANewAcronymDef

```
7401 \newcommand*{\DescriptionDUANewAcronymDef}{%
7402   \edef\@do@newglossaryentry{%
7403     \noexpand\newglossaryentry{\the\glslabeltok}%
7404     {%
7405       type=\acronymtype,%
7406       name={\the\glslongtok},%
7407       sort={\the\glslongtok},%
7408       text={\the\glslongtok},%
7409       first={\the\glslongtok},%
7410       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7411       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7412       short={\the\glsshorttok},%
7413       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7414       long={\the\glslongtok},%
7415       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7416       symbol={\the\glsshorttok},%
7417       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7418       \the\glskeylisttok
7419     }%
7420   }%
```

```

7421 \let\@org@gl@s@assign@firstpl\gl@s@assign@firstpl
7422 \let\@org@gl@s@assign@plural\gl@s@assign@plural
7423 \let\@org@gl@s@assign@symbolplural\gl@s@assign@symbolplural
7424 \def\gl@s@assign@firstpl##1##2{%
7425   \@@gl@s@expand@field{##1}{firstpl}{##2}%
7426 }%
7427 \def\gl@s@assign@plural##1##2{%
7428   \@@gl@s@expand@field{##1}{plural}{##2}%
7429 }%
7430 \def\gl@s@assign@symbolplural##1##2{%
7431   \@@gl@s@expand@field{##1}{symbolplural}{##2}%
7432 }%
7433 \do@newglossaryentry
7434 \let\gl@s@assign@firstpl\@org@gl@s@assign@firstpl
7435 \let\gl@s@assign@plural\@org@gl@s@assign@plural
7436 \let\gl@s@assign@symbolplural\@org@gl@s@assign@symbolplural
7437 }

```

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.

```

7438 \newcommand*{\SetDescriptionDUAACronymStyle}{%
7439   \ifgl@sacrsmallcaps
7440     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7441       can't both be set}{}%
7442   \else
7443     \ifgl@sacrsmaller
7444       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7445         can't both be set}{}%
7446     \fi
7447   \fi
7448   \renewcommand{\newacronym}[4][{}]{%
7449     \ifx\@gl@sacronymlists\@empty
7450       \def\@glo@type{\acronymtype}%
7451       \setkeys{glossentry}{##1}%
7452       \DeclareAcronymList{\@glo@type}%
7453       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
7454     \fi
7455     \gl@keylisttok{##1}%
7456     \gl@labeltok{##2}%
7457     \gl@shorttok{##3}%
7458     \gl@longtok{##4}%
7459     \newacronymhook
7460     \DescriptionDUANewAcronymDef
7461   }%

```

Set display.

```

7462 \@for\@gl@s@type:=\@gl@sacronymlists\do{%
7463   \SetDescriptionDUAACronymDisplayStyle{\@gl@s@type}%

```

```

7464 }%
7465 }%

```

nymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7466 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7467   \def\glsentryfmt[#1]{%

7468     \ifdefempty\glscustomtext
7469     {%
7470       \ifglssused{\glslabel}%
7471       {%

```

Move the inserted text outside of \acronymfont

```

7472       \let\gls@org@insert\glsinsert
7473       \let\glsinsert\@empty
7474       \acronymfont{\glsentryfmt}\gls@org@insert
7475     }%
7476   {%
7477     \glsentryfmt
7478     \ifglshassymbol{\glslabel}%
7479     {%
7480       \glsifplural
7481       {%
7482         \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7483       }%
7484       {%
7485         \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7486       }%
7487       \space(\protect\firstacronymfont
7488       {\glscapscase
7489       {\@glo@symbol}
7490       {\@glo@symbol}
7491       {\mfirstucMakeUppercase{\@glo@symbol}}})%
7492     }%
7493   }%
7494 }%
7495 }%
7496 {\glscustomtext\glsinsert}%
7497 }%
7498 }

```

onNewAcronymDef

```

7499 \newcommand*{\DescriptionNewAcronymDef}{%
7500   \edef\@do@newglossaryentry{%
7501     \noexpand\newglossaryentry{\the\glslabeltok}%
7502     {%
7503       type=\acronymtype,%
7504       name={\noexpand

```

```

7505     \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
7506     sort={\the\glsshorttok},%
7507     first={\the\glslongtok},%
7508     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7509     text={\the\glsshorttok},%
7510     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7511     short={\the\glsshorttok},%
7512     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7513     long={\the\glslongtok},%
7514     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7515     symbol={\noexpand\@glo@text},%
7516     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7517     \the\glskeylisttok}%
7518 }%
7519 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7520 \let\@org@gls@assign@plural\gls@assign@plural
7521 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7522 \def\gls@assign@firstpl##1##2{%
7523   \@@gls@expand@field{##1}{firstpl}{##2}%
7524 }%
7525 \def\gls@assign@plural##1##2{%
7526   \@@gls@expand@field{##1}{plural}{##2}%
7527 }%
7528 \def\gls@assign@symbolplural##1##2{%
7529   \@@gls@expand@field{##1}{symbolplural}{##2}%
7530 }%
7531 \do@newglossaryentry
7532 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7533 \let\gls@assign@plural\@org@gls@assign@plural
7534 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7535 }

```

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 \acronymformat to allow the user to override the way the name is displayed in the list of acronyms.

```

7536 \newcommand*\SetDescriptionAcronymStyle{%
7537   \renewcommand{\newacronym}[4][]{%
7538     \ifx\@glsacronymlists\@empty
7539       \def\@glo@type{\acronymtype}%
7540       \setkeys{glossentry}{##1}%
7541       \DeclareAcronymList{\@glo@type}%
7542       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7543     \fi
7544     \glskeylisttok{##1}%
7545     \glslabeltok{##2}%
7546     \glsshorttok{##3}%
7547     \glslongtok{##4}%
7548     \newacronymhook
7549     \DescriptionNewAcronymDef

```


7550 }%

Set display.

```
7551 \@for\@gls@type:=\@glsacronymlists\do{%
7552   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7553 }%
```

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.

```
7554 \ifglsacrsmallcaps
7555   \renewcommand{\acronymfont}[1]{\textsc{##1}}
7556   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7557 \else
7558   \ifglsacrsmaller
7559     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7560   \fi
7561 \fi
7562 }%
```

`\glsacronymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```
7563 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7564   \defglsentryfmt[#1]{%
```

```
7565     \ifdefempty\glscustomtext
7566     {%
```

Move the inserted text outside of \acronymfont

```
7567     \let\gls@org@insert\glsinsert
7568     \let\glsinsert\@empty
7569     \ifglsused{\glslabel}%
7570     {%
7571       \acronymfont{\glsentryfmt}\gls@org@insert
7572     }%
7573     {%
7574       \firstacronymfont{\glsentryfmt}\gls@org@insert
7575       \ifglschaslong{\glslabel}%
7576       {%
7577         \expandafter\protect\expandafter\acrfootnote\expandafter
7578         {\@gls@link@opts}{\@gls@link@label}%
7579         {%
7580           \glsifplural
7581             {\glsentrylongpl{\glslabel}}%
7582             {\glsentrylong{\glslabel}}%
7583           }%
7584         }%
7585       }%
7586     }%
7587   }%
```

```

7588     {\glscustomtext\glsinsert}%
7589   }%
7590 }

```

teNewAcronymDef

```

7591 \newcommand*{\FootnoteNewAcronymDef}{%
7592   \edef\@do@newglossaryentry{%
7593     \noexpand\newglossaryentry{\the\glslabeltok}%
7594     {%
7595       type=\acronymtype,%
7596       name={\noexpand\acronymfont{\the\glsshorttok}},%
7597       sort={\the\glsshorttok},%
7598       text={\the\glsshorttok},%
7599       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7600       first={\the\glsshorttok},%
7601       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7602       short={\the\glsshorttok},%
7603       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7604       long={\the\glslongtok},%
7605       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7606       description={\the\glslongtok},%
7607       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7608       \the\glskeylisttok
7609     }%
7610   }%
7611   \let\@org@gls@assign@plural\gls@assign@plural
7612   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7613   \let\@org@gls@assign@descplural\gls@assign@descplural
7614   \def\gls@assign@firstpl##1##2{%
7615     \@@gls@expand@field{##1}{firstpl}{##2}%
7616   }%
7617   \def\gls@assign@plural##1##2{%
7618     \@@gls@expand@field{##1}{plural}{##2}%
7619   }%
7620   \def\gls@assign@descplural##1##2{%
7621     \@@gls@expand@field{##1}{descplural}{##2}%
7622   }%
7623   \@do@newglossaryentry
7624   \let\gls@assign@plural\@org@gls@assign@plural
7625   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7626   \let\gls@assign@descplural\@org@gls@assign@descplural
7627 }

```

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.

```

7628 \newcommand*{\SetFootnoteAcronymStyle}{%
7629   \renewcommand{\newacronym}[4][\]{%
7630     \ifx\@glsacronymlists\empty
7631       \def\@glo@type{\acronymtype}%

```

```

7632     \setkeys{glossentry}{##1}%
7633     \DeclareAcronymList{\@glo@type}%
7634     \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7635     \fi
7636     \glskeylisttok{##1}%
7637     \glslabeltok{##2}%
7638     \glsshorttok{##3}%
7639     \glslongtok{##4}%
7640     \newacronymhook
7641     \FootnoteNewAcronymDef
7642 }%

```

Set display

```

7643 \@for\@gls@type:=\@glsacronymlists\do{%
7644   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7645 }%

```

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.

```

7646 \ifglsacrsmallcaps
7647   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7648   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7649 \else
7650   \ifglsacrsmaller
7651     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7652   \fi
7653 \fi

```

Check for option clash

```

7654 \ifglsacrdua
7655   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7656     can’t both be set}{}%
7657 \fi
7658 }%

```

`\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.

```

7659 \DeclareRobustCommand*\glsdoparenifnotempty[2]{%
7660   \protected@edef\gls@tmp{##1}%
7661   \ifdefempty\gls@tmp
7662   {}%
7663   {%
7664     \ifx\gls@tmp\@gls@default@value
7665     \else
7666       \space (#2{##1})%
7667     \fi
7668   }%
7669 }

```

`\acronymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7670 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7671   \def\glsentryfmt[#1]{%

7672     \ifdefempty\glscustomtext
7673     {%

      Move the inserted text outside of \acronymfont

7674       \let\gls@org@insert\glsinsert
7675       \let\glsinsert\@empty
7676       \ifglsused{\glslabel}%
7677       {%
7678         \acronymfont{\glsentryfmt}\gls@org@insert
7679       }%
7680       {%
7681         \glsentryfmt
7682         \ifgls hassymbol{\glslabel}%
7683         {%
7684           \glsifplural
7685           {%
7686             \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7687           }%
7688           {%
7689             \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7690           }%
7691           \space
7692           (\gls caps case
7693             {\firstacronymfont{\@glo@symbol}}%
7694             {\firstacronymfont{\@glo@symbol}}%
7695             {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7696           }%
7697         }%
7698       }%
7699     }%
7700     {\gls customtext\glsinsert}%
7701   }%
7702 }
```

`\allNewAcronymDef`

```

7703 \newcommand*{\SmallNewAcronymDef}{%
7704   \edef\@do@newglossaryentry{%
7705     \noexpand\newglossaryentry{\the\glslabeltok}%
7706     {%
7707       type=\acronymtype,%
7708       name={\noexpand\acronymfont{\the\glsshorttok}},%
7709       sort={\the\glsshorttok},%
7710       text={\the\glsshorttok},%
```

Default to the short plural.

```

7711 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7712 first={\the\glslongtok},%
    Default to the long plural.
7713 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7714 short={\the\glsshorttok},%
7715 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7716 long={\the\glslongtok},%
7717 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7718 description={\noexpand\@glo@first},%
7719 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7720 symbol={\the\glsshorttok},%
    Default to the short plural.
7721 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7722 \the\glskeylisttok
7723 }%
7724 }%
7725 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7726 \let\@org@gls@assign@plural\gls@assign@plural
7727 \let\@org@gls@assign@descplural\gls@assign@descplural
7728 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7729 \def\gls@assign@firstpl##1##2{%
7730 \@@gls@expand@field{##1}{firstpl}{##2}%
7731 }%
7732 \def\gls@assign@plural##1##2{%
7733 \@@gls@expand@field{##1}{plural}{##2}%
7734 }%
7735 \def\gls@assign@descplural##1##2{%
7736 \@@gls@expand@field{##1}{descplural}{##2}%
7737 }%
7738 \def\gls@assign@symbolplural##1##2{%
7739 \@@gls@expand@field{##1}{symbolplural}{##2}%
7740 }%
7741 \do@newglossaryentry
7742 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7743 \let\gls@assign@plural\@org@gls@assign@plural
7744 \let\gls@assign@descplural\@org@gls@assign@descplural
7745 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7746 }

```

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.

```

7747 \newcommand*{\SetSmallAcronymStyle}{%
7748 \renewcommand{\newacronym}[4][\]{%
7749 \ifx\@glsacronymlists\@empty
7750 \def\@glo@type{\acronymtype}%
7751 \setkeys{glossentry}{##1}%
7752 \DeclareAcronymList{\@glo@type}%
7753 \SetSmallAcronymDisplayStyle{\@glo@type}%

```

```

7754 \fi
7755 \glskeylisttok{##1}%
7756 \glslabeltok{##2}%
7757 \glsshorttok{##3}%
7758 \glslongtok{##4}%
7759 \newacronymhook
7760 \SmallNewAcronymDef
7761 }%

```

Change the display since first only contains long form.

```

7762 \@for\@gls@type:=\@glsacronymlists\do{%
7763 \SetSmallAcronymDisplayStyle{\@gls@type}%
7764 }%

```

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.

```

7765 \ifglsacrsmallcaps
7766 \renewcommand*\acronymfont[1]{\textsc{##1}}
7767 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7768 \else
7769 \renewcommand*\acronymfont[1]{\textsmaller{##1}}
7770 \fi

```

check for option clash

```

7771 \ifglsacrdua
7772 \ifglsacrsmallcaps
7773 \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7774 can’t both be set}{}%
7775 \else
7776 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7777 can’t both be set}{}%
7778 \fi
7779 \fi
7780 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7781 \newcommand*\SetDUADisplayStyle[1]{%
7782 \defglsentryfmt[1]{\glsgenentryfmt}%
7783 }

```

UANewAcronymDef

```

7784 \newcommand*\DUANewAcronymDef{%
7785 \edef\@do@newglossaryentry{%
7786 \noexpand\newglossaryentry{\the\glslabeltok}%
7787 {%
7788 type=\acronymtype,%
7789 name={\the\glsshorttok},%
7790 text={\the\glslongtok},%
7791 first={\the\glslongtok},%
7792 plural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

```

7793     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7794     short={\the\glsshorttok},%
7795     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7796     long={\the\glslongtok},%
7797     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7798     description={\the\glslongtok},%
7799     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7800     symbol={\the\glsshorttok},%
7801     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7802     \the\glskeylisttok
7803   }%
7804 }%
7805 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7806 \let\@org@gls@assign@plural\gls@assign@plural
7807 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7808 \let\@org@gls@assign@descplural\gls@assign@descplural
7809 \def\gls@assign@firstpl##1##2{%
7810   \@@gls@expand@field{##1}{firstpl}{##2}%
7811 }%
7812 \def\gls@assign@plural##1##2{%
7813   \@@gls@expand@field{##1}{plural}{##2}%
7814 }%
7815 \def\gls@assign@symbolplural##1##2{%
7816   \@@gls@expand@field{##1}{symbolplural}{##2}%
7817 }%
7818 \def\gls@assign@descplural##1##2{%
7819   \@@gls@expand@field{##1}{descplural}{##2}%
7820 }%
7821 \do@newglossaryentry
7822 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7823 \let\gls@assign@plural\@org@gls@assign@plural
7824 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7825 \let\gls@assign@descplural\@org@gls@assign@descplural
7826 }

```

\SetDUASStyle Always expand acronyms.

```

7827 \newcommand*{\SetDUASStyle}{%
7828   \renewcommand{\newacronym}[4][]{%
7829     \ifx\@glsacronymlists\@empty
7830       \def\@glo@type{\acronymtype}%
7831       \setkeys{glossentry}{##1}%
7832       \DeclareAcronymList{\@glo@type}%
7833       \SetDUADisplayStyle{\@glo@type}%
7834     \fi
7835     \glskeylisttok{##1}%
7836     \glslabeltok{##2}%
7837     \glsshorttok{##3}%
7838     \glslongtok{##4}%
7839     \newacronymhook

```

```

7840     \DUANewAcronymDef
7841 }%

Set the display
7842 \@for\@gls@type:=\@glsacronymlists\do{%
7843     \SetDUADisplayStyle{\@gls@type}%
7844 }%
7845 }

```

SetAcronymStyle

```

7846 \newcommand*\SetAcronymStyle{%
7847     \SetDefaultAcronymStyle
7848     \ifglsacrdescription
7849         \ifglsacrfootnote
7850             \SetDescriptionFootnoteAcronymStyle
7851         \else
7852             \ifglsacrdua
7853                 \SetDescriptionDUAAcronymStyle
7854             \else
7855                 \SetDescriptionAcronymStyle
7856             \fi
7857         \fi
7858     \else
7859         \ifglsacrfootnote
7860             \SetFootnoteAcronymStyle
7861         \else
7862             \ifthenelse{\boolean{glsacrsmalldcaps}\OR
7863                 \boolean{glsacrsmaller}}{%
7864                 {%
7865                     \SetSmallAcronymStyle
7866                 }%
7867             {%
7868                 \ifglsacrdua
7869                     \SetDUASyle
7870                 \fi
7871             }%
7872         \fi
7873     \fi
7874 }

```

Set the acronym style according to the package options

```
7875 \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.

```
7876 \newcommand*\SetCustomDisplayStyle}[1]{%
```



```

7877 \defglentryfmt[#1]{\glsgenentryfmt}%
7878 }

```

omAcronymFields

```

7879 \newcommand*{\CustomAcronymFields}{%
7880   name={\the\glsshorttok},%
7881   description={\the\glslongtok},%
7882   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7883   firstplural={\acrfullformat
7884     {\noexpand\glentrylongpl{\the\glslabeltok}}%
7885     {\noexpand\glentryshortpl{\the\glslabeltok}}},%

7886   text={\the\glsshorttok},%
7887   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7888 }

```

omNewAcronymDef

```

7889 \newcommand*{\CustomNewAcronymDef}{%
7890   \protected@edef\@do@newglossaryentry{%
7891     \noexpand\newglossaryentry{\the\glslabeltok}%
7892     {%
7893       type=\acronymtype,%
7894       short={\the\glsshorttok},%
7895       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7896       long={\the\glslongtok},%
7897       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7898       user1={\the\glsshorttok},%
7899       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7900       user3={\the\glslongtok},%
7901       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7902       \CustomAcronymFields,%
7903       \the\glskeylisttok
7904     }%
7905   }%
7906   \@do@newglossaryentry
7907 }

```

\SetCustomStyle

```

7908 \newcommand*{\SetCustomStyle}{%
7909   \renewcommand{\newacronym}[4][\{%
7910     \ifx\@glsacronymlists\@empty
7911       \def\@glo@type{\acronymtype}%
7912       \setkeys{glossentry}{##1}%
7913       \DeclareAcronymList{\@glo@type}%
7914       \SetCustomDisplayStyle{\@glo@type}%
7915     \fi
7916     \glskeylisttok{##1}%
7917     \glslabeltok{##2}%
7918     \glsshorttok{##3}%

```

```

7919 \glslongtok{##4}%
7920 \newacronymhook
7921 \CustomNewAcronymDef
7922 }%
Set the display
7923 \@for\@gls@type:=\@glsacronymlists\do{%
7924 \SetCustomDisplayStyle{\@gls@type}%
7925 }%
7926 }

```

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.

```
7927 \RequirePackage{glossary-hypermnav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
7928 \@gls@loadlist
```

The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
7929 \@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.

```
7930 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7931 \@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.

```

7932 \ifx\@glossary@default@style\relax
7933 \else
7934 \setglossarystyle{\@glossary@default@style}
7935 \fi

```

1.20 Debugging Commands

\showgloparent \showgloparent{<label>}

```

7936 \newcommand*{\showgloparent}[1]{%
7937 \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
7938 }

```

`\showglolevel` `\showglolevel{<label>}`

```
7939 \newcommand*{\showglolevel}[1]{%
7940   \expandafter\show\csname glo@glstetoklabel{#1}@level\endcsname
7941 }
```

`\showglotext` `\showglotext{<label>}`

```
7942 \newcommand*{\showglotext}[1]{%
7943   \expandafter\show\csname glo@glstetoklabel{#1}@text\endcsname
7944 }
```

`\showgloplural` `\showgloplural{<label>}`

```
7945 \newcommand*{\showgloplural}[1]{%
7946   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7947 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7948 \newcommand*{\showglofirst}[1]{%
7949   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7950 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7951 \newcommand*{\showglofirstpl}[1]{%
7952   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7953 }
```

`\showglotype` `\showglotype{<label>}`

```
7954 \newcommand*{\showglotype}[1]{%
7955   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7956 }
```

`\showglocounter` `\showglocounter{<label>}`

```
7957 \newcommand*{\showglocounter}[1]{%
7958   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7959 }
```

`\showglouser` `\showglouser{<label>}`

```
7960 \newcommand*{\showglouser}[1]{%
7961   \expandafter\show\csname glo@glstetoklabel{#1}@user\endcsname
7962 }
```

`\showglouserii` `\showglouserii{<label>}`

```
7963 \newcommand*{\showglouserii}[1]{%
7964   \expandafter\show\csname glo@glstetoklabel{#1}@userii\endcsname
7965 }
```

`\showglouseriii` `\showglouseriii{<label>}`

```
7966 \newcommand*{\showglouseriii}[1]{%
7967   \expandafter\show\csname glo@glstetoklabel{#1}@useriii\endcsname
7968 }
```

`\showglouseriv` `\showglouseriv{<label>}`

```
7969 \newcommand*{\showglouseriv}[1]{%
7970   \expandafter\show\csname glo@glstetoklabel{#1}@useriv\endcsname
7971 }
```

`\showglouserv` `\showglouserv{<label>}`

```
7972 \newcommand*{\showglouserv}[1]{%
7973   \expandafter\show\csname glo@glstetoklabel{#1}@userv\endcsname
7974 }
```

\showglouservi \showglouservi{<label>}

```
7975 \newcommand*{\showglouservi}[1]{%
7976   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7977 }
```

\showgloname \showgloname{<label>}

```
7978 \newcommand*{\showgloname}[1]{%
7979   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7980 }
```

\showglodesc \showglodesc{<label>}

```
7981 \newcommand*{\showglodesc}[1]{%
7982   \expandafter\show\csname glo@glstdetoklabel{#1}@desc\endcsname
7983 }
```

howglodescplural \showglodescplural{<label>}

```
7984 \newcommand*{\showglodescplural}[1]{%
7985   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7986 }
```

\showglosort \showglosort{<label>}

```
7987 \newcommand*{\showglosort}[1]{%
7988   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7989 }
```

\showglosymbol \showglosymbol{<label>}

```
7990 \newcommand*{\showglosymbol}[1]{%
7991   \expandafter\show\csname glo@glstdetoklabel{#1}@symbol\endcsname
7992 }
```

wglosymbolplural `\showglosymbolplural{<label>}`

```
7993 \newcommand*{\showglosymbolplural}[1]{%
7994   \expandafter\show\csname glo@glstdetoklabel{#1}@symbolplural\endcsname
7995 }
```

\showgloshort `\showgloshort{<label>}`

```
7996 \newcommand*{\showgloshort}[1]{%
7997   \expandafter\show\csname glo@glstdetoklabel{#1}@short\endcsname
7998 }
```

\showglolong `\showglolong{<label>}`

```
7999 \newcommand*{\showglolong}[1]{%
8000   \expandafter\show\csname glo@glstdetoklabel{#1}@long\endcsname
8001 }
```

\showgloindex `\showgloindex{<label>}`

```
8002 \newcommand*{\showgloindex}[1]{%
8003   \expandafter\show\csname glo@glstdetoklabel{#1}@index\endcsname
8004 }
```

\showgloflag `\showgloflag{<label>}`

```
8005 \newcommand*{\showgloflag}[1]{%
8006   \expandafter\show\csname ifglo@glstdetoklabel{#1}@flag\endcsname
8007 }
```

\showgloloclist `\showgloloclist{<label>}`

```
8008 \newcommand*{\showgloloclist}[1]{%
8009   \expandafter\show\csname glo@glstdetoklabel{#1}@loclist\endcsname
8010 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
8011 \newcommand*{\showglofield}[2]{%
8012   \csshow{glo@glstetoklabel{#1}@#2}%
8013 }
```

`showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
8014 \newcommand*{\showacronymlists}{%
8015   \show\@glsacronymlists
8016 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
8017 \newcommand*{\showglossaries}{%
8018   \show\@glo@types
8019 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
8020 \newcommand*{\showglossaryin}[1]{%
8021   \expandafter\show\csname @glotype@#1@in\endcsname
8022 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
8023 \newcommand*{\showglossaryout}[1]{%
8024   \expandafter\show\csname @glotype@#1@out\endcsname
8025 }
```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
8026 \newcommand*{\showglossarytitle}[1]{%
8027   \expandafter\show\csname @glotype@#1@title\endcsname
8028 }
```

wglossarycounter `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
8029 \newcommand*{\showglossarycounter}[1]{%
8030   \expandafter\show\csname @glotype@#1@counter\endcsname
8031 }
```

wglossaryentries `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
8032 \newcommand*{\showglossaryentries}[1]{%
8033   \expandafter\show\csname glolist@#1\endcsname
8034 }
```

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.

```
8035 \csname ifglscpatible-2.07\endcsname
8036   \RequirePackage{glossaries-compatible-207}
8037 \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.

```
8038 \NeedsTeXFormat{LaTeX2e}
```

```
8039 \ProvidesPackage{glossaries-prefix}[2019/12/06 v4.44 (NLCT)]
```

Pass all options to glossaries:

```
8040 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
8041 \ProcessOptions
```

Load glossaries:

```
8042 \RequirePackage{glossaries}
```

Add the new keys:

```
8043 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
8044 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
8045 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
8046 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\gls@keymap`:

```
8047 \appto\@gls@keymap{,%
```

```
8048   {prefixfirst}{prefixfirst},%
```

```
8049   {prefixfirstplural}{prefixfirstplural},%
```

```
8050   {prefix}{prefix},%
```

```
8051   {prefixplural}{prefixplural}}%
```

```
8052 }
```

Set the default values:

```
8053 \appto\@newglossaryentryprehook{%
```

```
8054   \def\@glo@entryprefix{}}%
```

```
8055   \def\@glo@entryprefixplural{}}%
```

```
8056   \let\@glo@entryprefixfirst\@gls@default@value
```

```
8057   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
8058 }
```

Set the assignment code:

```
8059 \appto\@newglossaryentryposthook{%
```

```
8060   \gls@assign@field{}{\@glo@label}{prefix}{\@glo@entryprefix}}%
```

```
8061   \gls@assign@field{}{\@glo@label}{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
8062 \expandafter\gls@assign@field\expandafter
```

```
8063   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
8064   {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```
8065 \expandafter\gls@assign@field\expandafter
8066 {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
8067 {prefixfirstplural}{\@glo@entryprefixfirstplural}%
8068 }
```

Define commands to access these fields:

entryprefixfirst

```
8069 \newcommand*\glsentryprefixfirst[1]{\csuse{glo@#1@prefixfirst}}
```

entryfirstplural

```
8070 \newcommand*\glsentryprefixfirstplural[1]{\csuse{glo@#1@prefixfirstplural}}
```

\glsentryprefix

```
8071 \newcommand*\glsentryprefix[1]{\csuse{glo@#1@prefix}}
```

entryprefixplural

```
8072 \newcommand*\glsentryprefixplural[1]{\csuse{glo@#1@prefixplural}}
```

Now for the initial upper case variants:

entryprefixfirst

```
8073 \newrobustcmd*\Glsentryprefixfirst[1]{%
8074 \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
8075 \xmakefirstuc\@glo@text
8076 }
```

entryfirstplural

```
8077 \newrobustcmd*\Glsentryprefixfirstplural[1]{%
8078 \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
8079 \xmakefirstuc\@glo@text
8080 }
```

\Glsentryprefix

```
8081 \newrobustcmd*\Glsentryprefix[1]{%
8082 \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
8083 \xmakefirstuc\@glo@text
8084 }
```

entryprefixplural

```
8085 \newrobustcmd*\Glsentryprefixplural[1]{%
8086 \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
8087 \xmakefirstuc\@glo@text
8088 }
```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
8089 \newcommand*{\ifglshasprefix}[3]{%
8090   \ifcempty{glo@#1@prefix}%
8091   {#3}%
8092   {#2}%
8093 }
```

hasprefixplural

```
8094 \newcommand*{\ifglshasprefixplural}[3]{%
8095   \ifcempty{glo@#1@prefixplural}%
8096   {#3}%
8097   {#2}%
8098 }
```

shasprefixfirst

```
8099 \newcommand*{\ifglshasprefixfirst}[3]{%
8100   \ifcempty{glo@#1@prefixfirst}%
8101   {#3}%
8102   {#2}%
8103 }
```

efixfirstplural

```
8104 \newcommand*{\ifglshasprefixfirstplural}[3]{%
8105   \ifcempty{glo@#1@prefixfirstplural}%
8106   {#3}%
8107   {#2}%
8108 }
```

fix@record@hook Need to take into account the possibility that glossaries-extra might be loaded with the record option.

```
8109 \providecommand{\@glsprefix@record@hook}[2]{%
8110   \ifdef\@glsextr@record
8111   {\@glsextr@record{#1}{#2}{glslink}}%
8112   {}%
8113 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
8114 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
8115 \newcommand*{\@pgls}[2][ ]{%
8116   \new@ifnextchar[%
8117   {\@pgls@{#1}{#2}}%
8118   {\@pgls@{#1}{#2}[ ]}%
8119 }
```

\@pgls@ Read in the final optional argument:

```
8120 \def\@pgls@#1#2[#3]{%
8121   \@glsprefix@record@hook{#1}{#2}%
8122   \glsdoifexists{#2}%
8123   {%
8124     \ifglsused{#2}%
8125     {%
8126       \glsentryprefix{#2}%
8127     }%
8128     {%
8129       \glsentryprefixfirst{#2}%
8130     }%
8131     \@gls@{#1}{#2}[#3]%
8132   }%
8133 }
```

Similarly for the plural version:

\pglsp1

```
8134 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}
```

\@pglsp1 Unstarred version.

```
8135 \newcommand*{\@pglsp1}[2][ ]{%
8136   \new@ifnextchar[%
8137   {\@pglsp1@{#1}{#2}}%
8138   {\@pglsp1@{#1}{#2}[ ]}%
8139 }
```

\@pglsp1@ Read in the final optional argument:

```
8140 \def\@pglsp1@#1#2[#3]{%
8141   \@glsprefix@record@hook{#1}{#2}%
8142   \glsdoifexists{#2}%
8143   {%
8144     \ifglsused{#2}%
8145     {%
8146       \glsentryprefixplural{#2}%
8147     }%
8148     {%
8149       \glsentryprefixfirstplural{#2}%
8150     }%
8151     \@glspl@{#1}{#2}[#3]%
8152   }%
8153 }
```

Now for the first letter upper case versions:

\Pgls

```
8154 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}
```

\@PglS Unstarred version.

```
8155 \newcommand*{\@PglS}[2] [] {%
8156   \new@ifnextchar[%
8157     {\@PglS@{#1}{#2}}%
8158     {\@PglS@{#1}{#2} []}%
8159 }
```

\@PglS@ Read in the final optional argument:

```
8160 \def\@PglS@#1#2[#3] {%
8161   \@glsprefix@record@hook{#1}{#2}%
8162   \glsoifexists{#2}%
8163   {%
8164     \ifglSused{#2}%
8165     {%
8166       \ifglshasprefix{#2}%
8167       {%
8168         \GlSentryprefix{#2}%
8169         \@glS@{#1}{#2}[#3]%
8170       }%
8171       {\@GlS@{#1}{#2}[#3]}%
8172     }%
8173     {%
8174       \ifglshasprefixfirst{#2}%
8175       {%
8176         \GlSentryprefixfirst{#2}%
8177         \@glS@{#1}{#2}[#3]%
8178       }%
8179       {\@GlS@{#1}{#2}[#3]}%
8180     }%
8181   }%
8182 }
```

Similarly for the plural version:

\PglSp1

```
8183 \newrobustcmd{\PglSp1}{\@glS@hyp@opt\@PglSp1}
```

\@PglSp1 Unstarred version.

```
8184 \newcommand*{\@PglSp1}[2] [] {%
8185   \new@ifnextchar[%
8186     {\@PglSp1@{#1}{#2}}%
8187     {\@PglSp1@{#1}{#2} []}%
8188 }
```

\@PglSp1@ Read in the final optional argument:

```
8189 \def\@PglSp1@#1#2[#3] {%
8190   \@glsprefix@record@hook{#1}{#2}%
8191   \glsoifexists{#2}%

```

```

8192  {%
8193    \ifglused{#2}%
8194  {%
8195    \ifglshasprefixplural{#2}%
8196    {%
8197      \Glsentryprefixplural{#2}%
8198      \@glspl@{#1}{#2}[#3]%
8199    }%
8200    {\@Glspl@{#1}{#2}[#3]}%
8201  }%
8202  {%
8203    \ifglshasprefixfirstplural{#2}%
8204    {%
8205      \Glsentryprefixfirstplural{#2}%
8206      \@glspl@{#1}{#2}[#3]%
8207    }%
8208    {\@Glspl@{#1}{#2}[#3]}%
8209  }%
8210 }%
8211 }

```

Finally the all upper case versions:

\PGLS

```

8212 \newrobustcmd{\PGLS}{\@glshyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

8213 \newcommand*{\@PGLS}[2][ ]{%
8214   \new@ifnextchar[%
8215     {\@PGLS@{#1}{#2}}%
8216     {\@PGLS@{#1}{#2}[ ]}%
8217 }

```

\@PGLS@ Read in the final optional argument:

```

8218 \def\@PGLS@#1#2[#3]{%
8219   \@glsprefix@record@hook{#1}{#2}%
8220   \glsoifexists{#2}%
8221   {%
8222     \ifglused{#2}%
8223     {%
8224       \mfirstucMakeUppercase{\glentryprefix{#2}}%
8225     }%
8226     {%
8227       \mfirstucMakeUppercase{\glentryprefixfirst{#2}}%
8228     }%
8229     \@GLS@{#1}{#2}[#3]%
8230   }%
8231 }

```

Plural version:

\PGLSp1

```
8232 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}
```

\@PGLSp1 Unstarred version.

```
8233 \newcommand*{\@PGLSp1}[2] [] {%
8234   \new@ifnextchar[%
8235     {\@PGLSp1@{#1}{#2}}%
8236     {\@PGLSp1@{#1}{#2} []}%
8237 }
```

\@PGLSp1@ Read in the final optional argument:

```
8238 \def\@PGLSp1@#1#2[#3] {%
8239   \@glsprefix@record@hook{#1}{#2}%
8240   \glsdoifexists{#2}%
8241   {%
8242     \ifglsused{#2}%
8243     {%
8244       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
8245     }%
8246     {%
8247       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
8248     }%
8249     \@GLSp1@{#1}{#2}[#3]%
8250   }%
8251 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8252 \ProvidesPackage{glossary-hypernav}[2019/12/06 v4.44 (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`

```
8253 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8254   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8255   \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

`glsnavhyperlinkname`

Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
8256 \newcommand*{\glsnavhyperlinkname}[2]{\glsn:#1@#2}
```

```
\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.

`glsnavhypertarget`

```
8257 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
8258   \@glsnavhypertarget{#1}{#2}{#3}%
8259 }
```

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.

`glsnavhypertarget`

```
8260 \newcommand*{\@glsnavhypertarget}[3]{%}
```


Add this group to the aux file for re-run check.

```
8261 \protected@write\auxout{}\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
8262 \@gls@target{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8263 \expandafter\let
```

```
8264 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8265 \@for\@gls@elem:=\@gls@list\do{%
```

```
8266 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
8267 \if@endfor
```

```
8268 \else
```

This group was not included in the list, so issue a warning.

```
8269 \GlossariesWarningNoLine{Navigation panel
```

```
8270 for glossary type ‘#1’~Jmissing group ‘#2’}%
```

```
8271 \gdef\gls@hypergroup@rerun{%
```

```
8272 \GlossariesWarningNoLine{Navigation panel
```

```
8273 has changed. Rerun LaTeX}}%
```

```
8274 \fi
```

```
8275 }
```

`\hypergroup@rerun` Give a warning at the end if re-run required

```
8276 \let\gls@hypergroup@rerun\relax
```

```
8277 \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.

```
8278 \newcommand*{\@gls@hypergroup}[2]{%
```

```
8279 \@ifundefined{\@gls@hypergroup@list@#1}{%
```

```
8280 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}}%
```

```
8281 }{%
```

```
8282 \expandafter\let\expandafter\@gls@tmp
```

```
8283 \csname @gls@hypergroup@list@#1\endcsname
```

```
8284 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
```

```
8285 \@gls@tmp,#2}}%
```

```
8286 }%
```

```
8287 }
```

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`

```
8288 \newcommand*{\glsnavigation}{%
8289   \def\@gls@between{}%
8290   \ifcsundef{\@gls@hypergroupplist@\@glo@type}%
8291   {%
8292     \def\@gls@list{}%
8293   }%
8294   {%
8295     \expandafter\let\expandafter\@gls@list
8296       \csname \@gls@hypergroupplist@\@glo@type\endcsname
8297   }%
8298   \@for\@gls@tmp:=\@gls@list\do{%
8299     \@gls@between

8300     \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8301     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8302     \let\@gls@between\glshypernavsep
8303   }%
8304 }
```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8305 \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`

```
8306 \newcommand*{\glssymbolnav}{%
8307   \glsnavhyperlink{glssymbols}{\@gls@getgrouptitle{glssymbols}}%
8308   \glshypernavsep
8309   \glsnavhyperlink{glsnumbers}{\@gls@getgrouptitle{glsnumbers}}%
8310   \glshypernavsep
8311 }
```

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.

```
8312 \ProvidesPackage{glossary-inline}[2019/12/06 v4.44 (NLCT)]
```

`inline` Define the inline style.

```
8313 \newglossarystyle{inline}{%
      Start of glossary sets up first empty separator between entries. (This is then changed by
      \glossentry)
8314   \renewenvironment{theglossary}%
8315   {%
```

```

8316      \def\gls@inlinesep{}%
8317      \def\gls@inlinesubsep{}%
8318      \def\gls@inlinepostchild{}%
8319      }%
8320      {\glspostinline}%

```

No header:

```

8321 \renewcommand*{\glossaryheader}{}%

```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```

8322 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

8323 \renewcommand{\glossentry}[2]{%
8324   \glsinlinedopostchild
8325   \gls@inlinesep
8326   \glsentryitem{##1}%
8327   \glsinlinenameformat{##1}%
8328   \glossentryname{##1}%
8329   }%
8330 \ifglstdescsuppressed{##1}%
8331 {%
8332   \glsinlineemptydescformat
8333   {%
8334     \glossentrysymbol{##1}%
8335   }%
8336   {%
8337     ##2%
8338   }%
8339 }%
8340 {%
8341   \ifglshasdesc{##1}%
8342   {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8343   {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8344 }%
8345 \ifglshaschildren{##1}%
8346 {%
8347   \glsresetsubentrycounter
8348   \glsinlineparentchildseparator
8349   \def\gls@inlinesubsep{}%
8350   \def\gls@inlinepostchild{\glsinlinepostchild}%
8351 }%
8352 {}%
8353 \def\gls@inlinesep{\glsinlineseparator}%
8354 }%

```

Sub-entries display description:

```

8355 \renewcommand{\subglossentry}[3]{%
8356   \gls@inlinesubsep%
8357   \glsinlinesubnameformat{##2}%

```

```

8358     \glossentryname{##2}}}%
8359     \glssubentryitem{##2}%
8360     \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8361     \def\gls@inlinesubsep{\glsinlinesubseparator}%
8362 }%

```

Nothing special between groups:

```

8363 \renewcommand*{\glsgroupskip}{}%
8364 }

```

linedopostchild

```

8365 \newcommand*{\glsinlinedopostchild}{%
8366     \gls@inlinepostchild
8367     \def\gls@inlinepostchild{}}%
8368 }

```

inlineseparator Separator to use between entries.

```

8369 \newcommand*{\glsinlineseparator}{;\space}

```

inlinesubseparator Separator to use between sub-entries.

```

8370 \newcommand*{\glsinlinesubseparator}{,\space}

```

parentchildseparator Separator to use between parent and children.

```

8371 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

inlinepostchild Hook to use between child and next entry

```

8372 \newcommand*{\glsinlinepostchild}{ }

```

\glspostinline Terminator for inline glossary.

```

8373 \newcommand*{\glspostinline}{\glspostdescription\space}

```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```

8374 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

inlinedescformat Formats the entry's description, symbol and location list:

```

8375 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```

8376 \newcommand*{\glsinlineemptydescformat}[2]{ }

```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```

8377 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{ }}

```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```

8378 \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.

8379 \ProvidesPackage{glossary-list}[2019/12/06 v4.44 (NLCT)]

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8380 \providecommand{\indexspace}{%
8381   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8382 }
```

`tgrouphaderfmt` Provide a way of adjusting the format of the group headings.

```
8383 \newcommand*{\glslistgroupheaderfmt}[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.

```
8384 \newcommand*{\glslistnavigationitem}[1]{\item[#1]}
```

`list` The list glossary style uses the description environment. The group separator `\glsgroupskip` 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.

```
8385 \newglossarystyle{list}{%
```

Use description environment:

```
8386 \renewenvironment{theglossary}%
8387   {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8388 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8389 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8390 \renewcommand*{\glossentry}[2]{%
8391   \item[\glssentryitem{##1}%
8392     \glstarget{##1}{\glossentryname{##1}}]
8393   \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
8394 \renewcommand*{\subglossentry}[3]{%
8395   \glssubentryitem{##2}%
```

```

8396 \glstarget{##2}{\strut}\space
8397 \glossentrydesc{##2}\glspostdescription\space ##3.}%

Add vertical space between groups:

8398 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8399 }

```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```

8400 \newglossarystyle{listgroup}{%
    Base it on the list style:
8401 \setglossarystyle{list}%

    Each group has a heading:
8402 \renewcommand*{\glsgroupheading}[1]{%
8403 \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.

```

8404 \newglossarystyle{listhypergroup}{%
    Base it on the list style:
8405 \setglossarystyle{list}%

    Add navigation links at the start of the environment.
8406 \renewcommand*{\glossaryheader}{%
8407 \glslistnavigationitem{\glslnavigation}}%

    Each group has a heading with a hypertarget:
8408 \renewcommand*{\glsgroupheading}[1]{%
8409 \item[\glslistgroupheaderfmt
8410 {\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.

```

8411 \newglossarystyle{altlist}{%
    Base it on the list style:
8412 \setglossarystyle{list}%

    Main (level 0) entries start a new item in the list with a line break after the entry name:
8413 \renewcommand*{\glossentry}[2]{%
8414 \item[\glsentryitem{##1}%
8415 \glstarget{##1}{\glossentryname{##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.

```

8416 \mbox{}\par\nobreak\@afterheading
8417 \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries start a new paragraph:

```
8418 \renewcommand{\subglossentry}[3]{%
8419   \par
8420   \glssubentryitem{##2}%
8421   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
8422 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8423 \newglossarystyle{altlistgroup}{%
      Base it on the altlist style:
8424   \setglossarystyle{altlist}%
      Each group has a heading:
8425   \renewcommand*{\glsgroupeheading}[1]{%
8426     \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.

```
8427 \newglossarystyle{altlisthypergroup}{%
      Base it on the altlist style:
8428   \setglossarystyle{altlist}%
      Add navigation links at the start of the environment.
8429   \renewcommand*{\glossaryheader}{%
8430     \glslistnavigationitem{\glslistnavigation}}%
      Each group has a heading with a hypertarget:
8431   \renewcommand*{\glsgroupeheading}[1]{%
8432     \item[\glslistgroupheaderfmt
8433       {\glslistnavigationhypertarget{##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.

```
8434 \newglossarystyle{listdotted}{%
      Base it on the list style:
8435   \setglossarystyle{list}%
      Each main (level 0) entry starts a new item:
8436   \renewcommand*{\glossentry}[2]{%
8437     \item[\makebox[\glslistdottedwidth][l]{%
8438       \glssentryitem{##1}%
8439       \glstarget{##1}{\glossentryname{##1}}}%
8440     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```

8441 \renewcommand*{\subglossentry}[3]{%
8442   \item[\makebox[\glslistdottedwidth][l]{%
8443     \glssubentryitem{##2}%
8444     \glstarget{##2}{\glossentryname{##2}}}%
8445   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8446 }

```

`listdottedwidth`

```

8447 \newlength\glslistdottedwidth
8448 \setlength{\glslistdottedwidth}{.5\hsize}

```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8449 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
8450 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```

8451 \renewcommand*{\glossentry}[2]{%
8452   \item[\glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
8453 }

```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8454 \ProvidesPackage{glossary-long}[2019/12/06 v4.44 (NLCT)]
```

Requires the package:

```
8455 \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`.)

```

8456 \@ifundefined{glsdescwidth}{%
8457   \newlength\glsdescwidth
8458   \setlength{\glsdescwidth}{0.6\hsize}
8459 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column.

```

8460 \@ifundefined{glspagelistwidth}{%
8461   \newlength\glspagelistwidth
8462   \setlength{\glspagelistwidth}{0.1\hsize}
8463 }{}

```


long The long glossary style command which uses the longtable environment:

```
8464 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
8465 \renewenvironment{theglossary}{%
8466     {\begin{longtable}\lp{\glstdescwidth}}}%
8467     {\end{longtable}}}%
```

Do nothing at the start of the environment:

```
8468 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8469 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8470 \renewcommand{\glossentry}[2]{%
8471     \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8472     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8473 }%
```

Sub entries displayed on the following row without the name:

```
8474 \renewcommand{\subglossentry}[3]{%
8475     &
8476     \glssubentryitem{##2}%
8477     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8478     ##3\tabularnewline
8479 }%
```

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>)

```
8480 \ifglsnogroupskip
8481 \renewcommand*{\glsgroupskip}{}%
8482 \else
8483 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8484 \fi
8485 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
8486 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
8487 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
8488 \renewenvironment{theglossary}{%
8489     \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8490 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8491 }
```

longheader The longheader style is like the long style but with a header:

```
8492 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8493 \setglossarystyle{long}{%
```

Set the table's header:

```
8494 \renewcommand*{\glossaryheader}{%
8495 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8496 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8497 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8498 \setglossarystyle{longborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8499 \renewcommand*{\glossaryheader}{%
8500 \hline\bfseries \entryname & \bfseries
8501 \descriptionname\tabularnewline\hline
8502 \endhead
8503 \hline\endfoot}%
8504 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
8505 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
8506 \renewenvironment{theglossary}%
8507 {\begin{longtable}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8508 {\end{longtable}}%
```

No table header:

```
8509 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8510 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8511 \renewcommand{\glossentry}[2]{%
8512 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8513 \glossentrydesc{##1} & ##2\tabularnewline
8514 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8515 \renewcommand{\subglossentry}[3]{%
8516 &
8517 \glssubentryitem{##2}%
8518 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8519 ##3\tabularnewline
8520 }%
```

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>)

```
8521 \ifglsgroupskip
8522 \renewcommand*{\glsgroupskip}{}%
8523 \else
8524 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8525 \fi
8526 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
8527 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8528 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8529 \renewenvironment{theglossary}%
8530 {\begin{longtable}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8531 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8532 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8533 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
8534 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
8535 \setglossarystyle{long3col}%
    Set the table's header:
8536 \renewcommand*{\glossaryheader}{%
8537 \bfseries\entryname&\bfseries\descriptionname&
8538 \bfseries\pagelistname\tabularnewline\endhead}%
8539 }
```

colheaderborder The long3colheaderborder style is like the above but with a border

```
8540 \newglossarystyle{long3colheaderborder}{%
    Base it on the glostylelong3colborder style:
8541 \setglossarystyle{long3colborder}%
    Set the table's header and add horizontal line at table's foot:
8542 \renewcommand*{\glossaryheader}{%
8543 \hline
8544 \bfseries\entryname&\bfseries\descriptionname&
8545 \bfseries\pagelistname\tabularnewline\hline\endhead
8546 \hline\endfoot}%
8547 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8548 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8549 \renewenvironment{theglossary}{%
```

```
8550 {\begin{longtable}{llll}}%
```

```
8551 {\end{longtable}}%
```

No table header:

```
8552 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8553 \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):

```
8554 \renewcommand{\glossentry}[2]{%
```

```
8555 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
8556 \glossentrydesc{##1} &
```

```
8557 \glossentrysymbol{##1} &
```

```
8558 ##2\tabularnewline
```

```
8559 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8560 \renewcommand{\subglossentry}[3]{%
```

```
8561 &
```

```
8562 \glssubentryitem{##2}%
```

```
8563 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
8564 \glossentrysymbol{##2} & ##3\tabularnewline
```

```
8565 }%
```

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>)

```
8566 \ifglsgnogroupskip
```

```
8567 \renewcommand*{\glsgroupskip}{}%
```

```
8568 \else
```

```
8569 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
```

```
8570 \fi
```

```
8571 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8572 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8573 \setglossarystyle{long4col}%
```

Table has a header:

```
8574 \renewcommand*{\glossaryheader}{%
```

```
8575 \bfseries\entryname&\bfseries\descriptionname&
```

```
8576 \bfseries \symbolname&
```

```

8577 \bfseries\pagelistname\tabularnewline\endhead}%
8578 }

```

long4colborder The long4colborder style is like long4col but with a border.

```

8579 \newglossarystyle{long4colborder}{%
    Base it on the glostylelong4col style:
8580 \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8581 \renewenvironment{theglossary}%
8582 {\begin{longtable}{|l|l|l|l|}}%
8583 {\end{longtable}}%
    Add horizontal lines to the head and foot of the table:
8584 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8585 }

```

colheaderborder The long4colheaderborder style is like the above but with a border.

```

8586 \newglossarystyle{long4colheaderborder}{%
    Base it on the glostylelong4col style:
8587 \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8588 \renewenvironment{theglossary}%
8589 {\begin{longtable}{|l|l|l|l|}}%
8590 {\end{longtable}}%
    Add table header and horizontal line at the table's foot:
8591 \renewcommand*{\glossaryheader}{%
8592 \hline\bfseries\entryname&\bfseries\descriptionname&
8593 \bfseries \symbolname&
8594 \bfseries\pagelistname\tabularnewline\hline\endhead
8595 \hline\endfoot}%
8596 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8597 \newglossarystyle{altlong4col}{%
    Base it on the glostylelong4col style:
8598 \setglossarystyle{long4col}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8599 \renewenvironment{theglossary}%
8600 {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8601 {\end{longtable}}%
8602 }

```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```

8603 \newglossarystyle{altlong4colheader}{%
      Base it on the glostylelong4colheader style:
8604   \setglossarystyle{long4colheader}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8605   \renewenvironment{theglossary}%
8606     {\begin{longtable}{\lp{\glsgdescwidth}\lp{\glspagelistwidth}}}%
8607     {\end{longtable}}%
8608 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```

8609 \newglossarystyle{altlong4colborder}{%
      Base it on the glostylelong4colborder style:
8610   \setglossarystyle{long4colborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8611   \renewenvironment{theglossary}%
8612     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8613     {\end{longtable}}%
8614 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```

8615 \newglossarystyle{altlong4colheaderborder}{%
      Base it on the glostylelong4colheaderborder style:
8616   \setglossarystyle{long4colheaderborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8617   \renewenvironment{theglossary}%
8618     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8619     {\end{longtable}}%
8620 }
```

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>

```

8621 \ProvidesPackage{glossary-longbooktabs}[2019/12/06 v4.44 (NLCT)]
```

Requires `booktabs` package:

```

8622 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8623 \RequirePackage{glossary-long}
8624 \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.

```
8625 \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.

```
8626 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8627 \setglossarystyle{long}%
```

Add a header with rules.

```
8628 \renewcommand*{\glossaryheader}{%
8629 \toprule \bfseries \entryname & \bfseries
8630 \descriptionname\tabularnewline\midrule\endhead
8631 \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.

```
8632 \ifglsgnogroupskip
8633 \renewcommand*{\glsgroupskip}{}%
8634 \else
8635 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8636 \fi
8637 }
```

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.

```
8638 \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.

```
8639 \glspatchLToutput
```

Use the long3col style as a base.

```
8640 \setglossarystyle{long3col}%
```

Add a header with rules.

```
8641 \renewcommand*{\glossaryheader}{%
8642 \toprule \bfseries \entryname &
8643 \bfseries \descriptionname &
8644 \bfseries \pagelistname
8645 \tabularnewline\midrule\endhead
8646 \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.

```
8647 \ifglsgroupskip
8648 \renewcommand*{\glsgroupskip}{}%
8649 \else
8650 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8651 \fi
8652 }
```

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.

```
8653 \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.

```
8654 \glspatchLToutput
```

Use the long4col style as a base.

```
8655 \setglossarystyle{long4col}{%
```

Add a header with rules.

```
8656 \renewcommand*{\glossaryheader}{%
8657 \toprule \bfseries \entryname &
8658 \bfseries \descriptionname &
8659 \bfseries \symbolname &
8660 \bfseries \pagelistname
8661 \tabularnewline\midrule\endhead
8662 \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.

```
8663 \ifglsgroupskip
8664 \renewcommand*{\glsgroupskip}{}%
8665 \else
8666 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8667 \fi
8668 }
```

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.

```
8669 \newglossarystyle{altlong4col-booktabs}{%
```

The patch \glspatchLToutput is already applied in long4col-booktabs and so doesn't need to be here.

```
8670 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8671 \setglossarystyle{long4col-booktabs}{%
```


Change the column specifications:

```
8672 \renewenvironment{theglossary}%  
8673   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%  
8674   {\end{longtable}}}%  
8675 }
```

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.

```
8676 \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.

```
8677 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8678 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8679 \renewenvironment{theglossary}%  
8680   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%  
8681   {\end{longtable}}}%  
8682 }
```

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.

```
8683 \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.

```
8684 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8685 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8686 \renewenvironment{theglossary}%  
8687   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%  
8688   >{\raggedright}p{\glspagelistwidth}}}%  
8689   {\end{longtable}}}%  
8690 }
```

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.

```
8691 \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.

```
8692 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8693 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8694 \renewenvironment{theglossary}%
8695   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8696     >{\raggedright}p{\glspagelistwidth}}}%
8697   {\end{longtable}}%
8698 }
```

sLTpenaltycheck

```
8699 \newcommand*{\glslTpenaltycheck}{%
8700   \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi
8701 }
```

enaltygroupskip

```
8702 \newcommand{\glspenaltygroupskip}{%
8703   \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8704 \let\@gls@org@LT@output\LT@output
8705 \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

```
8706 \newcommand*{\glspatchLToutput}{%
8707   \renewcommand*{\LT@output}{%
8708     \ifnum\outputpenalty < -\@Mi
8709       \ifnum\outputpenalty > -\LT@end@pen
8710         \LT@err{floats and marginpars not allowed in a longtable}\@ehc
8711       \else
8712         \setbox\z@\vbox{\unvbox\@cclv}%
8713         \ifdim \ht\LT@lastfoot>\ht\LT@foot
8714           \dimen@pagegoal
8715           \advance\dimen@-\ht\LT@lastfoot
8716           \ifdim\dimen@<\ht\z@
8717             \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
8718             \@makecol
8719             \@outputpage
8720             \setbox\z@\vbox{\box\LT@head\glslTpenaltycheck}%
8721           \fi
8722         \fi
8723         \global\@colroom\@colht
8724         \global\vsize\@colht
8725         {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8726       \fi
8727     \else
```

```

8728 \setbox\@cclv\vbox{\unvbox\@ccclv\copy\LT@foot\vss}%
8729 \@makecol
8730 \@outputpage
8731 \global\ysize\@colroom
8732 \copy\LT@head
8733 \glsLTpenaltycheck
8734 \nobreak
8735 \fi
8736 }%
8737 }

```

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.

```
8738 \ProvidesPackage{glossary-longragged}[2019/12/06 v4.44 (NLCT)]
```

Requires the package:

```
8739 \RequirePackage{array}
```

Requires the package:

```
8740 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8741 \@ifundefined{glsdescwidth}{%
8742 \newlength\glsdescwidth
8743 \setlength{\glsdescwidth}{0.6\hsize}
8744 }{}

```

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8745 \@ifundefined{glspagelistwidth}{%
8746 \newlength\glspagelistwidth
8747 \setlength{\glspagelistwidth}{0.1\hsize}
8748 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8749 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8750 \renewenvironment{theglossary}%
8751 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
8752 {\end{longtable}}%

```

Do nothing at the start of the environment:

```
8753 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8754 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8755 \renewcommand{\glossentry}[2]{%
8756   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8757   \glossentrydesc{##1}\glspostdescription\space ##2%
8758   \tabularnewline
8759 }%
```

Sub entries displayed on the following row without the name:

```
8760 \renewcommand{\subglossentry}[3]{%
8761   &
8762   \glssubentryitem{##2}%
8763   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8764   \glspostdescription\space ##3%
8765   \tabularnewline
8766 }%
```

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>)

```
8767 \ifglsgroupskip
8768 \renewcommand*{\glsgroupskip}{}%
8769 \else
8770 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8771 \fi
8772 }
```

`ongraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8773 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8774 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8775 \renewenvironment{theglossary}{%
8776   \begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|}%
8777   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8778 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8779 }
```

`ongraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8780 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8781 \setglossarystyle{longragged}%
```

Set the table's header:

```
8782 \renewcommand*{\glossaryheader}{%
8783   \bfseries \entryname & \bfseries \descriptionname
```

```

8784 \tabularnewline\endhead}%
8785 }

```

gedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```

8786 \newglossarystyle{longraggedheaderborder}{%

```

Base it on the glostylelongraggedborder style:

```

8787 \setglossarystyle{longraggedborder}%

```

Set the table's header and add horizontal line to table's foot:

```

8788 \renewcommand*{\glossaryheader}{%
8789 \hline\bfseries \entryname & \bfseries \descriptionname
8790 \tabularnewline\hline
8791 \endhead
8792 \hline\endfoot}%
8793 }

```

longragged3col The longragged3col style is like longragged but with 3 columns

```

8794 \newglossarystyle{longragged3col}{%

```

Use a longtable with 3 columns:

```

8795 \renewenvironment{theglossary}%
8796 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8797 >{\raggedright}p{\glspagelistwidth}}}%
8798 {\end{longtable}}%

```

No table header:

```

8799 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```

8800 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8801 \renewcommand{\glossentry}[2]{%
8802 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8803 \glossentrydesc{##1} & ##2\tabularnewline
8804 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8805 \renewcommand{\subglossentry}[3]{%
8806 &
8807 \glssubentryitem{##2}%
8808 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8809 ##3\tabularnewline
8810 }%

```

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>)

```

8811 \ifglsnogroupskip
8812 \renewcommand*{\glsgroupskip}{}%

```

```

8813 \else
8814 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8815 \fi
8816 }

```

ragged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```

8817 \newglossarystyle{longragged3colborder}{%
    Base it on the glostylelongragged3col style:
8818 \setglossarystyle{longragged3col}%
    Use a longtable with 3 columns with vertical lines around them:
8819 \renewenvironment{theglossary}%
8820 {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|}%
8821 >{\raggedright}p{\glspagelistwidth}|}%
8822 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8823 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8824 }

```

ragged3colheader The longragged3colheader style is like longragged3col but with a header row:

```

8825 \newglossarystyle{longragged3colheader}{%
    Base it on the glostylelongragged3col style:
8826 \setglossarystyle{longragged3col}%
    Set the table's header:
8827 \renewcommand*{\glossaryheader}{%
8828 \bfseries\entryname&\bfseries\descriptionname&
8829 \bfseries\pagelistname\tabularnewline\endhead}%
8830 }

```

colheaderborder The longragged3colheaderborder style is like the above but with a border

```

8831 \newglossarystyle{longragged3colheaderborder}{%
    Base it on the glostylelongragged3colborder style:
8832 \setglossarystyle{longragged3colborder}%
    Set the table's header and add horizontal line at table's foot:
8833 \renewcommand*{\glossaryheader}{%
8834 \hline
8835 \bfseries\entryname&\bfseries\descriptionname&
8836 \bfseries\pagelistname\tabularnewline\hline\endhead
8837 \hline\endfoot}%
8838 }

```

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.

```

8839 \newglossarystyle{altlongragged4col}{%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8840 \renewenvironment{theglossary}%
8841   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8842     >{\raggedright}p{\glspagelistwidth}}}%
8843   {\end{longtable}}%
```

No table header:

```
8844 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8845 \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):

```
8846 \renewcommand{\glossentry}[2]{%
8847   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8848   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8849   ##2\tabularnewline
8850 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8851 \renewcommand{\subglossentry}[3]{%
8852   &
8853   \glssubentryitem{##2}%
8854   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8855   \glossentrysymbol{##2} & ##3\tabularnewline
8856 }%
```

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>)

```
8857 \ifglsgroupskip
8858   \renewcommand*{\glsgroupskip}{}%
8859 \else
8860   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8861 \fi
8862 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8863 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8864 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8865 \renewenvironment{theglossary}%
8866   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8867     >{\raggedright}p{\glspagelistwidth}}}%
8868   {\end{longtable}}%
```

Table has a header:

```
8869 \renewcommand*{\glossaryheader}{%
8870 \bfseries\entryname&\bfseries\descriptionname&
8871 \bfseries \symbolname&
8872 \bfseries\pagelistname\tabularnewline\endhead}%
8873 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8874 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8875 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8876 \renewenvironment{theglossary}%
8877 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8878 >\raggedright}p{\glspagelistwidth}|}%
8879 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8880 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8881 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8882 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8883 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8884 \renewenvironment{theglossary}%
8885 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8886 >\raggedright}p{\glspagelistwidth}|}%
8887 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8888 \renewcommand*{\glossaryheader}{%
8889 \hline\bfseries\entryname&\bfseries\descriptionname&
8890 \bfseries \symbolname&
8891 \bfseries\pagelistname\tabularnewline\hline\endhead
8892 \hline\endfoot}%
8893 }
```

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.

```
8894 \ProvidesPackage{glossary-mcols}[2019/12/06 v4.44 (NLCT)]
```


Required packages:

```
8895 \RequirePackage{multicol}
8896 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8897 \providecommand{\indexspace}{%
8898   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8899 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8900 \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.)

```
8901 \newglossarystyle{mcolindex}{%
8902   \setglossarystyle{index}%
8903   \renewenvironment{theglossary}%
8904     {%
8905       \begin{multicols}{\glsmcols}
8906       \setlength{\parindent}{0pt}%
8907       \setlength{\parskip}{0pt plus 0.3pt}%
8908       \let\item\glstreeitem
8909       \let\subitem\glstreesubitem
8910       \let\subsubitem\glstreesubsubitem
8911     }%
8912     {\end{multicols}}}%
8913 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8914 \newglossarystyle{mcolindexgroup}{%
8915   \setglossarystyle{mcolindex}%
8916   \renewcommand*{\glsgroupheading}[1]{%
8917     \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\indexspace}%
8918 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8919 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8920   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8921   \renewcommand*{\glossaryheader}{%
8922     \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.

```
8923 \renewcommand*{\glsgroupheading}[1]{%
8924   \item\glstreegroupheaderfmt
8925     {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8926   \indexspace}%
8927 }
```

colindexspannav Similar to **colindexhypergroup**, but puts the navigation line in the optional argument of **multicols**.

```
8928 \newglossarystyle{colindexspannav}{%
8929   \setglossarystyle{index}%
8930   \renewenvironment{theglossary}%
8931     {%
8932       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8933       \setlength{\parindent}{0pt}%
8934       \setlength{\parskip}{0pt plus 0.3pt}%
8935       \let\item\glstreeitem}%
8936     {\end{multicols}}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8937 \renewcommand*{\glsgroupheading}[1]{%
8938   \item\glstreegroupheaderfmt
8939     {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8940   \indexspace}%
8941 }
```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8942 \newglossarystyle{mcoltree}{%
8943   \setglossarystyle{tree}%
8944   \renewenvironment{theglossary}%
8945     {%
8946       \begin{multicols}{\glsmcols}
8947       \setlength{\parindent}{0pt}%
8948       \setlength{\parskip}{0pt plus 0.3pt}%
8949     }%
8950     {\end{multicols}}%
8951 }
```

mcoltreegroup Like the **mcoltree** style but the glossary groups have headings.

```
8952 \newglossarystyle{mcoltreegroup}{%
  Base it on the glostylemcoltree style:
8953   \setglossarystyle{mcoltree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```
8954 \renewcommand{\glsgroupheading}[1]{\par
8955 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8956 }
```

ltreehypergroup The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8957 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the `glostylemcoltree` style:

```
8958 \setglossarystyle{mcoltree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8959 \renewcommand*{\glossaryheader}{%
8960 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8961 \renewcommand*{\glsgroupheading}[1]{%
8962 \par\noindent
8963 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8964 \indexspace}%
8965 }
```

mcoltreespannav Similar to the `mcoltreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8966 \newglossarystyle{mcoltreespannav}{%
8967 \setglossarystyle{tree}%
8968 \renewenvironment{theglossary}%
8969 {%
8970 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8971 \setlength{\parindent}{0pt}%
8972 \setlength{\parskip}{0pt plus 0.3pt}%
8973 }%
8974 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8975 \renewcommand*{\glsgroupheading}[1]{%
8976 \par\noindent
8977 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8978 \indexspace}%
8979 }
```

mcoltreenoname Multi-column index style. Same as the `treenoname`, but puts the glossary in multiple columns.

```
8980 \newglossarystyle{mcoltreenoname}{%
8981 \setglossarystyle{treenoname}%
8982 \renewenvironment{theglossary}%
8983 {%
```

```

8984     \begin{multicols}{\glsmcols}
8985     \setlength{\parindent}{0pt}%
8986     \setlength{\parskip}{0pt plus 0.3pt}%
8987 }%
8988 {\end{multicols}}%
8989 }

```

treenonamegroup Like the `mcoltreenoname` style but the glossary groups have headings.

```

8990 \newglossarystyle{mcoltreenonamegroup}{%
      Base it on the glostylemcoltreenoname style:
8991   \setglossarystyle{mcoltreenoname}%
      Give each group a heading:
8992   \renewcommand{\glsgroupheading}[1]{\par
8993     \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8994 }

```

onamehypergroup The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

8995 \newglossarystyle{mcoltreenonamehypergroup}{%
      Base it on the glostylemcoltreenoname style:
8996   \setglossarystyle{mcoltreenoname}%
      Put navigation links to the groups at the start of the theglossary environment:
8997   \renewcommand*{\glossaryheader}{%
8998     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8999   \renewcommand*{\glsgroupheading}[1]{%
9000     \par\noindent
9001     \glstreegroupheaderfmt{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
9002     \indexspace}%
9003 }

```

eenonamespannav Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

9004 \newglossarystyle{mcoltreenonamespannav}{%
9005   \setglossarystyle{treenoname}%
9006   \renewenvironment{theglossary}%
9007   {%
9008     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
9009     \setlength{\parindent}{0pt}%
9010     \setlength{\parskip}{0pt plus 0.3pt}%
9011   }%
9012   {\end{multicols}}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
9013   \renewcommand*{\glsgroupheading}[1]{%
9014     \par\noindent

```

```

9015     \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgrouptitle{##1}}}\par
9016     \indexspace}%
9017 }

```

mcolalttree Multi-column index style. Same as the alttree, but puts the glossary in multiple columns.

```

9018 \newglossarystyle{mcolalttree}{%
9019   \setglossarystyle{alttree}%
9020   \renewenvironment{theglossary}%
9021   {%
9022     \begin{multicols}{\glsmcols}
9023     \def\@gls@prevlevel{-1}%
9024     \mbox{}\par
9025   }%
9026   {\par\end{multicols}}}%
9027 }

```

colalttreegroup Like the mcolalttree style but the glossary groups have headings.

```

9028 \newglossarystyle{colalttreegroup}{%
    Base it on the glostylemcolalttree style:
9029   \setglossarystyle{mcolalttree}%
    Give each group a heading.
9030   \renewcommand{\glsgroupheading}[1]{\par
9031     \def\@gls@prevlevel{-1}%
9032     \hangindent0pt\relax
9033     \parindent0pt\relax
9034     \glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
9035 }

```

treehypergroup The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9036 \newglossarystyle{mcolalttreehypergroup}{%
    Base it on the glostylemcolalttree style:
9037   \setglossarystyle{mcolalttree}%
    Put the navigation links in the header
9038   \renewcommand*{\glossaryheader}{%
9039     \par
9040     \def\@gls@prevlevel{-1}%
9041     \hangindent0pt\relax
9042     \parindent0pt\relax
9043     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9044   \renewcommand*{\glsgroupheading}[1]{%
9045     \par
9046     \def\@gls@prevlevel{-1}%
9047     \hangindent0pt\relax

```

```

9048 \parindent0pt\relax
9049 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
9050 \indexspace}%
9051 }

```

`lalttreespannav` Similar to the `mcolalttreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

9052 \newglossarystyle{mcolalttreespannav}{%
9053 \setglossarystyle{alttree}%
9054 \renewenvironment{theglossary}%
9055 {%
9056 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
9057 \def\@gls@prevlevel{-1}%
9058 \mbox{}\par
9059 }%
9060 {\par\end{multicols}}}%

```

Put a hypertarget at the start of each group

```

9061 \renewcommand*{\glsgroupheading}[1]{%
9062 \par
9063 \def\@gls@prevlevel{-1}%
9064 \hangindent0pt\relax
9065 \parindent0pt\relax
9066 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
9067 \indexspace}%
9068 }

```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the `supertabular` environment.

```

9069 \ProvidesPackage{glossary-super}[2019/12/06 v4.44 (NLCT)]

```

Requires the package:

```

9070 \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.

```

9071 \@ifundefined{glsdescwidth}{%
9072 \newlength{glsdescwidth}
9073 \setlength{glsdescwidth}{0.6\hspace}
9074 }{}

```

`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.

```

9075 \@ifundefined{glspagelistwidth}{%
9076 \newlength{glspagelistwidth}
9077 \setlength{glspagelistwidth}{0.1\hspace}

```

9078 }{}

super The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

9079 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

9080 \renewenvironment{theglossary}{%

9081 {\tablehead{}\tabletail{}}%

9082 \begin{supertabular}{lp{\glsdescwidth}}%

9083 {\end{supertabular}}%

Do nothing at the start of the table:

9084 \renewcommand*{\glossaryheader}{}%

No group headings:

9085 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

9086 \renewcommand{\glossentry}[2]{%

9087 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

9088 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline

9089 }%

Sub entries put in a row (no name, description and page list in second column):

9090 \renewcommand{\subglossentry}[3]{%

9091 &

9092 \glssubentryitem{##2}%

9093 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space

9094 ##3\tabularnewline

9095 }%

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>)

9096 \ifglsnogroupskip

9097 \renewcommand*{\glsgroupskip}{}%

9098 \else

9099 \renewcommand*{\glsgroupskip}{& \tabularnewline}%

9100 \fi

9101 }

superborder The superborder style is like the above, but with horizontal and vertical lines:

9102 \newglossarystyle{superborder}{%

Base it on the glostylesuper style:

9103 \setglossarystyle{super}%

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

9104 \renewenvironment{theglossary}{%

9105 {\tablehead{\hline}\tabletail{\hline}%

```

9106      \begin{supertabular}{|l|p{\glsdescwidth}|}%
9107      {\end{supertabular}}%
9108 }

```

superheader The superheader style is like the super style, but with a header:

```

9109 \newglossarystyle{superheader}{%
      Base it on the glostylesuper style:
9110   \setglossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and no tail:
9111 \renewenvironment{theglossary}%
9112   {\tablehead{\bfseries \entryname &
9113     \bfseries\descriptionname\tabularnewline}%
9114     \tabletail{}}%
9115   \begin{supertabular}{lp{\glsdescwidth}}%
9116   {\end{supertabular}}%
9117 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```

9118 \newglossarystyle{superheaderborder}{%
      Base it on the glostylesuper style:
9119   \setglossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and horizontal
      lines above and below the table:
9120   \renewenvironment{theglossary}%
9121     {\tablehead{\hline\bfseries \entryname &
9122       \bfseries \descriptionname\tabularnewline\hline}%
9123       \tabletail{\hline}}
9124     \begin{supertabular}{|l|p{\glsdescwidth}|}%
9125     {\end{supertabular}}%
9126 }

```

super3col The super3col style is like the super style, but with 3 columns:

```

9127 \newglossarystyle{super3col}{%
      Put the glossary in a supertabular environment with three columns and no head or tail:
9128   \renewenvironment{theglossary}%
9129     {\tablehead{}\tabletail{}}%
9130     \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
9131     {\end{supertabular}}%
      Do nothing at the start of the table:
9132   \renewcommand*{\glossaryheader}{}%
      No group headings:
9133   \renewcommand*{\glsgroupheading}[1]{}%

```


Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

9134 \renewcommand{\glossentry}[2]{%
9135   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9136   \glossentrydesc{##1} & ##2\tabularnewline
9137 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```

9138 \renewcommand{\subglossentry}[3]{%
9139   &
9140   \glssubentryitem{##2}%
9141   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9142   ##3\tabularnewline
9143 }%
```

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>)

```

9144 \ifglsgroupskip
9145   \renewcommand*{\glsgroupskip}{}%
9146 \else
9147   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9148 \fi
9149 }
```

super3colborder The super3colborder style is like the super3col style, but with a border:

```

9150 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```

9151 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9152 \renewenvironment{theglossary}%
9153   {\tablehead{\hline}\tabletail{\hline}%
9154   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth|}}%
9155   {\end{supertabular}}%
9156 }
```

super3colheader The super3colheader style is like the super3col style but with a header row:

```

9157 \newglossarystyle{super3colheader}{%
```

Base it on the glostylesuper3col style:

```

9158 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9159 \renewenvironment{theglossary}%
9160   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9161     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9162   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
9163   {\end{supertabular}}%
9164 }
```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
9165 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
9166 \setglossarystyle{super3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9167 \renewenvironment{theglossary}{%
9168   {\tablehead{\hline
9169     \bfseries\entryname&\bfseries\descriptionname&
9170     \bfseries\pagelistname\tabularnewline\hline}%
9171   \tabletail{\hline}%
9172   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
9173   {\end{supertabular}}}%
9174 }
```

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.

```
9175 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9176 \renewenvironment{theglossary}{%
9177   {\tablehead{}\tabletail{}}%
9178   \begin{supertabular}{|l|l|l|l|}%
9179   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9180 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9181 \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:

```
9182 \renewcommand{\glossentry}[2]{%
9183   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9184   \glossentrydesc{##1} &
9185   \glossentrysymbol{##1} & ##2\tabularnewline
9186   }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9187 \renewcommand{\subglossentry}[3]{%
9188   &
9189   \glssubentryitem{##2}%
9190   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9191   \glossentrysymbol{##2} & ##3\tabularnewline
9192   }%
```

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>)

```

9193 \ifglsgroupskip
9194 \renewcommand*{\glsgroupskip}{}%
9195 \else
9196 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9197 \fi
9198 }

```

super4colheader The super4colheader style is like the super4col but with a header row.

```

9199 \newglossarystyle{super4colheader}{%
    Base it on the glostylesuper4col style:
9200 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns, a header and no tail:
9201 \renewenvironment{theglossary}%
9202 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9203 \bfseries\symbolname &
9204 \bfseries\pagelistname\tabularnewline}%
9205 \tabletail{}}%
9206 \begin{supertabular}{l111}}%
9207 {\end{supertabular}}%
9208 }

```

super4colborder The super4colborder style is like the super4col but with a border.

```

9209 \newglossarystyle{super4colborder}{%
    Base it on the glostylesuper4col style:
9210 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns and a horizontal line in the
    head and tail:
9211 \renewenvironment{theglossary}%
9212 {\tablehead{\hline}\tabletail{\hline}%
9213 \begin{supertabular}{|l|l|l|l|}%
9214 {\end{supertabular}}%
9215 }

```

colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```

9216 \newglossarystyle{super4colheaderborder}{%
    Base it on the glostylesuper4col style:
9217 \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:
9218 \renewenvironment{theglossary}%
9219 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
9220 \bfseries\symbolname &

```

```

9221      \bfseries\pagelistname\tabularnewline\hline}%
9222      \tabletail{\hline}%
9223      \begin{supertabular}{|l|l|l|l|}%
9224      {\end{supertabular}}}%
9225 }

```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```

9226 \newglossarystyle{altsuper4col}{%
      Base it on the glostylesuper4col style:
9227   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and no head or tail:
9228   \renewenvironment{theglossary}%
9229   {\tablehead{}\tabletail{}}%
9230   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
9231   {\end{supertabular}}}%
9232 }

```

super4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```

9233 \newglossarystyle{altsuper4colheader}{%
      Base it on the glostylesuper4colheader style:
9234   \setglossarystyle{super4colheader}%
      Put the glossary in a supertabular environment with four columns, a header and no tail:
9235   \renewenvironment{theglossary}%
9236   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9237     \bfseries\symbolname &
9238     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9239   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
9240   {\end{supertabular}}}%
9241 }

```

super4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```

9242 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
9243   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a horizontal line in the
      head and tail:
9244   \renewenvironment{theglossary}%
9245   {\tablehead{\hline}\tabletail{\hline}%
9246   \begin{supertabular}%
9247     {|l|lp{\glsgdescwidth}|l|lp{\glspagelistwidth}|}%
9248     {\end{supertabular}}}%
9249 }

```

colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```

9250 \newglossarystyle{altsuper4colheaderborder}{%

```

Base it on the `glostylesuper4colheaderborder` style:

```
9251 \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:

```
9252 \renewenvironment{theglossary}%
9253   {\tablehead{\hline
9254     \bfseries\entryname &
9255     \bfseries\descriptionname &
9256     \bfseries\symbolname &
9257     \bfseries\pagelistname\tabularnewline\hline}%
9258   \tabletail{\hline}%
9259   \begin{supertabular}%
9260     {lllp{\glsdescwidth}llp{\glspagelistwidth}}}%
9261   {\end{supertabular}}%
9262 }
```

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.

```
9263 \ProvidesPackage{glossary-superragged}[2019/12/06 v4.44 (NLCT)]
```

Requires the package:

```
9264 \RequirePackage{array}
```

Requires the package:

```
9265 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
9266 \@ifundefined{glsdescwidth}{%
9267   \newlength\glsdescwidth
9268   \setlength{\glsdescwidth}{0.6\hsize}
9269 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
9270 \@ifundefined{glspagelistwidth}{%
9271   \newlength\glspagelistwidth
9272   \setlength{\glspagelistwidth}{0.1\hsize}
9273 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
9274 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9275 \renewenvironment{theglossary}%
9276   {\tablehead{}\tabletail{}}%
9277   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9278   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9279 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9280 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9281 \renewcommand{\glossentry}[2]{%
9282   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9283   \glossentrydesc{##1}\glspostdescription\space ##2%
9284   \tabularnewline
9285   }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9286 \renewcommand{\subglossentry}[3]{%
9287   &
9288   \glssubentryitem{##2}%
9289   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9290   ##3%
9291   \tabularnewline
9292   }%
```

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>)

```
9293 \ifglsgroupskip
9294   \renewcommand*{\glsgroupskip}{}%
9295 \else
9296   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9297 \fi
9298 }
```

`\perraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
9299 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylesuperragged` style:

```
9300 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9301 \renewenvironment{theglossary}%
9302   {\tablehead{\hline}\tabletail{\hline}%
9303   \begin{supertabular}{l|l>{\raggedright}p{\glsgdescwidth}|}%
9304   {\end{supertabular}}%
9305 }
```

`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
9306 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
9307 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
9308 \renewenvironment{theglossary}%
```

```
9309 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
```

```
9310 \tabularnewline}%
```

```
9311 \tabletail{}}%
```

```
9312 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}%
```

```
9313 {\end{supertabular}}%
```

```
9314 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
9315 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
9316 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
9317 \renewenvironment{theglossary}%
```

```
9318 {\tablehead{\hline\bfseries \entryname &
```

```
9319 \bfseries \descriptionname\tabularnewline\hline}%
```

```
9320 \tabletail{\hline}
```

```
9321 \begin{supertabular}{ll>{\raggedright}p{\glsgdescwidth}}%
```

```
9322 {\end{supertabular}}%
```

```
9323 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
9324 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
9325 \renewenvironment{theglossary}%
```

```
9326 {\tablehead{}\tabletail{}}%
```

```
9327 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
```

```
9328 >{\raggedright}p{\glspagelistwidth}}%
```

```
9329 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9330 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9331 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9332 \renewcommand{\glossentry}[2]{%
```

```
9333 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9334 \glossentrydesc{##1} &
```

```

9335     ##2\tabularnewline
9336 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9337 \renewcommand{\subglossentry}[3]{%
9338     &
9339     \glssubentryitem{##2}%
9340     \glstarget{##2}{\strut}\glossentrydesc{##2} &
9341     ##3\tabularnewline
9342 }%

```

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>)

```

9343 \ifglsnogroupskip
9344 \renewcommand*{\glsgroupskip}{}%
9345 \else
9346 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9347 \fi
9348 }

```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```

9349 \newglossarystyle{superragged3colborder}{%

```

Base it on the glostylesuperragged3col style:

```

9350 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9351 \renewenvironment{theglossary}%
9352 {\tablehead{\hline}\tabletail{\hline}%
9353 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}||%
9354 >{\raggedright}p{\glspagelistwidth}||}%
9355 {\end{supertabular}}%
9356 }

```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```

9357 \newglossarystyle{superragged3colheader}{%

```

Base it on the glostylesuperragged3col style:

```

9358 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9359 \renewenvironment{theglossary}%
9360 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9361 \bfseries\pagelistname\tabularnewline}\tabletail{}%
9362 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9363 >{\raggedright}p{\glspagelistwidth}}%
9364 {\end{supertabular}}%
9365 }

```


colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9366 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9367 \setglossarystyle{superragged3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9368 \renewenvironment{theglossary}%
9369 {\tablehead{\hline
9370 \bfseries\entryname&\bfseries\descriptionname&
9371 \bfseries\pagelistname\tabularnewline\hline}%
9372 \tabletail{\hline}%
9373 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
9374 >{\raggedright}p{\glspagelistwidth}|}%
9375 {\end{supertabular}}%
9376 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9377 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9378 \renewenvironment{theglossary}%
9379 {\tablehead{}\tabletail{}%
9380 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9381 >{\raggedright}p{\glspagelistwidth}}%
9382 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9383 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9384 \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:

```
9385 \renewcommand{\glossentry}[2]{%
9386 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9387 \glossentrydesc{##1} &
9388 \glossentrysymbol{##1} & ##2\tabularnewline
9389 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9390 \renewcommand{\subglossentry}[3]{%
9391 &
9392 \glssubentryitem{##2}%
9393 \glstarget{##2}{\strut}\glossentrydesc{##2} &
9394 \glossentrysymbol{##2} & ##3\tabularnewline
9395 }%
```

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>)

```
9396 \ifglsgroupskip
9397 \renewcommand*{\glsgroupskip}{}%
9398 \else
9399 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9400 \fi
9401 }
```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
9402 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
9403 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9404 \renewenvironment{theglossary}%
9405 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9406 \bfseries\symbolname &
9407 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9408 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9409 >{\raggedright}p{\glspagelistwidth}}}%
9410 {\end{supertabular}}}%
9411 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
9412 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9413 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9414 \renewenvironment{theglossary}%
9415 {\tablehead{\hline}\tabletail{\hline}%
9416 \begin{supertabular}%
9417 {l|>{\raggedright}p{\glsdescwidth}l|}%
9418 >{\raggedright}p{\glspagelistwidth}l}}%
9419 {\end{supertabular}}}%
9420 }
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
9421 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9422 \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:

```

9423 \renewenvironment{theglossary}%
9424   {\tablehead{\hline
9425     \bfseries\entryname &
9426     \bfseries\descriptionname &
9427     \bfseries\symbolname &
9428     \bfseries\pagelistname\tabularnewline\hline}%
9429   \tabletail{\hline}%
9430   \begin{supertabular}%
9431     {||>{\raggedright}p{\glsgdescwidth}||}%
9432     >{\raggedright}p{\glspagelistwidth}||}%
9433   {\end{supertabular}}%
9434 }

```

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.

```

9435 \ProvidesPackage{glossary-tree}[2019/12/06 v4.44 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9436 \providecommand{\indexspace}{%
9437   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9438 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glsglfont`.) This command was previously also used to format the group headings.

```

9439 \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.

```

9440 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9441 \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.

```

9442 \ifdef\@idxitem
9443 {\newcommand{\glstreeitem}{\@idxitem}}
9444 {\newcommand{\glstreeitem}{\par\hangindent4\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```
9445 \ifdef\subitem
9446 {\let\glstreesubitem\subitem}
9447 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

`\glstreesubsubitem` Level 1 item used in index style.

```
9448 \ifdef\subsubitem
9449 {\let\glstreesubsubitem\subsubitem}
9450 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `alttree` style).

```
9451 \newcommand{\glstreepredesc}{\space}
```

`\glstreechildpredesc` Allow the user to adjust the space before the description for sub-entries (except for the `treenoname` and `alttree` style).

```
9452 \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.

```
9453 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9454 \renewenvironment{theglossary}%
9455 {\setlength{\parindent}{0pt}}%
9456 {\setlength{\parskip}{0pt plus 0.3pt}}%
9457 \let\item\glstreeitem
9458 \let\subitem\glstreesubitem
9459 \let\subsubitem\glstreesubsubitem
9460 }%
```

```
9461 {\par}}%
```

Do nothing at the start of the environment:

```
9462 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9463 \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.

```
9464 \renewcommand*{\glossentry}[2]{%
9465 \item\glstreeitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9466 \ifglshassymbol{##1}{\space\glossentrysymbol{##1}}{}%
9467 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9468 }%
```

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.

```

9469 \renewcommand{\subglossentry}[3]{%
9470   \ifcase##1\relax
9471     % level 0
9472     \item
9473   \or
9474     % level 1
9475     \subitem
9476     \glssubentryitem{##2}%
9477   \else
9478     % all other levels
9479     \subsubitem
9480   \fi
9481   \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}%
9482   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9483   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9484 }%
```

Vertical gap between groups is the same as that used by indices:

```

9485 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The `indexgroup` style is like the `index` style but has headings.

```

9486 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```

9487 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9488 \renewcommand*{\glsgroupheading}[1]{%
9489   \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9490   \indexspace
9491 }%
9492 }
```

indexhypergroup The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

9493 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```

9494 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```

9495 \renewcommand*{\glossaryheader}{%
9496   \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.

```

9497 \renewcommand*{\glsgroupheading}[1]{%
9498   \item\glstreegroupheaderfmt

```

```

9499      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
9500      \indexspace}%
9501 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9502 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9503   \renewenvironment{theglossary}%
9504     {\setlength{\parindent}{0pt}%
9505      \setlength{\parskip}{0pt plus 0.3pt}}%
9506     {}%

```

Do nothing at the start of the theglossary environment:

```

9507   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9508   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

9509   \renewcommand{\glossentry}[2]{%
9510     \hangindent0pt\relax
9511     \parindent0pt\relax
9512     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9513     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9514     \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9515   }%

```

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.

```

9516   \renewcommand{\subglossentry}[3]{%
9517     \hangindent##1\glstreeindent\relax
9518     \parindent##1\glstreeindent\relax
9519     \ifnum##1=1\relax
9520       \glssubentryitem{##2}%
9521       \fi
9522       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9523       \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9524       \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9525   }%

```

Vertical gap between groups is the same as that used by indices:

```

9526   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

9527 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

9528   \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```
9529 \renewcommand{\glsgroupheading}[1]{\par
9530 \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par
9531 \indexspace}%
9532 }
```

treehypergroup The **treehypergroup** style is like the **treegroup** style, but has a set of links to the groups at the start of the glossary.

```
9533 \newglossarystyle{treehypergroup}{%
```

Base it on the **glostyletree** style:

```
9534 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the **theglossary** environment:

```
9535 \renewcommand*{\glossaryheader}{%
9536 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9537 \renewcommand*{\glsgroupheading}[1]{%
9538 \par\noindent
9539 \glstreegroupheaderfmt
9540 {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9541 \indexspace}%
9542 }
```

\glstreeindent Length governing left indent for each level of the tree style.

```
9543 \newlength\glstreeindent
9544 \setlength{\glstreeindent}{10pt}
```

treenoname The **treenoname** glossary style is like the **tree** style, but doesn't print the name or symbol for sub-levels.

```
9545 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9546 \renewenvironment{theglossary}%
9547 {\setlength{\parindent}{0pt}%
9548 \setlength{\parskip}{0pt plus 0.3pt}}%
9549 {}%
```

No header:

```
9550 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9551 \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.

```
9552 \renewcommand{\glossentry}[2]{%
9553 \hangindent0pt\relax
9554 \parindent0pt\relax
9555 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```

9556 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9557 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9558 }%

```

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.

```

9559 \renewcommand{\subglossentry}[3]{%
9560 \hangindent##1\glstreeindent\relax
9561 \parindent##1\glstreeindent\relax
9562 \ifnum##1=1\relax
9563 \glssubentryitem{##2}%
9564 \fi
9565 \glstarget{##2}{\strut}%
9566 \glossentrydesc{##2}\glspostdescription\space##3\par
9567 }%

```

Vertical gap between groups is the same as that used by indices:

```

9568 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9569 }

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```

9570 \newglossarystyle{treenonamegroup}{%
  Base it on the glostyletreenoname style:
9571 \setglossarystyle{treenoname}%
  Give each group a heading:
9572 \renewcommand{\glsgroupheading}[1]{\par
9573 \noindent\glstreegroupheaderfmt
9574 {\glsgrouptitle{##1}}\par\indexspace}%
9575 }

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9576 \newglossarystyle{treenonamehypergroup}{%
  Base it on the glostyletreenoname style:
9577 \setglossarystyle{treenoname}%
  Put navigation links to the groups at the start of the theglossary environment:
9578 \renewcommand*{\glossaryheader}{%
9579 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9580 \renewcommand*{\glsgroupheading}[1]{%
9581 \par\noindent
9582 \glstreegroupheaderfmt
9583 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9584 \indexspace}%
9585 }

```


`esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```

9586 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9587   \dimen@=0pt\relax
9588   \gls@tmplen=0pt\relax
9589   \forallglossaries[#1]{\@gls@type}%
9590   {%
9591     \forallglsentries[\@gls@type]{\@glo@label}%
9592     {%
9593       \ifglsahasparent{\@glo@label}%
9594       }%
9595       {%
9596         \settowidth{\dimen@}%
9597           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
9598         \ifdim\dimen@>\gls@tmplen
9599           \gls@tmplen=\dimen@
9600           \letcs{\@glswidestname}{glo\glsdetoklabel{\@glo@label}@name}%
9601         \fi
9602       }%
9603     }%
9604   }%
9605 }
```

`\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.

```

9606 \newcommand*{\glssetwidest}[2][0]{%
9607   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9608     #2}%
9609 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```

9610 \newcommand*{\@glswidestname}{}
```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9611 \newcommand*{\glstreenamebox}[2]{%
9612   \makebox[#1][l]{#2}%
9613 }
```

`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`.

```

9614 \newglossarystyle{alttree}{%
  Redefine theglossary environment.
9615   \renewenvironment{theglossary}%
9616     {\def\@gls@prevlevel{-1}%
9617     \mbox{}\par}%
9618     {\par}%
  Set the header and group headers to nothing.
9619   \renewcommand*{\glossaryheader}{}%
9620   \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```
9621 \renewcommand{\glossentry}[2]{%
9622   \ifnum\@gls@prevlevel=0\relax
9623   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9624     \settowidth{\glstreeindent}{\glstreenamfmt{\@glswidestname\space}}%
9625   \fi
```

Set the hangindent and paragraph indent.

```
9626   \hangindent\glstreeindent
9627   \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9628   \makebox[0pt][r]{\glstreenambox{\glstreeindent}{%
9629     \glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9630   \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9631   \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9632   \def\@gls@prevlevel{0}%
9633 }%
```

Redefine the way sub-entries are displayed.

```
9634 \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.

```
9635   \ifnum##1=1\relax
9636     \glssubentryitem{##2}%
9637   \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9638   \ifnum\@gls@prevlevel=##1\relax
9639   \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9640     \@ifundefined{@glswidestname\romannumeral##1}{%
9641       \settowidth{\gls@tmplen}{\glstreenamfmt{\@glswidestname\space}}{%
9642       \settowidth{\gls@tmplen}{\glstreenamfmt{%
9643         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9644   \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9645      \setlength\glstreeindent\gls@tmplen
9646      \addtolength\glstreeindent\parindent
9647      \parindent\glstreeindent
9648      \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`.

```
9649      \ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9650      \settowidth{\glstreeindent}{\glstreenamfmt{%
9651      \@glswidestname\space}}}{%
9652      \settowidth{\glstreeindent}{\glstreenamfmt{%
9653      \csname @glswidestname\romannumeral\@gls@prevlevel
9654      \endcsname\space}}}{%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9655      \addtolength\parindent{-\glstreeindent}%
9656      \setlength\glstreeindent\parindent
9657      \fi
9658      \fi
```

Set the hanging indentation.

```
9659      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9660      \makebox[0pt][r]{\glstreenambox{\gls@tmplen}{%
9661      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9662      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9663      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9664      \def\@gls@prevlevel{##1}%
9665      }%
```

Vertical gap between groups is the same as that used by indices:

```
9666      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9667 }
```

`altnogroup` Like the `altnogroup` style but the glossary groups have headings.

```
9668 \newglossarystyle{altnogroup}{%
```

Base it on the `glostylealtnogroup` style:

```
9669      \setglossarystyle{altnogroup}%
```

Give each group a heading.

```
9670      \renewcommand{\glsgroupheading}[1]{\par
9671      \def\@gls@prevlevel{-1}%
9672      \hangindent0pt\relax
```

```

9673     \parindent0pt\relax
9674     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
9675     \par\indexspace}%
9676 }

```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9677 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9678   \setglossarystyle{alttree}%
    Put the navigation links in the header
9679   \renewcommand*{\glossaryheader}{%
9680     \par
9681     \def\@gls@prevlevel{-1}%
9682     \hangindent0pt\relax
9683     \parindent0pt\relax
9684     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9685   \renewcommand*{\glsgroupheading}[1]{%
9686     \par
9687     \def\@gls@prevlevel{-1}%
9688     \hangindent0pt\relax
9689     \parindent0pt\relax
9690     \glstreegroupheaderfmt
9691     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9692     \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.

```
9693 \NeedsTeXFormat{LaTeX2e}
9694 \ProvidesPackage{glossaries-compatible-207}[2019/12/06 v4.44 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9695 \ifglxsindy
9696   \renewcommand*\GlsAddXdyAttribute[1]{%
9697     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9698     \expandafter\toks@\expandafter{\@xdylocref}%
9699     \edef\@xdylocref{\the\toks@ ^^J%
9700     (markup-locref
9701     :open \string"\string~n\string\setentrycounter
9702     {\noexpand\glscounter}%
9703     \expandafter\string\csname#1\endcsname
9704     \expandafter\@gobble\string\{\string" ^^J
9705     :close \string"\expandafter\@gobble\string\}\string" ^^J
9706     :attr \string"#1\string")}}
```

Only has an effect before `\writeist`:

```
9707 \fi
```

sAddXdyCounters

```
9708 \renewcommand*\GlsAddXdyCounters[1]{%
9709   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9710     in compatibility mode.}%
9711 }
```

Add predefined attributes

```
9712 \GlsAddXdyAttribute{glsnumberformat}
9713 \GlsAddXdyAttribute{textrm}
9714 \GlsAddXdyAttribute{textsf}
9715 \GlsAddXdyAttribute{texttt}
9716 \GlsAddXdyAttribute{textbf}
9717 \GlsAddXdyAttribute{textmd}
9718 \GlsAddXdyAttribute{textit}
9719 \GlsAddXdyAttribute{textup}
9720 \GlsAddXdyAttribute{textsl}
```

```

9721 \GlsAddXdyAttribute{textsc}
9722 \GlsAddXdyAttribute{emph}
9723 \GlsAddXdyAttribute{glshypernumber}
9724 \GlsAddXdyAttribute{hyperrm}
9725 \GlsAddXdyAttribute{hypersf}
9726 \GlsAddXdyAttribute{hypertt}
9727 \GlsAddXdyAttribute{hyperbf}
9728 \GlsAddXdyAttribute{hypermd}
9729 \GlsAddXdyAttribute{hyperit}
9730 \GlsAddXdyAttribute{hyperup}
9731 \GlsAddXdyAttribute{hypersl}
9732 \GlsAddXdyAttribute{hypersc}
9733 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9734 \ifglxindy
9735   \renewcommand*{\GlsAddXdyLocation}[2]{%
9736     \edef\@xdyuserlocationdefs{%
9737       \@xdyuserlocationdefs ^^J%
9738       (define-location-class \string"#1\string"^^J\space\space
9739       \space(#2))
9740     }%
9741     \edef\@xdyuserlocationnames{%
9742       \@xdyuserlocationnames^^J\space\space\space
9743       \string"#1\string"}%
9744   }
9745 \fi

```

\@do@wrglossary

```

9746 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9747 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9748   \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9749   \def\@glo@range{}%
9750   \expandafter\if\@glo@prefix\relax
9751   \def\@glo@range{:open-range}%
9752   \else
9753   \expandafter\if\@glo@prefix\relax
9754   \def\@glo@range{:close-range}%
9755   \fi
9756 \fi

  Get the location and escape any special characters
9757   \protected@edef\@glslocref{\theglsentrycounter}%
9758   \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9759   \glossary[\csname glo@#1@type\endcsname]{%

```

```

9760 (indexentry :tkey (\csname glo@#1@index\endcsname)
9761   :locoref \string"\@glslocoref\string" %
9762   :attr \string"\@glo@suffix\string" \@glo@range
9763 )
9764 }%
9765 \else

```

Convert the format information into the format required for makeindex

```

9766 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9767 \glossary[\csname glo@#1@type\endcsname]{%
9768 \string\glossaryentry{\csname glo@#1@index\endcsname
9769   \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9770 \fi
9771 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9772 \def\@set@glo@numformat#1#2#3{%
9773   \expandafter\@glo@check@mkidxrangechar#3\@nil
9774   \protected@edef#1{%
9775     \@glo@prefix setentrycounter[]{\#2}%
9776     \expandafter\string\csname\@glo@suffix\endcsname
9777   }%
9778   \@gls@checkmkidxchars#1%
9779 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9780 \ifglxsindy
9781   \def\writeist{%
9782     \openout\glswrite=\istfilename
9783     \write\glswrite{;; xindy style file created by the glossaries
9784       package in compatible-2.07 mode}%
9785     \write\glswrite{;; for document '\jobname' on
9786       \the\year-\the\month-\the\day}%
9787     \write\glswrite{^^J; required styles^^J}
9788     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9789       \ifx\@xdystyle\@empty
9790       \else
9791         \protected@write\glswrite{{(require
9792           \string"\@xdystyle.xdy\string")}}%
9793       \fi
9794     }%
9795     \write\glswrite{^^J%
9796       ; list of allowed attributes (number formats)^^J}%
9797     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9798     \write\glswrite{^^J; user defined alphabets^^J}%
9799     \write\glswrite{\@xdyuseralphabets}%
9800     \write\glswrite{^^J; location class definitions^^J}%
9801     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9802     \string"roman-numbers-lowercase\string" :sep \string"}}%
9803 \@onelevel@sanitize\@gls@roman
9804 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9805     :sep \string"}}%
9806 \@onelevel@sanitize\@tmp
9807 \ifx\@tmp\@gls@roman
9808     \write\glswrite{(define-location-class
9809         \string"roman-page-numbers\string"^^J\space\space\space
9810         (\string"roman-numbers-lowercase\string")
9811         :min-range-length \@glsminrange)}}%
9812 \else
9813     \write\glswrite{(define-location-class
9814         \string"roman-page-numbers\string"^^J\space\space\space
9815         (:sep "\@gls@roman")
9816         :min-range-length \@glsminrange)}}%
9817 \fi
9818 \write\glswrite{(define-location-class
9819     \string"Roman-page-numbers\string"^^J\space\space\space
9820     (\string"roman-numbers-uppercase\string")
9821     :min-range-length \@glsminrange)}}%
9822 \write\glswrite{(define-location-class
9823     \string"arabic-page-numbers\string"^^J\space\space\space
9824     (\string"arabic-numbers\string")
9825     :min-range-length \@glsminrange)}}%
9826 \write\glswrite{(define-location-class
9827     \string"alpha-page-numbers\string"^^J\space\space\space
9828     (\string"alpha\string")
9829     :min-range-length \@glsminrange)}}%
9830 \write\glswrite{(define-location-class
9831     \string"Alpha-page-numbers\string"^^J\space\space\space
9832     (\string"ALPHA\string")
9833     :min-range-length \@glsminrange)}}%
9834 \write\glswrite{(define-location-class
9835     \string"Appendix-page-numbers\string"^^J\space\space\space
9836     (\string"ALPHA\string"
9837     :sep \string"\@glsAlphacompositor\string"
9838     \string"arabic-numbers\string")
9839     :min-range-length \@glsminrange)}}%
9840 \write\glswrite{(define-location-class
9841     \string"arabic-section-numbers\string"^^J\space\space\space
9842     (\string"arabic-numbers\string"
9843     :sep \string"\glscompositor\string"
9844     \string"arabic-numbers\string")
9845     :min-range-length \@glsminrange)}}%
9846 \write\glswrite{^^J; user defined location classes}%
9847 \write\glswrite{\@xdyuserlocationdefs}%
9848 \write\glswrite{^^J; define cross-reference class^^J}%
9849 \write\glswrite{(define-crossref-class \string"see\string"
9850     :unverified )}%

```



```

9851 \write\glswrite{(markup-crossref-list
9852 :class \string"see\string"^^J\space\space\space
9853 :open \string"\string\glssseeformat\string"
9854 :close \string"{}\string")}%
9855 \write\glswrite{^^J; define the order of the location classes}%
9856 \write\glswrite{(define-location-class-order
9857 (\@xdylocationclassorder))}%
9858 \write\glswrite{^^J; define the glossary markup^^J}%
9859 \write\glswrite{(markup-index^^J\space\space\space
9860 :open \string"\string
9861 \glossarysection[\string\glossarytoctitle]{\string
9862 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9863 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9864 \space\space:close \string"\expandafter\@gobble
9865 \string%\string~n\string
9866 \end{theglossary}\string\glossarypostamble
9867 \string~n\string" ^^J\space\space\space
9868 :tree)}}%
9869 \write\glswrite{(markup-letter-group-list
9870 :sep \string"\string\glsgroupskip\string~n\string")}%
9871 \write\glswrite{(markup-indexentry
9872 :open \string"\string\relax \string\glssresetentrylist
9873 \string~n\string")}%
9874 \write\glswrite{(markup-locclass-list :open
9875 \string"\glsoopenbrace\string\glossaryentrynumbers
9876 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9877 :sep \string", \string"
9878 :close \string"\glsclosebrace\glsclosebrace\string")}%
9879 \write\glswrite{(markup-locref-list
9880 :sep \string"\string\delimN\space\string")}%
9881 \write\glswrite{(markup-range
9882 :sep \string"\string\delimR\space\string")}%
9883 \@onelevel@sanitize\gls@suffixF
9884 \@onelevel@sanitize\gls@suffixFF
9885 \ifx\gls@suffixF\@empty
9886 \else
9887 \write\glswrite{(markup-range
9888 :close "\gls@suffixF" :length 1 :ignore-end)}%
9889 \fi
9890 \ifx\gls@suffixFF\@empty
9891 \else
9892 \write\glswrite{(markup-range
9893 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9894 \fi
9895 \write\glswrite{^^J; define format to use for locations^^J}%
9896 \write\glswrite{\@xdylocref}%
9897 \write\glswrite{^^J; define letter group list format^^J}%
9898 \write\glswrite{(markup-letter-group-list
9899 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9900 \write\glswrite{^^J; letter group headings^^J}%
9901 \write\glswrite{(markup-letter-group
9902   :open-head \string"\string\glsgroupheading
9903   \glsoopenbrace\string"^^J\space\space\space
9904   :close-head \string"\glsclosebrace\string")}%
9905 \write\glswrite{^^J; additional letter groups^^J}%
9906 \write\glswrite{\@xdylettergroups}%
9907 \write\glswrite{^^J; additional sort rules^^J}
9908 \write\glswrite{\@xdysortrules}%
9909 \noist}
9910 \else
9911 \edef\@gls@actualchar{\string?}
9912 \edef\@gls@encapchar{\string|}
9913 \edef\@gls@levelchar{\string!}
9914 \edef\@gls@quotechar{\string"}
9915 \def\writeist{\relax
9916   \openout\glswrite=\istfilename
9917   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9918     created by the glossaries package}
9919   \write\glswrite{\expandafter\@gobble\string\% for document
9920     '\jobname' on \the\year-\the\month-\the\day}
9921   \write\glswrite{actual '\@gls@actualchar'}
9922   \write\glswrite{encap '\@gls@encapchar'}
9923   \write\glswrite{level '\@gls@levelchar'}
9924   \write\glswrite{quote '\@gls@quotechar'}
9925   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9926   \write\glswrite{preamble \string"\string\glossarysection[\string
9927     \glossarytoctitle]{\string\glossarytitle}\string
9928     \glossarypreamble\string\n\string\begin{theglossary}\string
9929     \glossaryheader\string\n\string"}
9930   \write\glswrite{postamble \string"\string%\string\n\string
9931     \end{theglossary}\string\glossarypostamble\string\n
9932     \string"}
9933   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9934     \string"}
9935   \write\glswrite{item_0 \string"\string%\string\n\string"}
9936   \write\glswrite{item_1 \string"\string%\string\n\string"}
9937   \write\glswrite{item_2 \string"\string%\string\n\string"}
9938   \write\glswrite{item_01 \string"\string%\string\n\string"}
9939   \write\glswrite{item_x1
9940     \string"\string\relax \string\glsresetentrylist\string\n
9941     \string"}
9942   \write\glswrite{item_12 \string"\string%\string\n\string"}
9943   \write\glswrite{item_x2
9944     \string"\string\relax \string\glsresetentrylist\string\n
9945     \string"}
9946   \write\glswrite{delim_0 \string"\string\{\string
9947     \glossaryentrynumbers\string\{\string\relax \string"}
9948   \write\glswrite{delim_1 \string"\string\{\string

```

```

9949     \glossaryentrynumbers\string\{\string\relax \string}
9950 \write\glswrite{delim_2 \string"\string\{\string
9951     \glossaryentrynumbers\string\{\string\relax \string}
9952 \write\glswrite{delim_t \string"\string\}\string\}\string}
9953 \write\glswrite{delim_n \string"\string\delimN \string}
9954 \write\glswrite{delim_r \string"\string\delimR \string}
9955 \write\glswrite{headings_flag 1}
9956 \write\glswrite{heading_prefix
9957     \string"\string\glsgroupheading\string\{\string}
9958 \write\glswrite{heading_suffix
9959     \string"\string\}\string\relax
9960     \string\glresetentrylist \string}
9961 \write\glswrite{symhead_positive \string"glssymbols\string}
9962 \write\glswrite{numhead_positive \string"glnumbers\string}
9963 \write\glswrite{page_compositor \string"glscpositor\string}
9964 \@gls@escbsdq\gls@suffixF
9965 \@gls@escbsdq\gls@suffixFF
9966 \ifx\gls@suffixF\@empty
9967 \else
9968     \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9969 \fi
9970 \ifx\gls@suffixFF\@empty
9971 \else
9972     \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9973 \fi
9974 \noist
9975 }
9976 \fi

```

\noist

```

9977 \renewcommand*{\noist}{\let\writeist\relax}

```

4.2 glossaries-compatible-307

```

9978 \NeedsTeXFormat{LaTeX2e}
9979 \ProvidesPackage{glossaries-compatible-307}[2019/12/06 v4.44 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9980 \newcommand{\compatglossarystyle}[2]{%
9981   \ifcsundef{@glscompstyle@#1}%
9982   {%
9983     \csdef{@glscompstyle@#1}{#2}%
9984   }%
9985   {%
9986     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9987   }%
9988 }

```

Backward compatible inline style.

```

9989 \compatglossarystyle{inline}{%
9990   \renewcommand{\glossaryentryfield}[5]{%
9991     \glsinlinedopostchild
9992     \gls@inlinesep
9993     \def\glo@desc{##3}%
9994     \def\@no@post@desc{\nopostdesc}%
9995     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
9996     \ifx\glo@desc\@no@post@desc
9997       \glsinlineemptydescformat{##4}{##5}%
9998     \else
9999       \ifstrepty{##3}%
10000       {\glsinlineemptydescformat{##4}{##5}}%
10001       {\glsinlinedescformat{##3}{##4}{##5}}%
10002     \fi
10003     \ifglshaschildren{##1}%
10004     {%
10005       \glsresetsubentrycounter
10006       \glsinlineparentchildseparator
10007       \def\gls@inlinesubsep{}%
10008       \def\gls@inlinepostchild{\glsinlinepostchild}%
10009     }%
10010   }%
10011   \def\gls@inlinesep{\glsinlineseparator}%
10012 }%

```

Sub-entries display description:

```

10013 \renewcommand{\glossarysubentryfield}[6]{%
10014   \gls@inlinesubsep%
10015   \glsinlinesubnameformat{##2}{##3}%
10016   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
10017   \def\gls@inlinesubsep{\glsinlinesubseparator}%
10018 }%
10019 }

```

Backward compatible list style.

```

10020 \compatglossarystyle{list}{%
10021   \renewcommand*\{\glossaryentryfield}[5]{%
10022     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
10023     ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```

10024   \renewcommand*\{\glossarysubentryfield}[6]{%
10025     \glssubentryitem{##2}%
10026     \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
10027 }

```

Backward compatible listgroup style.

```

10028 \compatglossarystyle{listgroup}{%
10029   \csuse{@glscompstyle@list}%
10030 }%

```

Backward compatible listhypergroup style.

```
10031 \compatglossarystyle{listhypergroup}{%
10032   \csuse{@glscompstyle@list}%
10033 }%
```

Backward compatible altlist style.

```
10034 \compatglossarystyle{altlist}{%
10035   \renewcommand*{\glossaryentryfield}[5]{%
10036     \item[\glentryitem{##1}\glstarget{##1}{##2}]%
10037       \mbox{}\par\nobreak\@afterheading
10038       ##3\glspostdescription\space ##5}%
10039   \renewcommand{\glossarysubentryfield}[6]{%
10040     \par
10041     \glssubentryitem{##2}%
10042     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
10043 }%
```

Backward compatible altlistgroup style.

```
10044 \compatglossarystyle{altlistgroup}{%
10045   \csuse{@glscompstyle@altlist}%
10046 }%
```

Backward compatible altlisthypergroup style.

```
10047 \compatglossarystyle{altlisthypergroup}{%
10048   \csuse{@glscompstyle@altlist}%
10049 }%
```

Backward compatible listdotted style.

```
10050 \compatglossarystyle{listdotted}{%
10051   \renewcommand*{\glossaryentryfield}[5]{%
10052     \item[]\makebox[\glslistdottedwidth][l]{%
10053       \glentryitem{##1}\glstarget{##1}{##2}%
10054       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
10055   \renewcommand*{\glossarysubentryfield}[6]{%
10056     \item[]\makebox[\glslistdottedwidth][l]{%
10057       \glssubentryitem{##2}%
10058       \glstarget{##2}{##3}%
10059       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
10060 }%
```

Backward compatible sublistdotted style.

```
10061 \compatglossarystyle{sublistdotted}{%
10062   \csuse{@glscompstyle@listdotted}%
10063   \renewcommand*{\glossaryentryfield}[5]{%
10064     \item[\glentryitem{##1}\glstarget{##1}{##2}]}%
10065 }%
```

Backward compatible long style.

```
10066 \compatglossarystyle{long}{%
10067   \renewcommand*{\glossaryentryfield}[5]{%
10068     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10069   \renewcommand*{\glossarysubentryfield}[6]{%
10070     \glssubentryitem{##2}\glstarget{##2}{##3} & ##4\glspostdescription\space ##5\\}%
10071 }%
```

```

10070      &
10071      \glssubentryitem{##2}%
10072      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10073 }%

```

Backward compatible longborder style.

```

10074 \compatglossarystyle{longborder}{%
10075   \csuse{@glscmpstyle@long}%
10076 }%

```

Backward compatible longheader style.

```

10077 \compatglossarystyle{longheader}{%
10078   \csuse{@glscmpstyle@long}%
10079 }%

```

Backward compatible longheaderborder style.

```

10080 \compatglossarystyle{longheaderborder}{%
10081   \csuse{@glscmpstyle@long}%
10082 }%

```

Backward compatible long3col style.

```

10083 \compatglossarystyle{long3col}{%
10084   \renewcommand*{\glossaryentryfield}[5]{%
10085     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10086   \renewcommand*{\glossarysubentryfield}[6]{%
10087     &
10088     \glssubentryitem{##2}%
10089     \glstarget{##2}{\strut}##4 & ##6\\}%
10090 }%

```

Backward compatible long3colborder style.

```

10091 \compatglossarystyle{long3colborder}{%
10092   \csuse{@glscmpstyle@long3col}%
10093 }%

```

Backward compatible long3colheader style.

```

10094 \compatglossarystyle{long3colheader}{%
10095   \csuse{@glscmpstyle@long3col}%
10096 }%

```

Backward compatible long3colheaderborder style.

```

10097 \compatglossarystyle{long3colheaderborder}{%
10098   \csuse{@glscmpstyle@long3col}%
10099 }%

```

Backward compatible long4col style.

```

10100 \compatglossarystyle{long4col}{%
10101   \renewcommand*{\glossaryentryfield}[5]{%
10102     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10103   \renewcommand*{\glossarysubentryfield}[6]{%
10104     &
10105     \glssubentryitem{##2}%

```

10106 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10107 }%

Backward compatible long4colheader style.

10108 \compatglossarystyle{long4colheader}{%
10109 \csuse{@glscompstyle@long4col}%
10110 }%

Backward compatible long4colborder style.

10111 \compatglossarystyle{long4colborder}{%
10112 \csuse{@glscompstyle@long4col}%
10113 }%

Backward compatible long4colheaderborder style.

10114 \compatglossarystyle{long4colheaderborder}{%
10115 \csuse{@glscompstyle@long4col}%
10116 }%

Backward compatible altlong4col style.

10117 \compatglossarystyle{altlong4col}{%
10118 \csuse{@glscompstyle@long4col}%
10119 }%

Backward compatible altlong4colheader style.

10120 \compatglossarystyle{altlong4colheader}{%
10121 \csuse{@glscompstyle@long4col}%
10122 }%

Backward compatible altlong4colborder style.

10123 \compatglossarystyle{altlong4colborder}{%
10124 \csuse{@glscompstyle@long4col}%
10125 }%

Backward compatible altlong4colheaderborder style.

10126 \compatglossarystyle{altlong4colheaderborder}{%
10127 \csuse{@glscompstyle@long4col}%
10128 }%

Backward compatible long style.

10129 \compatglossarystyle{longragged}{%
10130 \renewcommand*{\glossaryentryfield}[5]{%
10131 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10132 \tabularnewline}%
10133 \renewcommand*{\glossarysubentryfield}[6]{%
10134 &
10135 \glssubentryitem{##2}%
10136 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10137 \tabularnewline}%
10138 }%

Backward compatible longraggedborder style.

10139 \compatglossarystyle{longraggedborder}{%
10140 \csuse{@glscompstyle@longragged}%
10141 }%

Backward compatible longraggedheader style.

```
10142 \compatglossarystyle{longraggedheader}{%  
10143 \csuse{@glscompstyle@longragged}%  
10144 }%
```

Backward compatible longraggedheaderborder style.

```
10145 \compatglossarystyle{longraggedheaderborder}{%  
10146 \csuse{@glscompstyle@longragged}%  
10147 }%
```

Backward compatible longragged3col style.

```
10148 \compatglossarystyle{longragged3col}{%  
10149 \renewcommand*{\glossaryentryfield}[5]{%  
10150 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
10151 \renewcommand*{\glossarysubentryfield}[6]{%  
10152 &  
10153 \glssubentryitem{##2}%  
10154 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
10155 }%
```

Backward compatible longragged3colborder style.

```
10156 \compatglossarystyle{longragged3colborder}{%  
10157 \csuse{@glscompstyle@longragged3col}%  
10158 }%
```

Backward compatible longragged3colheader style.

```
10159 \compatglossarystyle{longragged3colheader}{%  
10160 \csuse{@glscompstyle@longragged3col}%  
10161 }%
```

Backward compatible longragged3colheaderborder style.

```
10162 \compatglossarystyle{longragged3colheaderborder}{%  
10163 \csuse{@glscompstyle@longragged3col}%  
10164 }%
```

Backward compatible altlongragged4col style.

```
10165 \compatglossarystyle{altlongragged4col}{%  
10166 \renewcommand*{\glossaryentryfield}[5]{%  
10167 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
10168 \renewcommand*{\glossarysubentryfield}[6]{%  
10169 &  
10170 \glssubentryitem{##2}%  
10171 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
10172 }%
```

Backward compatible altlongragged4colheader style.

```
10173 \compatglossarystyle{altlongragged4colheader}{%  
10174 \csuse{@glscompstyle@altlong4col}%  
10175 }%
```

Backward compatible altlongragged4colborder style.

```
10176 \compatglossarystyle{altlongragged4colborder}{%
```



```
10177 \csuse{@glscompstyle@altlong4col}%
10178 }%
```

Backward compatible altlongragged4colheaderborder style.

```
10179 \compatglossarystyle{altlongragged4colheaderborder}{%
10180 \csuse{@glscompstyle@altlong4col}%
10181 }%
```

Backward compatible index style.

```
10182 \compatglossarystyle{index}{%
10183 \renewcommand*{\glossaryentryfield}[5]{%
10184 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10185 \ifx\relax##4\relax
10186 \else
10187 \space{##4}%
10188 \fi
10189 \space ##3\glspostdescription \space ##5}%
10190 \renewcommand*{\glossarysubentryfield}[6]{%
10191 \ifcase##1\relax
10192 % level 0
10193 \item
10194 \or
10195 % level 1
10196 \subitem
10197 \glssubentryitem{##2}%
10198 \else
10199 % all other levels
10200 \subsubitem
10201 \fi
10202 \textbf{\glstarget{##2}{##3}}%
10203 \ifx\relax##5\relax
10204 \else
10205 \space{##5}%
10206 \fi
10207 \space##4\glspostdescription\space ##6}%
10208 }%
```

Backward compatible indexgroup style.

```
10209 \compatglossarystyle{indexgroup}{%
10210 \csuse{@glscompstyle@index}%
10211 }%
```

Backward compatible indexhypergroup style.

```
10212 \compatglossarystyle{indexhypergroup}{%
10213 \csuse{@glscompstyle@index}%
10214 }%
```

Backward compatible tree style.

```
10215 \compatglossarystyle{tree}{%
10216 \renewcommand{\glossaryentryfield}[5]{%
10217 \hangindent0pt\relax
```

```

10218 \parindent0pt\relax
10219 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10220 \ifx\relax##4\relax
10221 \else
10222 \space{##4}%
10223 \fi
10224 \space ##3\glspostdescription \space ##5\par}%
10225 \renewcommand{\glossarysubentryfield}[6]{%
10226 \hangindent##1\glstreeindent\relax
10227 \parindent##1\glstreeindent\relax
10228 \ifnum##1=1\relax
10229 \glssubentryitem{##2}%
10230 \fi
10231 \textbf{\glstarget{##2}{##3}}%
10232 \ifx\relax##5\relax
10233 \else
10234 \space{##5}%
10235 \fi
10236 \space##4\glspostdescription\space ##6\par}%
10237 }%

```

Backward compatible treegroup style.

```

10238 \compatglossarystyle{treegroup}{%
10239 \csuse{@glscmpstyle@tree}%
10240 }%

```

Backward compatible treehypergroup style.

```

10241 \compatglossarystyle{treehypergroup}{%
10242 \csuse{@glscmpstyle@tree}%
10243 }%

```

Backward compatible treenoname style.

```

10244 \compatglossarystyle{treenoname}{%
10245 \renewcommand{\glossaryentryfield}[5]{%
10246 \hangindent0pt\relax
10247 \parindent0pt\relax
10248 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10249 \ifx\relax##4\relax
10250 \else
10251 \space{##4}%
10252 \fi
10253 \space ##3\glspostdescription \space ##5\par}%
10254 \renewcommand{\glossarysubentryfield}[6]{%
10255 \hangindent##1\glstreeindent\relax
10256 \parindent##1\glstreeindent\relax
10257 \ifnum##1=1\relax
10258 \glssubentryitem{##2}%
10259 \fi
10260 \glstarget{##2}{\strut}%
10261 ##4\glspostdescription\space ##6\par}%
10262 }%

```

Backward compatible treenonamegroup style.

```
10263 \compatglossarystyle{treenonamegroup}{%
10264   \csuse{@glscompstyle@treenoname}%
10265 }%
```

Backward compatible treenonamehypergroup style.

```
10266 \compatglossarystyle{treenonamehypergroup}{%
10267   \csuse{@glscompstyle@treenoname}%
10268 }%
```

Backward compatible alttree style.

```
10269 \compatglossarystyle{alttree}{%
10270   \renewcommand{\glossaryentryfield}[5]{%
10271     \ifnum\@gls@prevlevel=0\relax
10272     \else
10273       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10274       \hangindent\glstreeindent
10275       \parindent\glstreeindent
10276     \fi
10277     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10278       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10279     \ifx\relax##4\relax
10280     \else
10281       (##4)\space
10282     \fi
10283     ##3\glspostdescription \space ##5\par
10284     \def\@gls@prevlevel{0}%
10285   }%
10286   \renewcommand{\glossarysubentryfield}[6]{%
10287     \ifnum##1=1\relax
10288       \glssubentryitem{##2}%
10289     \fi
10290     \ifnum\@gls@prevlevel=##1\relax
10291     \else
10292       \@ifundefined{@glswidestname\romannumeral##1}{%
10293         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10294         \settowidth{\gls@tmplen}{\textbf{%
10295           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10296         \ifnum\@gls@prevlevel<##1\relax
10297           \setlength\glstreeindent\gls@tmplen
10298           \addtolength\glstreeindent\parindent
10299           \parindent\glstreeindent
10300         \else
10301           \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10302             \settowidth{\glstreeindent}{\textbf{%
10303               \@glswidestname\space}}{%
10304             \settowidth{\glstreeindent}{\textbf{%
10305               \csname @glswidestname\romannumeral\@gls@prevlevel
10306               \endcsname\space}}}%
10307             \addtolength\parindent{-\glstreeindent}%

```

```

10308      \setlength\glstreeindent\parindent
10309      \fi
10310      \fi
10311      \hangindent\glstreeindent
10312      \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10313        \textbf{\glstarget{##2}{##3}}}%
10314      \ifx##5\relax\relax
10315      \else
10316        (##5)\space
10317      \fi
10318      ##4\glspostdescription\space ##6\par
10319      \def\@gls@prevlevel{##1}%
10320    }%
10321  }%

```

Backward compatible alttreegroup style.

```

10322 \compatglossarystyle{alttreegroup}{%
10323   \csuse{@glscompstyle@alttree}%
10324 }%

```

Backward compatible alttreehypergroup style.

```

10325 \compatglossarystyle{alttreehypergroup}{%
10326   \csuse{@glscompstyle@alttree}%
10327 }%

```

Backward compatible mcolindex style.

```

10328 \compatglossarystyle{mcolindex}{%
10329   \csuse{@glscompstyle@index}%
10330 }%

```

Backward compatible mcolindexgroup style.

```

10331 \compatglossarystyle{mcolindexgroup}{%
10332   \csuse{@glscompstyle@index}%
10333 }%

```

Backward compatible mcolindexhypergroup style.

```

10334 \compatglossarystyle{mcolindexhypergroup}{%
10335   \csuse{@glscompstyle@index}%
10336 }%

```

Backward compatible mcoltree style.

```

10337 \compatglossarystyle{mcoltree}{%
10338   \csuse{@glscompstyle@tree}%
10339 }%

```

Backward compatible mcoltreegroup style.

```

10340 \compatglossarystyle{mcolindextreegroup}{%
10341   \csuse{@glscompstyle@tree}%
10342 }%

```

Backward compatible mcoltreehypergroup style.

```

10343 \compatglossarystyle{mcolindextreehypergroup}{%

```

10344 \csuse{@glscompstyle@tree}%
 10345 }%

Backward compatible mcoltreenoname style.

10346 \compatglossarystyle{mcoltreenoname}{%
 10347 \csuse{@glscompstyle@tree}%
 10348 }%

Backward compatible mcoltreenonamegroup style.

10349 \compatglossarystyle{mcoltreenonamegroup}{%
 10350 \csuse{@glscompstyle@tree}%
 10351 }%

Backward compatible mcoltreenonamehypergroup style.

10352 \compatglossarystyle{mcoltreenonamehypergroup}{%
 10353 \csuse{@glscompstyle@tree}%
 10354 }%

Backward compatible mcolalmtree style.

10355 \compatglossarystyle{mcolalmtree}{%
 10356 \csuse{@glscompstyle@almtree}%
 10357 }%

Backward compatible mcolalmtreegroup style.

10358 \compatglossarystyle{mcolalmtreegroup}{%
 10359 \csuse{@glscompstyle@almtree}%
 10360 }%

Backward compatible mcolalmtreehypergroup style.

10361 \compatglossarystyle{mcolalmtreehypergroup}{%
 10362 \csuse{@glscompstyle@almtree}%
 10363 }%

Backward compatible superragged style.

10364 \compatglossarystyle{superragged}{%
 10365 \renewcommand*{\glossaryentryfield}[5]{%
 10366 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 10367 \tabularnewline}%
 10368 \renewcommand*{\glossarysubentryfield}[6]{%
 10369 &
 10370 \glssubentryitem{##2}%
 10371 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 10372 \tabularnewline}%
 10373 }%

Backward compatible superraggedborder style.

10374 \compatglossarystyle{superraggedborder}{%
 10375 \csuse{@glscompstyle@superragged}%
 10376 }%

Backward compatible superraggedheader style.

10377 \compatglossarystyle{superraggedheader}{%
 10378 \csuse{@glscompstyle@superragged}%
 10379 }%

Backward compatible superraggedheaderborder style.

```
10380 \compatglossarystyle{superraggedheaderborder}{%
10381 \csuse{@glscompstyle@superragged}%
10382 }%
```

Backward compatible superragged3col style.

```
10383 \compatglossarystyle{superragged3col}{%
10384 \renewcommand*{\glossaryentryfield}[5]{%
10385 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10386 \renewcommand*{\glossarysubentryfield}[6]{%
10387 &
10388 \glssubentryitem{##2}%
10389 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10390 }%
```

Backward compatible superragged3colborder style.

```
10391 \compatglossarystyle{superragged3colborder}{%
10392 \csuse{@glscompstyle@superragged3col}%
10393 }%
```

Backward compatible superragged3colheader style.

```
10394 \compatglossarystyle{superragged3colheader}{%
10395 \csuse{@glscompstyle@superragged3col}%
10396 }%
```

Backward compatible superragged3colheaderborder style.

```
10397 \compatglossarystyle{superragged3colheaderborder}{%
10398 \csuse{@glscompstyle@superragged3col}%
10399 }%
```

Backward compatible altsuperragged4col style.

```
10400 \compatglossarystyle{altsuperragged4col}{%
10401 \renewcommand*{\glossaryentryfield}[5]{%
10402 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10403 \renewcommand*{\glossarysubentryfield}[6]{%
10404 &
10405 \glssubentryitem{##2}%
10406 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10407 }%
```

Backward compatible altsuperragged4colheader style.

```
10408 \compatglossarystyle{altsuperragged4colheader}{%
10409 \csuse{@glscompstyle@altsuperragged4col}%
10410 }%
```

Backward compatible altsuperragged4colborder style.

```
10411 \compatglossarystyle{altsuperragged4colborder}{%
10412 \csuse{@glscompstyle@altsuperragged4col}%
10413 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10414 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10415 \csuse{@glscompstyle@altsuperragged4col}%
10416 }%
```

Backward compatible super style.

```
10417 \compatglossarystyle{super}{%
10418   \renewcommand*{\glossaryentryfield}[5]{%
10419     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10420   \renewcommand*{\glossarysubentryfield}[6]{%
10421     &
10422     \glssubentryitem{##2}%
10423     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10424 }%
```

Backward compatible superborder style.

```
10425 \compatglossarystyle{superborder}{%
10426   \csuse{@glscompstyle@super}%
10427 }%
```

Backward compatible superheader style.

```
10428 \compatglossarystyle{superheader}{%
10429   \csuse{@glscompstyle@super}%
10430 }%
```

Backward compatible superheaderborder style.

```
10431 \compatglossarystyle{superheaderborder}{%
10432   \csuse{@glscompstyle@super}%
10433 }%
```

Backward compatible super3col style.

```
10434 \compatglossarystyle{super3col}{%
10435   \renewcommand*{\glossaryentryfield}[5]{%
10436     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10437   \renewcommand*{\glossarysubentryfield}[6]{%
10438     &
10439     \glssubentryitem{##2}%
10440     \glstarget{##2}{\strut}##4 & ##6\\}%
10441 }%
```

Backward compatible super3colborder style.

```
10442 \compatglossarystyle{super3colborder}{%
10443   \csuse{@glscompstyle@super3col}%
10444 }%
```

Backward compatible super3colheader style.

```
10445 \compatglossarystyle{super3colheader}{%
10446   \csuse{@glscompstyle@super3col}%
10447 }%
```

Backward compatible super3colheaderborder style.

```
10448 \compatglossarystyle{super3colheaderborder}{%
10449   \csuse{@glscompstyle@super3col}%
10450 }%
```

Backward compatible super4col style.

```
10451 \compatglossarystyle{super4col}{%
10452   \renewcommand*{\glossaryentryfield}[5]{%
10453     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10454   \renewcommand*{\glossarysubentryfield}[6]{%
10455     &
10456     \glssubentryitem{##2}%
10457     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10458 }%
```

Backward compatible super4colheader style.

```
10459 \compatglossarystyle{super4colheader}{%
10460   \csuse{@glscompstyle@super4col}%
10461 }%
```

Backward compatible super4colborder style.

```
10462 \compatglossarystyle{super4colborder}{%
10463   \csuse{@glscompstyle@super4col}%
10464 }%
```

Backward compatible super4colheaderborder style.

```
10465 \compatglossarystyle{super4colheaderborder}{%
10466   \csuse{@glscompstyle@super4col}%
10467 }%
```

Backward compatible altsuper4col style.

```
10468 \compatglossarystyle{altsuper4col}{%
10469   \csuse{@glscompstyle@super4col}%
10470 }%
```

Backward compatible altsuper4colheader style.

```
10471 \compatglossarystyle{altsuper4colheader}{%
10472   \csuse{@glscompstyle@super4col}%
10473 }%
```

Backward compatible altsuper4colborder style.

```
10474 \compatglossarystyle{altsuper4colborder}{%
10475   \csuse{@glscompstyle@super4col}%
10476 }%
```

Backward compatible altsuper4colheaderborder style.

```
10477 \compatglossarystyle{altsuper4colheaderborder}{%
10478   \csuse{@glscompstyle@super4col}%
10479 }%
```


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.

```
10480 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10481 \ProvidesPackage{glossaries-accsupp}[2019/12/06 v4.44 (NLCT)]
```

```
10482 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10483 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10484 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10485 \ifpackageloaded{glossaries-extra}
```

```
10486 {%
```

If the accsupp option was used, \@glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10487 \ifx\@glsxtr@doaccsupp\empty
```

```
10488 \GlossariesWarning{The ‘glossaries-accsupp’
```

```
10489 package has been loaded\MessageBreak
```

```
10490 after the ‘glossaries-extra’ package. This\MessageBreak
```

```
10491 can cause a failure to integrate both packages. \MessageBreak
```

```
10492 Either use the ‘accsupp’ option when you load\MessageBreak
```

```
10493 ‘glossaries-extra’ or load ‘glossaries-accsupp’\MessageBreak
```

```
10494 before loading ‘glossaries-extra’}%
```

```
10495 \fi
```

```
10496 }
```

```
10497 {}
```

tibleglossentry Override style compatibility macros:

```
10498 \def\compatibleglossentry#1#2{%
```

```
10499 \toks0{#2}%
```

```
10500 \protected@edef\do@glossentry{%
```

```
10501 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10502 {\noexpand\glsnamefont
```

```
10503 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```

10504     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10505     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10506     {\the\toks@}%
10507 }%
10508 \@do@glossentry
10509 }

```

lesubglossentry

```

10510 \def\compatiblesubglossentry#1#2#3{%
10511   \toks@{#3}%
10512   \protected@edef\@do@subglossentry{%
10513     \noexpand\accsuppglossarysubentryfield{\number#1}%
10514     {#2}%
10515     {\noexpand\glsnamefont
10516      {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}%
10517     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10518     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10519     {\the\toks@}%
10520   }%
10521   \@do@subglossentry
10522 }

```

Required packages:

```

10523 \RequirePackage{glossaries}
10524 \RequirePackage{accsupp}

```

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:

```

10525 \define@key{glossentry}{access}{%
10526   \def\@glo@access{#1}%
10527 }

```

textaccess The replacement text corresponding to the text key:

```

10528 \define@key{glossentry}{textaccess}{%
10529   \def\@glo@textaccess{#1}%
10530 }

```

firstaccess The replacement text corresponding to the first key:

```

10531 \define@key{glossentry}{firstaccess}{%
10532   \def\@glo@firstaccess{#1}%
10533 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10534 \define@key{glossentry}{pluralaccess}{%
10535   \def\@glo@pluralaccess{#1}%
10536 }

```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

10537 \define@key{glossentry}{firstpluralaccess}{%
10538   \def\@glo@firstpluralaccess{#1}%
10539 }

```

symbolaccess The replacement text corresponding to the symbol key:

```

10540 \define@key{glossentry}{symbolaccess}{%
10541   \def\@glo@symbolaccess{#1}%
10542 }

```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```

10543 \define@key{glossentry}{symbolpluralaccess}{%
10544   \def\@glo@symbolpluralaccess{#1}%
10545 }

```

descriptionaccess The replacement text corresponding to the description key:

```

10546 \define@key{glossentry}{descriptionaccess}{%
10547   \def\@glo@descaccess{#1}%
10548 }

```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```

10549 \define@key{glossentry}{descriptionpluralaccess}{%
10550   \def\@glo@descpluralaccess{#1}%
10551 }

```

shortaccess The replacement text corresponding to the short key:

```

10552 \define@key{glossentry}{shortaccess}{%
10553   \def\@glo@shortaccess{#1}%
10554 }

```

shortpluralaccess The replacement text corresponding to the shortplural key:

```

10555 \define@key{glossentry}{shortpluralaccess}{%
10556   \def\@glo@shortpluralaccess{#1}%
10557 }

```

longaccess The replacement text corresponding to the long key:

```

10558 \define@key{glossentry}{longaccess}{%
10559   \def\@glo@longaccess{#1}%
10560 }

```

longpluralaccess The replacement text corresponding to the longplural key:

```

10561 \define@key{glossentry}{longpluralaccess}{%
10562   \def\@glo@longpluralaccess{#1}%
10563 }

```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacccsup{inches}{in}}.

Append these new keys to \@gls@keymap:

```

10564 \appto\@gls@keymap{,%
10565   {access}{access},%
10566   {textaccess}{textaccess},%
10567   {firstaccess}{firstaccess},%
10568   {pluralaccess}{pluralaccess},%
10569   {firstpluralaccess}{firstpluralaccess},%
10570   {symbolaccess}{symbolaccess},%
10571   {symbolpluralaccess}{symbolpluralaccess},%
10572   {descaccess}{descaccess},%
10573   {descpluralaccess}{descpluralaccess},%
10574   {shortaccess}{shortaccess},%
10575   {shortpluralaccess}{shortpluralaccess},%
10576   {longaccess}{longaccess},%
10577   {longpluralaccess}{longpluralaccess}%
10578 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```

10579 \def\@gls@noaccess{\relax}
```

Add to the start hook (the access key is initialised to the value of the symbol key at the start for backwards compatibility):

```

10580 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10581 \renewcommand*{\@newglossaryentryprehook}{%
10582   \@gls@oldnewglossaryentryprehook
10583   \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```

10584   \def\@glo@textaccess{\@glo@access}%
10585   \def\@glo@firstaccess{\@glo@access}%
10586   \def\@glo@pluralaccess{\@glo@textaccess}%
10587   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10588   \def\@glo@symbolaccess{\relax}%
10589   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10590   \def\@glo@descaccess{\relax}%
10591   \def\@glo@descpluralaccess{\@glo@descaccess}%
10592   \def\@glo@shortaccess{\relax}%
10593   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10594   \def\@glo@longaccess{\relax}%
10595   \def\@glo@longpluralaccess{\@glo@longaccess}%
10596 }
```

Add to the end hook:

```

10597 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10598 \renewcommand*{\@newglossaryentryposthook}{%
10599   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```

10600 \expandafter
10601   \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10602     \@glo@access}%
10603 \expandafter
10604   \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10605     \@glo@textaccess}%
10606 \expandafter
10607   \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10608     \@glo@firstaccess}%
10609 \expandafter
10610   \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10611     \@glo@pluralaccess}%
10612 \expandafter
10613   \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10614     \@glo@firstpluralaccess}%
10615 \expandafter
10616   \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10617     \@glo@symbolaccess}%
10618 \expandafter
10619   \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10620     \@glo@symbolpluralaccess}%
10621 \expandafter
10622   \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10623     \@glo@descaccess}%
10624 \expandafter
10625   \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10626     \@glo@descpluralaccess}%
10627 \expandafter
10628   \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10629     \@glo@shortaccess}%
10630 \expandafter
10631   \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10632     \@glo@shortpluralaccess}%
10633 \expandafter
10634   \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10635     \@glo@longaccess}%
10636 \expandafter
10637   \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10638     \@glo@longpluralaccess}%
10639 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10640 \newcommand*{\glsentryaccess}[1]{%
10641   \@gls@entry@field{#1}{access}%
10642 }

```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
10643 \newcommand*{\glsentrytextaccess}[1]{%
10644   \@gls@entry@field{#1}{textaccess}%
10645 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10646 \newcommand*{\glsentryfirstaccess}[1]{%
10647   \@gls@entry@field{#1}{firstaccess}%
10648 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10649 \newcommand*{\glsentrypluralaccess}[1]{%
10650   \@gls@entry@field{#1}{pluralaccess}%
10651 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10652 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10653   \csname glo@#1@firstpluralaccess\endcsname
10654 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10655 \newcommand*{\glsentrysymbolaccess}[1]{%
10656   \@gls@entry@field{#1}{symbolaccess}%
10657 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10658 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10659   \@gls@entry@field{#1}{symbolpluralaccess}%
10660 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10661 \newcommand*{\glsentrydescaccess}[1]{%
10662   \@gls@entry@field{#1}{descaccess}%
10663 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10664 \newcommand*{\glsentrydescpluralaccess}[1]{%
10665   \@gls@entry@field{#1}{descaccess}%
10666 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10667 \newcommand*{\glsentryshortaccess}[1]{%
10668   \@gls@entry@field{#1}{shortaccess}%
10669 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10670 \newcommand*{\glsentryshortpluralaccess}[1]{%
10671   \@gls@entry@field{#1}{shortpluralaccess}%
10672 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10673 \newcommand*{\glsentrylongaccess}[1]{%
10674   \@gls@entry@field{#1}{longaccess}%
10675 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10676 \newcommand*{\glsentrylongpluralaccess}[1]{%
10677   \@gls@entry@field{#1}{longpluralaccess}%
10678 }
```

`\glsaccsupp` `\glsaccsupp{<replacement text>}{<text>}`

This can be redefined to use E or Alt instead of ActualText. (I don't have the software to test the E or Alt options.)

```
10679 \newcommand*{\glsaccsupp}[2]{%
10680   \BeginAccSupp{ActualText={#1}}#2\EndAccSupp{}}%
10681 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
10682 \newcommand*{\xglsaccsupp}[2]{%
10683   \protected@edef\@gls@replacementtext{#1}%
10684   \expandafter\xglsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10685 }
```

`@access@display`

```
10686 \newcommand*{\@gls@access@display}[2]{%
10687   \protected@edef\@glo@access{#2}%
10688   \ifx\@glo@access\@gls@noaccess
10689     #1%
10690   \else
10691     \xglsaccsupp{\@glo@access}{#1}%
10692   \fi
10693 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10694 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10695   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10696 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10697 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10698   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10699 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```

10700 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
10701   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10702 }

```

staccessdisplay As above but for the firstaccess replacement text.

```

10703 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
10704   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
10705 }

```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```

10706 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
10707   \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
10708 }

```

olaccessdisplay As above but for the symbolaccess replacement text.

```

10709 \DeclareRobustCommand*\glsymbolaccessdisplay}[2]{%
10710   \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
10711 }

```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```

10712 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
10713   \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}%
10714 }

```

onaccessdisplay As above but for the descriptionaccess replacement text.

```

10715 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10716   \@gls@access@display{#1}{\glsentrydescaccess{#2}}%
10717 }

```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

10718 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10719   \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}%
10720 }

```

rtaccessdisplay As above but for the shortaccess replacement text.

```

10721 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10722   \@gls@access@display{#1}{\glsentryshortaccess{#2}}%
10723 }

```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```

10724 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10725   \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}%
10726 }

```

ngaccessdisplay As above but for the longaccess replacement text.

```

10727 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10728   \@gls@access@display{#1}{\glsentrylongaccess{#2}}%
10729 }

```


alaccessdisplay As above but for the longpluralaccess replacement text.

```
10730 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10731   \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}%
10732 }
```

lsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
10733 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10734   \ifundefined{gls#1accessdisplay}%
10735   {%
10736     \PackageError{glossaries-accsupp}{No accessibility support
10737       for key ‘#1’}{}%
10738   }%
10739   {%
10740     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10741   }%
10742 }
```

efault@entryfmt Redefine the default entry format to use accessibility information

```
10743 \renewcommand*\@@gls@default@entryfmt}[2]{%
10744   \ifdefempty\glscustomtext
10745   {%
10746     \glsifplural
10747     {%
```

Plural form

```
10748     \glscapscase
10749     {%
```

Don't adjust case

```
10750     \ifglsused\glslabel
10751     {%
```

Subsequent use

```
10752     #2{\glspluralaccessdisplay
10753         {\glsentryplural{\glslabel}}{\glslabel}}%
10754     {\glsdescriptionpluralaccessdisplay
10755         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10756     {\glsymbolpluralaccessdisplay
10757         {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10758     {\glsinsert}%
10759   }%
10760   {%
```

First use

```
10761     #1{\glsfirstpluralaccessdisplay
10762         {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10763     {\glsdescriptionpluralaccessdisplay
10764         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10765     {\glsymbolpluralaccessdisplay
```

```

10766         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10767     {\glsinsert}%
10768 }%
10769 }%
10770 {%

```

Make first letter upper case

```

10771     \ifglsused\glslabel
10772     {%

```

Subsequent use.

```

10773     #2{\glspluralaccessdisplay
10774         {\Glsentryplural{\glslabel}}{\glslabel}}%
10775     {\glsdescriptionpluralaccessdisplay
10776         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10777     {\glsymbolpluralaccessdisplay
10778         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10779     {\glsinsert}%
10780 }%
10781 {%

```

First use

```

10782     #1{\glsfirstpluralaccessdisplay
10783         {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10784     {\glsdescriptionpluralaccessdisplay
10785         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10786     {\glsymbolpluralaccessdisplay
10787         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10788     {\glsinsert}%
10789 }%
10790 }%
10791 {%

```

Make all upper case

```

10792     \ifglsused\glslabel
10793     {%

```

Subsequent use

```

10794     \MakeUppercase{%
10795     #2{\glspluralaccessdisplay
10796         {\glsentryplural{\glslabel}}{\glslabel}}%
10797     {\glsdescriptionpluralaccessdisplay
10798         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10799     {\glsymbolpluralaccessdisplay
10800         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10801     {\glsinsert}}%
10802 }%
10803 {%

```

First use

```

10804     \MakeUppercase{%
10805     #1{\glsfirstpluralaccessdisplay

```

```

10806          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10807          {\glsdescriptionpluralaccessdisplay
10808          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10809          {\glssymbolpluralaccessdisplay
10810          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10811          {\glsinsert}}}%
10812      }%
10813  }%
10814 }%
10815 {%

```

Singular form

```

10816      \glscapscase
10817      {%

```

Don't adjust case

```

10818      \ifglused\glslabel
10819      {%

```

Subsequent use

```

10820      #2{\glstextaccessdisplay
10821          {\glsentrytext{\glslabel}}{\glslabel}}%
10822          {\glsdescriptionaccessdisplay
10823          {\glsentrydesc{\glslabel}}{\glslabel}}%
10824          {\glssymbolaccessdisplay
10825          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10826          {\glsinsert}}%
10827      }%
10828      {%

```

First use

```

10829      #1{\glsfirstaccessdisplay
10830          {\glsentryfirst{\glslabel}}{\glslabel}}%
10831          {\glsdescriptionaccessdisplay
10832          {\glsentrydesc{\glslabel}}{\glslabel}}%
10833          {\glssymbolaccessdisplay
10834          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10835          {\glsinsert}}%
10836      }%
10837      }%
10838      {%

```

Make first letter upper case

```

10839      \ifglused\glslabel
10840      {%

```

Subsequent use

```

10841      #2{\glstextaccessdisplay
10842          {\Glsentrytext{\glslabel}}{\glslabel}}%
10843          {\glsdescriptionaccessdisplay
10844          {\Glsentrydesc{\glslabel}}{\glslabel}}%
10845          {\glssymbolaccessdisplay

```

```

10846          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10847      {\glsinsert}%
10848  }%
10849  {%

```

First use

```

10850      #1{\glsfirstaccessdisplay
10851          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10852      {\glsdescriptionaccessdisplay
10853          {\glsentrydesc{\glslabel}}{\glslabel}}%
10854      {\glsymbolaccessdisplay
10855          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10856      {\glsinsert}%
10857  }%
10858  }%
10859  {%

```

Make all upper case

```

10860      \ifglsused\glslabel
10861      {%

```

Subsequent use

```

10862      \MakeUppercase{%
10863      #2{\glsfirstaccessdisplay
10864          {\glsentrytext{\glslabel}}{\glslabel}}%
10865      {\glsdescriptionaccessdisplay
10866          {\glsentrydesc{\glslabel}}{\glslabel}}%
10867      {\glsymbolaccessdisplay
10868          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10869      {\glsinsert}}%
10870  }%
10871  {%

```

First use

```

10872      \MakeUppercase{%
10873      #1{\glsfirstaccessdisplay
10874          {\glsentryfirst{\glslabel}}{\glslabel}}%
10875      {\glsdescriptionaccessdisplay
10876          {\glsentrydesc{\glslabel}}{\glslabel}}%
10877      {\glsymbolaccessdisplay
10878          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10879      {\glsinsert}}%
10880  }%
10881  }%
10882  }%
10883  }%
10884  {%

```

Custom text provided in \glsdisp

```

10885      \ifglsused{\glslabel}%
10886      {%

```

Subsequent use

```

10887      #2{\glscustomtext}%
10888      {\glsdescriptionaccessdisplay
10889       {\glsentrydesc{\glslabel}}{\glslabel}}%
10890      {\glssymbolaccessdisplay
10891       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10892      {\glsinsert}%
10893    }%
10894    {%

```

First use

```

10895      #1{\glscustomtext}%
10896      {\glsdescriptionaccessdisplay
10897       {\glsentrydesc{\glslabel}}{\glslabel}}%
10898      {\glssymbolaccessdisplay
10899       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10900      {\glsinsert}%
10901    }%
10902  }%
10903 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10904 \renewcommand*{\glsgenentryfmt}{%
10905   \ifdefempty\glscustomtext
10906     {%
10907       \glsifplural
10908     {%

```

Plural form

```

10909       \glscapscase
10910     {%

```

Don't adjust case

```

10911       \ifglsused\glslabel
10912     {%

```

Subsequent use

```

10913       \glspluralaccessdisplay
10914       {\glsentryplural{\glslabel}}{\glslabel}%
10915       \glsinsert
10916     }%
10917     {%

```

First use

```

10918       \glsfirstpluralaccessdisplay
10919       {\glsentryfirstplural{\glslabel}}{\glslabel}%
10920       \glsinsert
10921     }%
10922   }%
10923   {%

```

Make first letter upper case

10924 \ifglused\glslabel
10925 {%

Subsequent use.

10926 \glspluralaccessdisplay
10927 {\Glentryplural{\glslabel}}{\glslabel}%
10928 \glinsert
10929 }%
10930 {%

First use

10931 \glfirstpluralaccessdisplay
10932 {\Glentryfirstplural{\glslabel}}{\glslabel}%
10933 \glinsert
10934 }%
10935 }%
10936 {%

Make all upper case

10937 \ifglused\glslabel
10938 {%

Subsequent use

10939 \glspluralaccessdisplay
10940 {\mfirstucMakeUppercase{\glentryplural{\glslabel}}}%
10941 {\glslabel}%
10942 \mfirstucMakeUppercase{\glinsert}%
10943 }%
10944 {%

First use

10945 \glfirstpluralacessdisplay
10946 {\mfirstucMakeUppercase{\glentryfirstplural{\glslabel}}}%
10947 {\glslabel}%
10948 \mfirstucMakeUppercase{\glinsert}%
10949 }%
10950 }%
10951 }%
10952 {%

Singular form

10953 \glscapscase
10954 {%

Don't adjust case

10955 \ifglused\glslabel
10956 {%

Subsequent use

10957 \glstextaccessdisplay{\glentrytext{\glslabel}}{\glslabel}%
10958 \glinsert

10959 }%
 10960 {%

First use

10961 \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
 10962 \glsinsert
 10963 }%
 10964 }%
 10965 {%

Make first letter upper case

10966 \ifglsused\glslabel
 10967 {%

Subsequent use

10968 \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
 10969 \glsinsert
 10970 }%
 10971 {%

First use

10972 \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
 10973 \glsinsert
 10974 }%
 10975 }%
 10976 {%

Make all upper case

10977 \ifglsused\glslabel
 10978 {%

Subsequent use

10979 \glstextaccessdisplay
 10980 {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
 10981 \mfirstucMakeUppercase{\glsinsert}%
 10982 }%
 10983 {%

First use

10984 \glsfirstaccessdisplay
 10985 {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
 10986 \mfirstucMakeUppercase{\glsinsert}%
 10987 }%
 10988 }%
 10989 }%
 10990 }%
 10991 {%

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.

10992 \glscustomtext\glsinsert
 10993 }%
 10994 }

`\glsgenacfmt` Redefine to include accessibility information.

```
10995 \renewcommand*{\glsgenacfmt}{%
10996   \ifdefempty\glscustomtext
10997     {%
10998       \ifglused\glslabel
10999       {%
```

Subsequent use:

```
11000   \glsifplural
11001   {%
```

Subsequent plural form:

```
11002   \glscapscase
11003   {%
```

Subsequent plural form, don't adjust case:

```
11004   \acronymfont
11005   {\glsshortpluralaccessdisplay
11006     {\glentryshortpl{\glslabel}}{\glslabel}}%
11007   \glsinsert
11008   }%
11009   {%
```

Subsequent plural form, make first letter upper case:

```
11010   \acronymfont
11011   {\glsshortpluralaccessdisplay
11012     {\Glsentryshortpl{\glslabel}}{\glslabel}}%
11013   \glsinsert
11014   }%
11015   {%
```

Subsequent plural form, all caps:

```
11016   \mfirstucMakeUppercase
11017   {\acronymfont
11018     {\glsshortpluralaccessdisplay
11019       {\glentryshortpl{\glslabel}}{\glslabel}}%
11020     \glsinsert}%
11021   }%
11022   }%
11023   {%
```

Subsequent singular form

```
11024   \glscapscase
11025   {%
```

Subsequent singular form, don't adjust case:

```
11026   \acronymfont
11027   {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
11028   \glsinsert
11029   }%
11030   {%
```


Subsequent singular form, make first letter upper case:

```
11031      \acronymfont
11032      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
11033      \glsinsert
11034      }%
11035      {%
```

Subsequent singular form, all caps:

```
11036      \mfirstucMakeUppercase
11037      {\acronymfont{%
11038      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
11039      \glsinsert}%
11040      }%
11041      }%
11042      }%
11043      {%
```

First use:

```
11044      \glsifplural
11045      {%
```

First use plural form:

```
11046      \glscapscase
11047      {%
```

First use plural form, don't adjust case:

```
11048      \genplacrfullformat{\glslabel}{\glsinsert}%
11049      }%
11050      {%
```

First use plural form, make first letter upper case:

```
11051      \Genplacrfullformat{\glslabel}{\glsinsert}%
11052      }%
11053      {%
```

First use plural form, all caps:

```
11054      \mfirstucMakeUppercase
11055      {\genplacrfullformat{\glslabel}{\glsinsert}}%
11056      }%
11057      }%
11058      {%
```

First use singular form

```
11059      \glscapscase
11060      {%
```

First use singular form, don't adjust case:

```
11061      \genacrfullformat{\glslabel}{\glsinsert}%
11062      }%
11063      {%
```

First use singular form, make first letter upper case:

```
11064      \Genacrfullformat{\glslabel}{\glsinsert}%
11065      }%
11066      {%
```

First use singular form, all caps:

```
11067      \mfirstucMakeUppercase
11068      {\genacrfullformat{\glslabel}{\glsinsert}}%
11069      }%
11070      }%
11071      }%
11072      }%
11073      {%
```

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.

```
11074      \glscustomtext
11075      }%
11076 }
```

enacrfullformat Redefine to include accessibility information.

```
11077 \renewcommand*{\genacrfullformat}[2]{%
11078   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
11079   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
11080 }
```

enacrfullformat Redefine to include accessibility information.

```
11081 \renewcommand*{\Genacrfullformat}[2]{%
11082   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
11083   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
11084 }
```

placrfullformat Redefine to include accessibility information.

```
11085 \renewcommand*{\genplacrfullformat}[2]{%
11086   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
11087   (\glsshortpluralaccessdisplay
11088     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
11089 }
```

placrfullformat Redefine to include accessibility information.

```
11090 \renewcommand*{\Genplacrfullformat}[2]{%
11091   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
11092   (\glsshortpluralaccessdisplay
11093     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
11094 }
```

\@acrshort

```
11095 \def\@acrshort#1#2[#3]{%
11096   \glsdoifexists{#2}%
```

```

11097 {%
11098   \let\do@gls@link@checkfirsthyper\relax

11099   \let\glsifplural\@secondoftwo
11100   \let\glsupcase\@firstofthree
11101   \let\glsinsert\@empty
11102   \def\glscustomtext{%
11103     \acronymfont{\glsshortaccessdisplay{\glsentryshort{#2}}{#2}}#3%
11104   }%

   Call \@gls@link
11105   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11106   }%

11107   \glspostlinkhook
11108 }

```

\@Acrshort

```

11109 \def\@Acrshort#1#2[#3]{%
11110   \glsdoifexists{#2}%
11111   {%
11112     \let\do@gls@link@checkfirsthyper\relax

11113     \let\glsifplural\@secondoftwo
11114     \let\glsupcase\@secondofthree
11115     \let\glsinsert\@empty
11116     \def\glscustomtext{%
11117       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
11118     }%

     Call \@gls@link
11119     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11120     }%

11121     \glspostlinkhook
11122 }

```

\@ACRshort

```

11123 \def\@ACRshort#1#2[#3]{%
11124   \glsdoifexists{#2}%
11125   {%
11126     \let\do@gls@link@checkfirsthyper\relax

11127     \let\glsifplural\@secondoftwo
11128     \let\glsupcase\@thirdofthree
11129     \let\glsinsert\@empty
11130     \def\glscustomtext{%
11131       \acronymfont{\glsshortaccessdisplay
11132         {\MakeUppercase{\glsentryshort{#2}}}{#2}}#3%
11133     }%

```

```

    Call \@gls@link
11134   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11135   }%

11136   \glspostlinkhook
11137 }

```

\@acrlong

```

11138 \def\@acrlong#1#2[#3]{%
11139   \glsdoifexists{#2}%
11140   {%
11141     \let\do@gls@link@checkfirsthyper\relax

11142     \let\glsifplural\@secondoftwo
11143     \let\glscapscase\@firstofthree
11144     \let\glsinsert\@empty
11145     \def\glscustomtext{%
11146       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
11147     }%

```

```

    Call \@gls@link
11148   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11149   }%

11150   \glspostlinkhook
11151 }

```

\@Acrlong

```

11152 \def\@Acrlong#1#2[#3]{%
11153   \glsdoifexists{#2}%
11154   {%
11155     \let\do@gls@link@checkfirsthyper\relax

11156     \let\glsifplural\@secondoftwo
11157     \let\glscapscase\@firstofthree
11158     \let\glsinsert\@empty
11159     \def\glscustomtext{%
11160       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
11161     }%

```

```

    Call \@gls@link
11162   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11163   }%

11164   \glspostlinkhook
11165 }

```

\@ACRlong

```

11166 \def\@ACRlong#1#2[#3]{%
11167   \glsdoifexists{#2}%
11168   {%
11169     \let\do@gls@link@checkfirsthyper\relax

```

```

11170 \let\glsifplural\@secondoftwo
11171 \let\glsifscapscase\@firstofthree
11172 \let\glsinsert\@empty
11173 \def\glscustomtext{%
11174     \acronymfont{\glslongaccessdisplay{%
11175         \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
11176     }%

    Call \@gls@link
11177     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11178 }%

11179 \glspostlinkhook
11180 }

```

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.

```

11181 \renewcommand*{\glossentryname}[1]{%
11182     \glsdoifexists{#1}%
11183     {%
11184         \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
11185     }%
11186 }

11187 \renewcommand*{\glossentrydesc}[1]{%
11188     \glsdoifexists{#1}%
11189     {%
11190         \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
11191     }%
11192 }

11193 \renewcommand*{\glossentrydesc}[1]{%
11194     \glsdoifexists{#1}%
11195     {%
11196         \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
11197     }%
11198 }

11199 \renewcommand*{\Glossentrydesc}[1]{%
11200     \glsdoifexists{#1}%
11201     {%
11202         \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
11203     }%
11204 }

```

```

11205 \renewcommand*{\glossentrysymbol}[1]{%
11206   \glsdoifexists{#1}%
11207   {%
11208     \glssymbolaccessdisplay{\glentrysymbol{#1}}{#1}%
11209   }%
11210 }

11211 \renewcommand*{\Glossentrysymbol}[1]{%
11212   \glsdoifexists{#1}%
11213   {%
11214     \glssymbolaccessdisplay{\Glentrysymbol{#1}}{#1}%
11215   }%
11216 }

```

ssaryentryfield

```

11217 \newcommand*{\accsuppglossaryentryfield}[5]{%
11218   \glossaryentryfield{#1}%
11219   {\glsnameaccessdisplay{#2}{#1}}%
11220   {\glsdescriptionaccessdisplay{#3}{#1}}%
11221   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
11222 }

```

rysubentryfield

```

11223 \newcommand*{\accsuppglossarysubentryfield}[6]{%
11224   \glossarysubentryfield{#1}{#2}%
11225   {\glsnameaccessdisplay{#3}{#2}}%
11226   {\glsdescriptionaccessdisplay{#4}{#2}}%
11227   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
11228 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

11229 \renewacronymstyle{long-short}%
11230 {%

```

Check for long form in case this is a mixed glossary.

```

11231   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11232 }%
11233 {%
11234   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11235   \renewcommand*{\genacrfullformat}[2]{%
11236     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
11237     (\glsshortaccessdisplay
11238       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11239   }%
11240   \renewcommand*{\Genacrfullformat}[2]{%

```

```

11241 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11242 (\glsshortaccessdisplay
11243   {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
11244 }%
11245 \renewcommand*{\genplacrfullformat}[2]{%
11246   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11247   (\glsshortpluralaccessdisplay
11248     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11249 }%
11250 \renewcommand*{\Genplacrfullformat}[2]{%
11251   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11252   (\glsshortpluralaccessdisplay
11253     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11254 }%
11255 \renewcommand*{\acronymentry}[1]{%
11256   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
11257 \renewcommand*{\acronymsort}[2]{##1}%
11258 \renewcommand*{\acronymfont}[1]{##1}%
11259 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11260 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11261 }

```

short-long (*short*) (*long*) acronym style.

```

11262 \renewacronymstyle{short-long}%
11263 {%

```

Check for long form in case this is a mixed glossary.

```

11264 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11265 }%
11266 {%
11267   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11268   \renewcommand*{\genacrfullformat}[2]{%
11269     \glsshortaccessdisplay
11270     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
11271     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11272   }%
11273   \renewcommand*{\Genacrfullformat}[2]{%
11274     \glsshortaccessdisplay
11275     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
11276     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11277   }%
11278   \renewcommand*{\genplacrfullformat}[2]{%
11279     \glsshortpluralaccessdisplay
11280     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
11281     (\glslongpluralaccessdisplay
11282       {\glsentrylongpl{##1}}{##1}})%
11283   }%
11284   \renewcommand*{\Genplacrfullformat}[2]{%
11285     \glsshortpluralaccessdisplay
11286     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

11287 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11288 }%
11289 \renewcommand*{\acronymentry}[1]{%
11290   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11291 \renewcommand*{\acronymsort}[2]{##1}%
11292 \renewcommand*{\acronymfont}[1]{##1}%
11293 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11294 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11295 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

11296 \renewacronymstyle{long-short-desc}%
11297 {%
11298   \GlsUseAcrEntryDispStyle{long-short}%
11299 }%
11300 {%
11301   \GlsUseAcrStyleDefs{long-short}%
11302   \renewcommand*{\GenericAcronymFields}{}%
11303   \renewcommand*{\acronymsort}[2]{##2}%
11304   \renewcommand*{\acronymentry}[1]{%
11305     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11306     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11307 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11308 \renewacronymstyle{long-sc-short-desc}%
11309 {%
11310   \GlsUseAcrEntryDispStyle{long-sc-short}%
11311 }%
11312 {%
11313   \GlsUseAcrStyleDefs{long-sc-short}%
11314   \renewcommand*{\GenericAcronymFields}{}%
11315   \renewcommand*{\acronymsort}[2]{##2}%
11316   \renewcommand*{\acronymentry}[1]{%
11317     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11318     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11319 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11320 \renewacronymstyle{long-sm-short-desc}%
11321 {%
11322   \GlsUseAcrEntryDispStyle{long-sm-short}%
11323 }%
11324 {%
11325   \GlsUseAcrStyleDefs{long-sm-short}%
11326   \renewcommand*{\GenericAcronymFields}{}%

```



```

11327 \renewcommand*{\acronymsort}[2]{##2}%
11328 \renewcommand*{\acronymentry}[1]{%
11329     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11330     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11331 }

```

short-long-desc *(short)* (*long*) acronym style that has an accompanying description (which the user needs to supply).

```

11332 \renewacronymstyle{short-long-desc}%
11333 {%
11334     \GlsUseAcrEntryDispStyle{short-long}%
11335 }%
11336 {%
11337     \GlsUseAcrStyleDefs{short-long}%
11338     \renewcommand*{\GenericAcronymFields}{}%
11339     \renewcommand*{\acronymsort}[2]{##2}%
11340     \renewcommand*{\acronymentry}[1]{%
11341         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11342         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11343 }

```

short-long-desc *long* (\textsc{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11344 \renewacronymstyle{sc-short-long-desc}%
11345 {%
11346     \GlsUseAcrEntryDispStyle{sc-short-long}%
11347 }%
11348 {%
11349     \GlsUseAcrStyleDefs{sc-short-long}%
11350     \renewcommand*{\GenericAcronymFields}{}%
11351     \renewcommand*{\acronymsort}[2]{##2}%
11352     \renewcommand*{\acronymentry}[1]{%
11353         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11354         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11355 }

```

short-long-desc *long* (\textsmaller{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11356 \renewacronymstyle{sm-short-long-desc}%
11357 {%
11358     \GlsUseAcrEntryDispStyle{sm-short-long}%
11359 }%
11360 {%
11361     \GlsUseAcrStyleDefs{sm-short-long}%
11362     \renewcommand*{\GenericAcronymFields}{}%
11363     \renewcommand*{\acronymsort}[2]{##2}%
11364     \renewcommand*{\acronymentry}[1]{%
11365         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11366         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

11367 }

dua *<long>* only acronym style.

11368 \renewacronymstyle{dua}%
11369 {%

Check for long form in case this is a mixed glossary.

11370 \ifdefempty\glscustomtext
11371 {%
11372 \ifglshaslong{\glslabel}%
11373 {%
11374 \glsifplural
11375 {%

Plural form:

11376 \glscapscase
11377 {%

Plural form, don't adjust case:

11378 \glslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%
11379 \glsinsert
11380 }%
11381 {%

Plural form, make first letter upper case:

11382 \glslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%
11383 \glsinsert
11384 }%
11385 {%

Plural form, all caps:

11386 \glslongpluralaccessdisplay
11387 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%
11388 \mfirstucMakeUppercase{\glsinsert}%
11389 }%
11390 }%
11391 {%

Singular form

11392 \glscapscase
11393 {%

Singular form, don't adjust case:

11394 \glslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert
11395 }%
11396 {%

Subsequent singular form, make first letter upper case:

11397 \glslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert
11398 }%
11399 {%

Subsequent singular form, all caps:

```

11400      \glslongaccessdisplay
11401      {\mfirstucMakeUppercase
11402        {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11403      \mfirstucMakeUppercase{\glsinsert}%
11404      }%
11405      }%
11406      }%
11407      {%

```

Not an acronym:

```

11408      \glsgenentryfmt
11409      }%
11410      }%
11411      {\glscustomtext\glsinsert}%
11412      }%
11413      {%
11414      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11415      \renewcommand*{\acrfullfmt}[3]{%
11416        \glslink[##1]{##2}{%
11417          \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11418          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11419      \renewcommand*{\Acrfullfmt}[3]{%
11420        \glslink[##1]{##2}{%
11421          \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11422          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11423      \renewcommand*{\ACRfullfmt}[3]{%
11424        \glslink[##1]{##2}{%
11425          \glslongaccessdisplay
11426          {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11427          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11428      \renewcommand*{\acrfullplfmt}[3]{%
11429        \glslink[##1]{##2}{%
11430          \glslongpluralaccessdisplay
11431          {\glsentrylongpl{##2}}{##2}##3\space
11432          (\glsshortpluralaccessdisplay
11433          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11434      \renewcommand*{\ACRfullplfmt}[3]{%
11435        \glslink[##1]{##2}{%
11436          \glslongpluralaccessdisplay
11437          {\Glsentrylongpl{##2}}{##2}##3\space
11438          (\glsshortpluralaccessdisplay
11439          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11440      \renewcommand*{\ACRfullplplfmt}[3]{%
11441        \glslink[##1]{##2}{%
11442          \glslongpluralaccessdisplay
11443          {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11444          (\glsshortpluralaccessdisplay
11445          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11446      \renewcommand*{\glsentryfull}[1]{%

```

```

11447 \glslongaccessdisplay{\glsentrylong{##1}}\space
11448 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11449 }%
11450 \renewcommand*{\Glsentryfull}[1]{%
11451 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11452 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11453 }%
11454 \renewcommand*{\glsentryfullpl}[1]{%
11455 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11456 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11457 }%
11458 \renewcommand*{\Glsentryfullpl}[1]{%
11459 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11460 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11461 }%
11462 \renewcommand*{\acronymentry}[1]{%
11463 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11464 \renewcommand*{\acronymsort}[2]{##1}%
11465 \renewcommand*{\acronymfont}[1]{##1}%
11466 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11467 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11468 \renewacronymstyle{dua-desc}%
11469 {%
11470 \GlsUseAcrEntryDispStyle{dua}%
11471 }%
11472 {%
11473 \GlsUseAcrStyleDefs{dua}%
11474 \renewcommand*{\GenericAcronymFields}{}%
11475 \renewcommand*{\acronymentry}[1]{%
11476 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11477 \renewcommand*{\acronymsort}[2]{##2}%
11478 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11479 \renewacronymstyle{footnote}%
11480 {%
11481 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11482 }%
11483 {%
11484 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11485 \glshyperfirstfalse
11486 \renewcommand*{\genacrfullformat}[2]{%
11487 \glsshortaccessdisplay
11488 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

11489 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11490 }%
11491 \renewcommand*{\Genacrfullformat}[2]{%
11492 \glsshortaccessdisplay
11493   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11494 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11495 }%
11496 \renewcommand*{\genplacrfullformat}[2]{%
11497 \glsshortpluralaccessdisplay
11498   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
11499 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11500 }%
11501 \renewcommand*{\Genplacrfullformat}[2]{%
11502 \glsshortpluralaccessdisplay
11503   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11504 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11505 }%
11506 \renewcommand*{\acronymentry}[1]{%
11507 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11508 \renewcommand*{\acronymsort}[2]{##1}%
11509 \renewcommand*{\acronymfont}[1]{##1}%
11510 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11511 \renewcommand*{\acrfullfmt}[3]{%
11512 \glslink{##1}{##2}{%
11513 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11514 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11515 \renewcommand*{\Acrfullfmt}[3]{%
11516 \glslink{##1}{##2}{%
11517 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11518 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11519 \renewcommand*{\ACRfullfmt}[3]{%
11520 \glslink{##1}{##2}{%
11521 \glsshortaccessdisplay
11522   {\mfirstucMakeUppercase
11523   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11524   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11525 \renewcommand*{\acrfullplfmt}[3]{%
11526 \glslink{##1}{##2}{%
11527 \glsshortpluralaccessdisplay
11528   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11529   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11530 \renewcommand*{\Acrfullplfmt}[3]{%
11531 \glslink{##1}{##2}{%
11532 \glsshortpluralaccessdisplay
11533   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11534   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11535 \renewcommand*{\ACRfullplfmt}[3]{%
11536 \glslink{##1}{##2}{%

```

```

11537 \glsshortpluralaccessdisplay
11538 {\mfirstucMakeUppercase
11539 {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
11540 (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

11541 \renewcommand*{\glentryfull}[1]{%
11542 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
11543 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11544 \renewcommand*{\Glsentryfull}[1]{%
11545 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
11546 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11547 \renewcommand*{\glentryfullpl}[1]{%
11548 \glsshortpluralaccessdisplay
11549 {\acronymfont{\glentryshortpl{##1}}{##1}\space
11550 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11551 \renewcommand*{\Glsentryfullpl}[1]{%
11552 \glsshortpluralaccessdisplay
11553 {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
11554 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11555 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11556 \renewacronymstyle{footnote-sc}%
11557 {%
11558 \GlsUseAcrEntryDispStyle{footnote}%
11559 }%
11560 {%
11561 \GlsUseAcrStyleDefs{footnote}%
11562 \renewcommand{\acronymentry}[1]{%
11563 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11564 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11565 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11566 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11567 \renewacronymstyle{footnote-sm}%
11568 {%
11569 \GlsUseAcrEntryDispStyle{footnote}%
11570 }%
11571 {%
11572 \GlsUseAcrStyleDefs{footnote}%
11573 \renewcommand{\acronymentry}[1]{%
11574 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11575 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11576 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11577 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11578 \renewacronymstyle{footnote-desc}%
11579 {%
11580   \GlsUseAcrEntryDisplayStyle{footnote}%
11581 }%
11582 {%
11583   \GlsUseAcrStyleDefs{footnote}%
11584   \renewcommand*{\GenericAcronymFields}{}%
11585   \renewcommand*{\acronymsort}[2]{##2}%
11586   \renewcommand*{\acronymentry}[1]{%
11587     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11588     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11589 }

```

ootnote-sc-desc \textsc{\<short>}\footnote{\<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11590 \renewacronymstyle{footnote-sc-desc}%
11591 {%
11592   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11593 }%
11594 {%
11595   \GlsUseAcrStyleDefs{footnote-sc}%
11596   \renewcommand*{\GenericAcronymFields}{}%
11597   \renewcommand*{\acronymsort}[2]{##2}%
11598   \renewcommand*{\acronymentry}[1]{%
11599     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11600     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11601 }

```

ootnote-sm-desc \textsmaller{\<short>}\footnote{\<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11602 \renewacronymstyle{footnote-sm-desc}%
11603 {%
11604   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11605 }%
11606 {%
11607   \GlsUseAcrStyleDefs{footnote-sm}%
11608   \renewcommand*{\GenericAcronymFields}{}%
11609   \renewcommand*{\acronymsort}[2]{##2}%
11610   \renewcommand*{\acronymentry}[1]{%
11611     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11612     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11613 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11614 \renewcommand*{\newacronymhook}{%
11615   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11616     \the\glskeylisttok}%
11617   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11618 }

1tNewAcronymDef Modify default style to use access text:

```
11619 \renewcommand*{\DefaultNewAcronymDef}{%
11620   \edef\@do@newglossaryentry{%
11621     \noexpand\newglossaryentry{\the\glslabeltok}%
11622     {%
11623       type=\acronymtype,%
11624       name={\the\glsshorttok},%
11625       description={\the\glslongtok},%
11626       descriptionaccess=\relax,
11627       text={\the\glsshorttok},%
11628       access={\noexpand\@glo@textaccess},%
11629       sort={\the\glsshorttok},%
11630       short={\the\glsshorttok},%
11631       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11632       shortaccess={\the\glslongtok},%
11633       long={\the\glslongtok},%
11634       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11635       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11636       first={\noexpand\glslongaccessdisplay
11637         {\the\glslongtok}{\the\glslabeltok}\space
11638         (\noexpand\glsshortaccessdisplay
11639           {\the\glsshorttok}{\the\glslabeltok})},%
11640       plural={\the\glsshorttok\acrpluralsuffix},%
11641       firstplural={\noexpand\glslongpluralaccessdisplay
11642         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11643         (\noexpand\glsshortpluralaccessdisplay
11644           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11645       firstaccess=\relax,
11646       firstpluralaccess=\relax,
11647       textaccess={\noexpand\@glo@shortaccess},%
11648       \the\glskeylisttok
11649     }%
11650   }%
11651   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11652   \let\@org@gls@assign@plural\gls@assign@plural
11653   \let\@org@gls@assign@descplural\gls@assign@descplural
11654   \def\gls@assign@firstpl##1##2{%
11655     \@gls@expand@field{##1}{firstpl}{##2}%
11656   }%
11657   \def\gls@assign@plural##1##2{%
11658     \@gls@expand@field{##1}{plural}{##2}%
11659   }%
11660   \def\gls@assign@descplural##1##2{%
11661     \@gls@expand@field{##1}{descplural}{##2}%
11662   }%
11663   \@do@newglossaryentry
11664   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```



```

11665 \let\gls@assign@plural\@org@gls@assign@plural
11666 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11667 }

```

teNewAcronymDef

```

11668 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11669   \edef\@do@newglossaryentry{%
11670     \noexpand\newglossaryentry{\the\glslabeltok}%
11671     {%
11672       type=\acronymtype,%
11673       name={\noexpand\acronymfont{\the\glsshorttok}},%
11674       sort={\the\glsshorttok},%
11675       text={\the\glsshorttok},%
11676       short={\the\glsshorttok},%
11677       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11678       shortaccess={\the\glslongtok},%
11679       long={\the\glslongtok},%
11680       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11681       access={\noexpand\@glo@textaccess},%
11682       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11683       symbol={\the\glslongtok},%
11684       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11685       firstpluralaccess=\relax,
11686       textaccess={\noexpand\@glo@shortaccess},%
11687       \the\glskeylisttok
11688     }%
11689   }%
11690   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11691   \let\@org@gls@assign@plural\gls@assign@plural
11692   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11693   \def\gls@assign@firstpl##1##2{%
11694     \@@gls@expand@field{##1}{firstpl}{##2}%
11695   }%
11696   \def\gls@assign@plural##1##2{%
11697     \@@gls@expand@field{##1}{plural}{##2}%
11698   }%
11699   \def\gls@assign@symbolplural##1##2{%
11700     \@@gls@expand@field{##1}{symbolplural}{##2}%
11701   }%
11702   \@do@newglossaryentry
11703   \let\gls@assign@plural\@org@gls@assign@plural
11704   \let\gls@assign@firstpl\@org@gls@assign@firstpl
11705   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11706 }

```

onNewAcronymDef

```

11707 \renewcommand*{\DescriptionNewAcronymDef}{%
11708   \edef\@do@newglossaryentry{%
11709     \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11710  {%
11711      type=\acronymtype,%
11712      name={\noexpand
11713          \acrnameformat{\the\glssshorttok}{\the\glslongtok}},%
11714      access={\noexpand\@glo@textaccess},%
11715      sort={\the\glssshorttok},%
11716      short={\the\glssshorttok},%
11717      shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11718      shortaccess={\the\glslongtok},%
11719      long={\the\glslongtok},%
11720      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11721      first={\the\glslongtok},%
11722      firstaccess=\relax,
11723      firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11724      text={\the\glssshorttok},%
11725      textaccess={\the\glslongtok},%
11726      plural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11727      symbol={\noexpand\@glo@text},%
11728      symbolaccess={\noexpand\@glo@textaccess},%
11729      symbolplural={\noexpand\@glo@plural},%
11730      firstpluralaccess=\relax,
11731      textaccess={\noexpand\@glo@shortaccess},%
11732      \the\glskeylisttok}%
11733  }%
11734  \let\@org@gls@assign@firstpl\gls@assign@firstpl
11735  \let\@org@gls@assign@plural\gls@assign@plural
11736  \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11737  \def\gls@assign@firstpl##1##2{%
11738      \@gls@expand@field{##1}{firstpl}{##2}%
11739  }%
11740  \def\gls@assign@plural##1##2{%
11741      \@gls@expand@field{##1}{plural}{##2}%
11742  }%
11743  \def\gls@assign@symbolplural##1##2{%
11744      \@gls@expand@field{##1}{symbolplural}{##2}%
11745  }%
11746  \do@newglossaryentry
11747  \let\gls@assign@firstpl\@org@gls@assign@firstpl
11748  \let\gls@assign@plural\@org@gls@assign@plural
11749  \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11750 }

```

teNewAcronymDef

```

11751 \renewcommand*{\FootnoteNewAcronymDef}{%
11752     \edef\@do@newglossaryentry{%
11753         \noexpand\newglossaryentry{\the\glslabeltok}%
11754     }%
11755     type=\acronymtype,%
11756     name={\noexpand\acronymfont{\the\glssshorttok}},%

```

```

11757     sort={\the\glsshorttok},%
11758     text={\the\glsshorttok},%
11759     textaccess={\the\glslongtok},%
11760     access={\noexpand\@glo@textaccess},%
11761     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11762     short={\the\glsshorttok},%
11763     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11764     long={\the\glslongtok},%
11765     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11766     description={\the\glslongtok},%
11767     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11768     \the\glskeylisttok
11769   }%
11770 }%
11771 \let\@org@gls@assign@plural\gls@assign@plural
11772 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11773 \let\@org@gls@assign@descplural\gls@assign@descplural
11774 \def\gls@assign@firstpl##1##2{%
11775   \@gls@expand@field{##1}{firstpl}{##2}%
11776 }%
11777 \def\gls@assign@plural##1##2{%
11778   \@gls@expand@field{##1}{plural}{##2}%
11779 }%
11780 \def\gls@assign@descplural##1##2{%
11781   \@gls@expand@field{##1}{descplural}{##2}%
11782 }%
11783 \do@newglossaryentry
11784 \let\gls@assign@plural\@org@gls@assign@plural
11785 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11786 \let\gls@assign@descplural\@org@gls@assign@descplural
11787 }

```

11NewAcronymDef

```

11788 \renewcommand*{\SmallNewAcronymDef}{%
11789   \edef\@do@newglossaryentry{%
11790     \noexpand\newglossaryentry{\the\glslabeltok}%
11791     {%
11792       type=\acronymtype,%
11793       name={\noexpand\acronymfont{\the\glsshorttok}},%
11794       access={\noexpand\@glo@symbolaccess},%
11795       sort={\the\glsshorttok},%
11796       short={\the\glsshorttok},%
11797       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11798       shortaccess={\the\glslongtok},%
11799       long={\the\glslongtok},%
11800       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11801       text={\noexpand\@glo@short},%
11802       textaccess={\noexpand\@glo@shortaccess},%
11803       plural={\noexpand\@glo@shortpl},%

```

```

11804     first={\the\glslongtok},%
11805     firstaccess=\relax,
11806     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11807     description={\noexpand\@glo@first},%
11808     descriptionplural={\noexpand\@glo@firstplural},%
11809     symbol={\the\glsshorttok},%
11810     symbolaccess={\the\glslongtok},%
11811     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11812     \the\glskeylisttok
11813 }%
11814 }%
11815 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11816 \let\@org@gls@assign@plural\gls@assign@plural
11817 \let\@org@gls@assign@descplural\gls@assign@descplural
11818 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11819 \def\gls@assign@firstpl##1##2{%
11820   \@@gls@expand@field{##1}{firstpl}{##2}%
11821 }%
11822 \def\gls@assign@plural##1##2{%
11823   \@@gls@expand@field{##1}{plural}{##2}%
11824 }%
11825 \def\gls@assign@descplural##1##2{%
11826   \@@gls@expand@field{##1}{descplural}{##2}%
11827 }%
11828 \def\gls@assign@symbolplural##1##2{%
11829   \@@gls@expand@field{##1}{symbolplural}{##2}%
11830 }%
11831 \do@newglossaryentry
11832 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11833 \let\gls@assign@plural\@org@gls@assign@plural
11834 \let\gls@assign@descplural\@org@gls@assign@descplural
11835 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11836 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```

11837 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

```

pluralaccesskey

```

11838 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

```

lslongaccesskey

```

11839 \newcommand*{\glslongaccesskey}{\glslongkey access}%

```

pluralaccesskey

```

11840 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

5.5 Debugging Commands

owglonameaccess

```
11841 \newcommand*{\showglonameaccess}[1]{%  
11842   \expandafter\show\csname glo@\glsdetoklabel{#1}@access\endcsname  
11843 }
```

owglotextaccess

```
11844 \newcommand*{\showglotextaccess}[1]{%  
11845   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname  
11846 }
```

glopluralaccess

```
11847 \newcommand*{\showglopluralaccess}[1]{%  
11848   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname  
11849 }
```

wglofirstaccess

```
11850 \newcommand*{\showglofirstaccess}[1]{%  
11851   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname  
11852 }
```

rstpluralaccess

```
11853 \newcommand*{\showglofirstpluralaccess}[1]{%  
11854   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname  
11855 }
```

glosymbolaccess

```
11856 \newcommand*{\showglosymbolaccess}[1]{%  
11857   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname  
11858 }
```

bolpluralaccess

```
11859 \newcommand*{\showglosymbolpluralaccess}[1]{%  
11860   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname  
11861 }
```

owglodescaccess

```
11862 \newcommand*{\showglodescaccess}[1]{%  
11863   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname  
11864 }
```

escpluralaccess

```
11865 \newcommand*{\showglodescpluralaccess}[1]{%  
11866   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname  
11867 }
```

wgloshortaccess

```
11868 \newcommand*{\showgloshortaccess}[1]{%
11869   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname
11870 }
```

ortpluralaccess

```
11871 \newcommand*{\showgloshortpluralaccess}[1]{%
11872   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname
11873 }
```

owglolongaccess

```
11874 \newcommand*{\showglolongaccess}[1]{%
11875   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname
11876 }
```

ongpluralaccess

```
11877 \newcommand*{\showglolongpluralaccess}[1]{%
11878   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname
11879 }
```

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.

```
11880 \NeedsTeXFormat{LaTeX2e}
11881 \ProvidesPackage{glossaries-babel}[2019/12/06 v4.44 (NLCT)]
```

Load tracklang to obtain language settings.

```
11882 \RequirePackage{tracklang}
11883 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11884 \AnyTrackedLanguages
11885 {%
11886   \ForEachTrackedDialect{\this@dialect}{%
11887     \IfTrackedLanguageFileExists{\this@dialect}%
11888       {glossaries-}% prefix
11889       {.ldf}%
11890       {%
11891         \RequireGlossariesLang{\CurrentTrackedTag}%
11892       }%
11893       {%
11894         \PackageWarningNoLine{glossaries}%
11895           {No language module detected for ‘\this@dialect’.\MessageBreak
11896             Language modules need to be installed separately.\MessageBreak
11897             Please check on CTAN for a bundle called\MessageBreak
11898             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11899       }%
11900     }%
11901   }%
11902 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11903 \NeedsTeXFormat{LaTeX2e}
11904 \ProvidesPackage{glossaries-polyglossia}[2019/12/06 v4.44 (NLCT)]
```

Load tracklang to obtain language settings.

```
11905 \RequirePackage{tracklang}
11906 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11907 \AnyTrackedLanguages
```

```

11908 {%
11909     \ForEachTrackedDialect{\this@dialect}{%
11910         \IfTrackedLanguageFileExists{\this@dialect}%
11911         {glossaries-}% prefix
11912         {.ldf}%
11913         {%
11914             \RequireGlossariesLang{\CurrentTrackedTag}%
11915         }%
11916         {%
11917             \PackageWarningNoLine{glossaries}%
11918             {No language module detected for ‘\this@dialect’.\MessageBreak
11919             Language modules need to be installed separately.\MessageBreak
11920             Please check on CTAN for a bundle called\MessageBreak
11921             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11922         }%
11923     }%
11924 }%
11925 {}%

```


Glossary

`makeindex` An indexing application. [9](#), [13](#), [29](#), [30](#), [184](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [9](#), [13](#), [29](#), [30](#), [184](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 7
General: Added range facility in format key 117	
\writeist: Added spaces after \delimN and \delimR in ist file 164	
1.04 (2007-08-03)	1.12 (2008-03-08)
General: Added \glstextformat 101	\@GLSpl: now uses
1.05 (2007-08-10)	\glentrydescplural and
\glossarysection: added \@mkboth to \glossarysection 44	\glentrysymbolplural instead of
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 85	\glentrydesc and
1.07 (2007-09-13)	\glentrysymbol 131
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 115	\@Glspl@: now uses
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 161	\glentrydescplural and
1.08 (2007-10-13)	\glentrysymbolplural instead of
General: Added babel support 38	\glentrydesc and
listgroup: changed listgroup style to use \glsgetgrouptitle 278	\glentrysymbol 129
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 279	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 125
1.1 (2008-02-22)	descriptionplural: new 67
\@glossarysection: numbered sections and auto label added 45	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 85
\@gls@tmpb: changed \toksdef to \newtoks 119	descriptionplural support added 85
\@gls@toc: numberline added 46	symbolplural support added 85
\@p@glossarysection: numbered sections and auto label added 45	\Glsentrydescplural: New 154
General: amsgen now loaded (\new@ifnextchar needed) 4	\glentrydescplural: New 154
translate: translate option added 27	\Glsentrysymbolplural: New 155
\setglossarysection: new 45	\glentrysymbolplural: New 155
numberedsection: numberedsection package option added 8	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 245
	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 251
	symbolplural: new 68

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	132-140
\ACRfullpl: new	226
\Acrfullpl: new	225
\acrfullpl: new	225
\acrpluralsuffix: New	223
\gls@defglossaryentry: Changed default first value	85
Changed default firstplural value	85
Removed restriction on only using \newglossaryentry in the preamble	90
\newacronym: Removed restriction on only using \newacronym in the preamble	223
1.14 (2008-06-17)	
\@gls@hypergroup: new	273
General: added nonumberlist key to \printglossary	209
added numberedsection key to \printglossary	208
\firstacronymfont: new	226
\glsautoprefix: new	8
\glsnavhyperlink: changed \edef to \protected@edef	272
\glsnavhypertarget: added write to aux file	272
\glsnavigation: changed to only use labels for groups that are present ..	274
1.15 (2008-08-15)	
\@gls@link: added \glslabel	115
\gls@defglossaryentry: check for \@glo@first in description	89
check for \@glo@text in symbol	89
\gls@hypergroup: new	273
\glsnavhypertarget: added check if rerun required	272
\glssettoctitle: new	37
\printglossary: changed the way the TOC title is set	194
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	128
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	131
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	128
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	130
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	127
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	132
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	129
\@gls@target: raised the hypertarget so the target text doesn't scroll off the top of the page	125
\gls@defglossaryentry: Changed def to let	85
1.17 (2008-12-26)	
\@do@esc@wrglossary: new	188
\@do@seeglossary: new	191
\@glo@storeentry: new	91
\@gls@glossary: changed definition to use \index instead of \@index	184
\@glsdefaultplural: new	72
\@glsdefaultsort: new	72
\@gls@hypernumber: new	219
\@gls@noname: new	71
\@gls@nonextpages: new	210
General: added xindy support	29
parent: new	69
see: new	69
\gls@defglossaryentry: added nonumberlist key	85
added parent key	85
added see key	85
Stored main part of entry format when entry is defined	90
\gls@suffixF: new	42
\gls@suffixFF: new	42
\gls@wrglossary: modified to allow for xindy support	185

\glshyperlink: new	160	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	245
\glshypernumber: modified to allow material to be attached to location	219	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	251
\glshnavhyperlink: replaced \hyperlink to \@glslink	272	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	254
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	272	2.01 (2009 May 30)	
\glsssee: new	192	\@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	116
\glssseeformat: new	192	\forall glossaries: replaced \ifthenelse with \ifx	56
\glssSetSuffixF: new	42	\forall glossentries: replaced \ifthenelse with \ifx	56
\glssSetSuffixFF: new	42	\glssdefmain: new	16
\ifglssxindy: new	29	\glssdescwidth: changed \linewidth to \hsize	280, 302
\istfilename: added xindy support ...	41	\glsslistdottedwidth: changed \linewidth to \hsize	280
\newglossarystyle: made \newglossarystyle long	218	\glsspagelistwidth: changed \linewidth to \hsize	280, 302
\nopostdesc: new	40	nomain: added nomain package option	16
nonumberlist: new	69	\writeist: removed item_02 - no such makeindex key	168
\printglossary: added check to determine if \printglossary is already defined	194	2.02 (2007-07-13)	
added print language to aux file	194	\@printglossary: suppressed warning globally rather than locally	197
order: order package option added	29	2.02 (2009-07-13)	
\writeist: added xindy support	164	\glossarysection: changed \@mkboth to \glossarymark	44
1.18 (2009-01-14)		\glsglossarymark: New	44
\@glsl@loadlist: new	10	2.03 (2009-09-23)	
\@glsl@loadlong: new	9	\@GLS@: Added check for hyperfirst	128
\@glsl@loadsuper: new	10	\@GLSpl: Added check for hyperfirst	131
\@glsl@loadtree: new	10	\@GLs@: Added check for hyperfirst	128
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort	85	\@GLspl@: Added check for hyperfirst	130
moved sort sanitization to \newglossaryentry	89	\@glsl@: Added check for hyperfirst	127
\glstarget: new	212	\@glsl@link: new	114
\oldacronym: new	222	\@glsl@link: added \leavevmode	115
nolist: new	10	Moved entry existence check to avoid duplicate code	115
nolong: new	10	\@glsl@disp: Added check for hyperfirst	132
sort: moved sanitization to \newglossaryentry	67	\@glspl@: Added check for hyperfirst	129
nostyles: new	10	\glsglossarymark: Added check to see if it's already defined	44
nosuper: new	10	hyperfirst: new	28
notree: new	10		
1.19 (2009-03-02)			
\glsclearpage: new	46		
\glsl@disp: new	131		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	249		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	128
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms	131
\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms	128
\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms	130
\@glossaryentryfield: new	91
\@glossarysubentryfield: new	91
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	127
\@glsacronymlists: new	18
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	132
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	129
\@newglossaryentryposthook: new ..	90
\@newglossaryentryprehook: new ..	90
acronymlists: new	19
\DeclareAcronymList: new	18
\DefineAcronymSynonyms: new	239
\gls@defglossaryentry: added userl-6 keys	85
\glsadd: fixed bug that ignored counter	161
\Glsentryuseri: new	156
\glsentryuseri: new	156
\Glsentryuserii: new	157
\glsentryuserii: new	157
\Glsentryuseriii: new	157
\glsentryuseriii: new	157
\Glsentryuseriv: new	157
\glsentryuseriv: new	157
\Glsentryuserv: new	157
\glsentryuserv: new	157
\Glsentryuservi: new	158
\glsentryuservi: new	158
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	64
\SetAcronymLists: new	19
\SetDefaultAcronymDisplayStyle: new	241
\SetDefaultAcronymStyle: new	242
\SetDescriptionAcronymDisplayStyle: new	247
\SetDescriptionDUAAcronymDisplayStyle: new	245
\SetDescriptionFootnoteAcronymDisplayStyle: new	243
\SetDUADisplayStyle: new	254
\SetFootnoteAcronymDisplayStyle: new	249
\SetSmallAcronymDisplayStyle: new	252
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	131
Removed spurious brace. Patch provided by Sergiu Dotenco	132
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	169
2.06 (2010-06-14)	
\altnewglossary: new	65
\CustomAcronymFields: new	257
\CustomNewAcronymDef: new	257
\SetCustomDisplayStyle: new	256
\SetCustomStyle: new	257
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	161
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites	173
\gls@wrglossary: modified to take into account savewrites	185
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument	117
3.0 (2011-04-02)	
\@do@esc@wrglossary: added check for hyper location prefix	190
modified to use new format	188
\@@glossarysec: replaced \@ifundefined with \ifcsundef ...	7
\@do@seeglossary: Sanitize and escape cross-referencing information	191
\@gls@counterwithin: new	12

\@gls@ifinlist: new	47	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter	161
\@gls@saveentrycounter	116	\GlsAddXdyCounters: new	48
added \@gls@setsort	116	\glentrycounterlabel: new	211
\@gls@saveentrycounter: new	116	\glentryitem: new	212
\@gls@setupsort@def: new	14	\Glentrylong: new	158
\@gls@setupsort@standard: new	13	\glentrylong: new	158
\@gls@setupsort@use: new	14	\Glentrylongpl: new	158
\@gls@xdy@locationlist: new	50	\glentrylongpl: new	158
\@glslink: replaced \@ifundefined		\Glentryshort: new	158
with \ifcsundef	125	\glentryshort: new	158
\@glsnextpages: new	210	\Glentryshortpl: new	158
\@print@glossary: replaced		\glentryshortpl: new	158
\@ifundefined with \ifcsundef ..	197	\glsgetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	216
\currentglossary	196	\gls glossarymark: replaced	
added \glsnextpages	196	\@ifundefined with \ifcsundef ..	44
make toctitle default to title	196	\glshyperlink: changed default from	
\@xdyattributelist: new	47	\glentryname to \glentrytext ..	160
General: added prefix to hyperlink	220	\glshypernumber: replaced	
etoolbox now loaded	4	\@ifundefined with \ifcsundef ..	219
replaced \@ifundefined with		\glsnumberformat: replaced	
\ifcsundef	36, 38, 112, 208	\@ifundefined with \ifcsundef ..	42
\acrfootnote: new	242	\glsrefentry: new	211
\ACRfull: added starred version	224	\glsresetsubentrycounter: new ...	210
\Acrfull: added starred version	224	\glsseeitem: hyperlink uses	
\acrfull: added starred version	223	\glsseeitemformat instead of	
\ACRfullpl: added starred version ...	226	\glentryname	193
\Acrfullpl: added starred version ...	225	\glsseeitemformat: new	193
\acrfullpl: added starred version ...	225	\glssortnumberfmt: new	14
\acrlinkfootnote: new	242	\glsstepentry: new	211
\acrno linkfootnote: new	243	\glsstepsubentry: new	211
savewrites: new	31	\glssubentrycounterlabel: new ...	211
see: added \@glo@seeautonumberlist ..	69	\glssubentryitem: new	212
seeautonumberlist: new	9	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef	212
\@ifundefined with \ifcsundef ..	44	short: new	71
\glossarystyle: replaced		shortplural: new	71
\@ifundefined with \ifcsundef ..	218	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	57
\@ifundefined with \ifcsundef ..	30	\ifglentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	57
\@gls@defsort	89	\istfile: deprecated	183
added short and long keys	86	glossaryentry: new	11
replaced \@ifundefined with		glossarysubentry: new	12
\ifcsundef	86	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	46	\newrobustcmd	74

\newglossarystyle: replaced	
\@ifundefined with \ifcsundef .	218
\ns@newglossary: added	
\@gls@defsortcount	64
replaced \@ifundefined with	
\ifcsundef	64
entrycounter: new	12
\oldacronym: replaced \@ifundefined	
with \ifcsundef	222
compatible-2.07: compatible-2.07	
option added	31
long: new	71
longplural: new	71
nonumberlist: now boolean	69
sort: new	13
counter: replaced \@ifundefined with	
\ifcsundef	68
counterwithin: new	12
\printglossary: replaced	
\@ifundefined with \ifcsundef .	194
\SetDescriptionFootnoteAcronymDisplayStyle	
expanded options link options	243
\setentrycounter: added optional	
argument	217
\showacronymlists: new	263
\showglocounter: new	260
\showgloDESC: new	261
\showgloDESCplural: new	261
\showglofirst: new	259
\showglofirstpl: new	259
\showgloflag: new	262
\showgloindex: new	262
\showglolevel: new	259
\showglongname: new	261
\showgloparent: new	258
\showgloplural: new	259
\showglosort: new	261
\showglossaries: new	263
\showglossarycounter: new	264
\showglossaryentries: new	264
\showglossaryin: new	263
\showglossaryout: new	263
\showglossarytitle: new	263
\showglosymbol: new	261
\showglosymbolplural: new	262
\showgлотext: new	259
\showgлотype: new	259
\showglouserii: new	260
\showglouseriii: new	260
\showglouseriv: new	260
\showglouserv: new	260
\showglouservi: new	261
subentrycounter: new	12
\writeist: added xindy-only macro	
definitions to glossary open tag	166
modified to support new format	164
3.01 (2011-04-12)	
\@glswritefiles: added check for	
empty glossaries	183
General: made robust	128
\ACRfull: made robust	224
\Acrfull: made robust	224
\acrfull: made robust	223
\acrfullformat: removed	
\acronymfont as it should already be	
set in the second argument.	224
\ACRfullpl: made robust	226
\Acrfullpl: made robust	225
\acrfullpl: made robust	225
\ACRlong: made robust	149
\Acrlong: made robust	149
\acrlong: made robust	148
\ACRlongpl: made robust	151
\Acrlongpl: made robust	151
\acrlongpl: made robust	150
\ACRshort: made robust	146
\Acrshort: made robust	145
\acrshort: made robust	144
\ACRshortpl: made robust	147
\Acrshortpl: made robust	147
\acrshortpl: made robust	146
\Gls: made robust	127
\glsadd: made robust	161
\glsaddall: made robust	161
\GLSdesc: made robust	137
\Glsdesc: made robust	137
\glsdesc: made robust	136
\GLSdescplural: made robust	138
\Glsdescplural: made robust	137
\glsdescplural: made robust	137
\glsfirst: made robust	133
\GLSfirstplural: made robust	135
\Glsfirstplural: made robust	135
\glsfirstplural: made robust	135
\glslink: made robust	114
\GLSname: made robust	136
\Glsname: made robust	136

\glsname: made robust	136	\@printglossary: add a way to fetch	
\GLSpl: made robust	130	current entry label	196
\Glspl: made robust	130	savenumberlist: new	9
\glspl: made robust	129	ucmark: new	11
\GLSplural: made robust	134	\gls@defglossaryentry: added	
\GLSsymbol: made robust	139	numberlist element	89
\Glsymbol: made robust	138	\gls@save@numberlist: new	193
\glssymbol: made robust	138	\gls@wrglossary: added check for	
\GLSsymbolplural: made robust	139	glossary file defined	185
\Glsymbolplural: made robust	139	\glsdisplaynumberlist: new	159
\glssymbolplural: made robust	139	\glsentrycounter: set default value ..	116
\Glstext: made robust	133	\Glsentryfull: fixed bug (replaced	
\glstext: made robust	132	\glsentryshortpl with	
\GLSuseri: made robust	140	\glsentryshort)	159
\Glsuseri: made robust	140	\glsentryfullpl: fixed bug (replaced	
\glsuseri: made robust	140	\glsentryshort with	
\GLSuserii: made robust	141	\glsentryshortpl)	159
\Glsuserii: made robust	141	\glsentrynumberlist: new	159
\glsuserii: made robust	140	\glsmoveentry: new	91
\GLSuseriii: made robust	142	\glsresetsubentrycounter: new ...	211
\Glsuseriii: made robust	142	\ifglshaschildren: new	59
\glsuseriii: made robust	141	\ifglshasparent: new	59
\GLSuseriv: made robust	143	\makeglossaries: added list parser ..	178
\Glsuseriv: made robust	142	indexonlyfirst: new	28
\glsuseriv: made robust	142	\renewglossarystyle: new	219
\GLSuserv: made robust	143	\showglossaryentries: fixed misspelt	
\Glsuserv: made robust	143	command	264
\glsuserv: made robust	143	\SmallNewAcronymDef: fixed broken	
\GLSuservi: made robust	144	short and long plural	252
\Glsuservi: made robust	144	3.03 (2012/09/21)	
\glsuservi: made robust	144	\@gls@sanitizesort: new	22
3.02 (2012-05-19)		\@gls@setupsort@standard: used	
\glsnumlistlastsep: new	160	\@gls@sanitizesort	13
\glsnumlistsep: new	160	\@printglossary: allow title to override	
3.02 (2012-05-21)		default toctitle	195
\@do@wrglossary: changed		General: allow title to set toctitle	208
\@glslocref to		\glsinlinedescformat: new	276
\theglsentrycounter	190	\glsinlineemptydescformat: new ..	276
\@do@wrglossary: changed		\glsinlinenameformat: new	276
\@do@wr@glossary to test for		\glsinlinepostchild: new	276
indexonlyfirst option; put old		\glsinlinesubdescformat: new	276
\@do@wr@glossary code into		\glsinlinesubnameformat: new	276
\@do@wrglossary	185	\glspostinline: replaced “.” with	
\@gls@missingnumberlist: new	72	\glspostdescription	276
\@glswritefiles: added check for		list: added check for glsnogroupskip ..	278
existence of token in case		altlongragged4col: added check for	
\makeglossaries has been		glsnogroupskip	295
omitted	183	altsuperragged4col: added check for	
		glsnogroupskip	314

alttree: added check for		\gls@disablepagerefexpansion: new	186
glsnogroupskip	323	\gls@numberpage: new	186
index: added check for glsnogroupskip	317	\gls@protected@pagefmts: new	185
nogroupskip: new	11	\gls@romanpage: new	186
long: added check for glsnogroupskip	281	\glsdefmain: added check for doc	
long3col: added check for		package	16
glsnogroupskip	283	\glsorg@endtheglossary: new	5
long4col: added check for		\glsorg@theglossary: new	5
glsnogroupskip	284	\PrintChanges: new	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip	292	\@do@esc@wrglossary: add Roman	
longragged3col: added check for		case. Fixed bugs in the else	
glsnogroupskip	293	statements	189
nopostdot: new	11	\@gls@link: added check for	
tree: added check for glsnogroupskip	318	“nohypertypes”	115
treenoname: added check for		mcolalttree: replaced ‘2’ with	
glsnogroupskip	320	\glsmcols	301
super: added check for glsnogroupskip	303	mcolindex: replaced ‘2’ with \glsmcols	297
super3col: added check for		mcolindexspannav: replaced ‘2’ with	
glsnogroupskip	305	\glsmcols	298
super4col: added check for		mcoltree: replaced ‘2’ with \glsmcols	298
glsnogroupskip	307	mcoltreenoname: replaced ‘2’ with	
superragged: added check for		\glsmcols	300
glsnogroupskip	310	mcoltreespannav: replaced ‘2’ with	
superragged3col: added check for		\glsmcols	299
glsnogroupskip	312	\gls@protected@pagefmts: added	
3.04 (2012-11-11)		Roman to list	185
altlist: replaced \newline with		\gls@Romanpage: new	186
paragraph break	278	\glsgetgrouplabel: fixed bug (typo in	
3.04 (2012-11-18)		\equal)	217
\@do@wrglossary: changed		\nopostdesc: made robust	40
\theglsentrycounter back to		3.05 (2013/04/21)	
\@glslocref	190	\@gls@nohyperlist: new	19
\@do@esc@wrglossary: modified to		\GlsDeclareNoHyperList: new	19
compensate for possible incorrect		nohypertypes: new	19
page number	189	3.06 (2013/06/17)	
\@gls@escbsdq: unsanitize		\@xdy@main@language: Changed back to	
\gls@numberpage, \gls@alphpage,		using \language	29
\gls@Alphpage and		\findrootlanguage: Obsoleted	54
\gls@romanpage	118	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	115
unwanted whatsit occurring here. . .	197	\glossarypreamble: modified to work	
General: Added check for doc package	4	with \setglossarypreamble	43
added datatool-base as a required		\gls@docclearpage: added check for	
package	4	openright	46
added local key	112	\glspostdescription: Added	
\gls@Alphpage: new	186	spacefactor code	11
\gls@alphpage: new	186		

\GlsSetXdyCodePage: Added check for fontspec	55	\glssseelist: made robust	192
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	247	\ifglsdscsuppressed: new	60
\setglossarypreamble: new	43	\ifglshasdesc: new	60
3.08a (2013-08-30)		\ifglshassymbol: new	60
list: updated list style to use \glossentry and \subglossentry	277	altlongragged4col: updated to use \glossentry and \subglossentry	295
listdotted: updated listdotted style to use \glossentry and \subglossentry	279	alttree: updated to use \glossentry and \subglossentry	322
altlist: updated altlist style to use \glossentry and \subglossentry	278	index: added paragraph break at end of environment	316
inline: updated inline style to use \glossentry and \subglossentry	275	updated to use \glossentry and \subglossentry	316
3.08a (2013-09-28)		long: updated to use \glossentry and \subglossentry	281
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	92	longragged: updated to use \glossentry and \subglossentry	292
updated for \glossentry	92	longragged3col: updated to use \glossentry and \subglossentry	293
\@glossaryentryfield: switched to \glossentry	91	tree: updated to use \glossentry and \subglossentry	318
\@glossarysubentryfield: switched to \subglossentry	91	\setglossarystyle: new	217
General: added nogroupskip key to \printglossary	208	\setglossentrycompatibility: new	214
removed definition of \@glossaryentryfield	365	superragged: updated to use \glossentry and \subglossentry	310
removed definition of \@glossarysubentryfield	365	3.09a (2013-10-09)	
\compatibleglossentry: new	213	\@gls@assign@symbolplural@field: new	22
\compatiblesubglossentry: new ...	214	\@gls@default@value: new	68
\glossaryentryfield: deprecated ...	215	\Glsentrydesc: made robust	154
\Glossentrydesc: new	213	\Glsentrydescplural: made robust ..	154
\glossentrydesc: new	213	\Glsentryfirst: made robust	155
\Glossentryname: new	213	\Glsentryfirstplural: made robust .	156
\glossentryname: new	213	\Glsentryfull: made robust	159
\Glossentrysymbol: new	214	\Glsentryfullpl: made robust	159
\glossentrysymbol: new	214	\Glsentrylong: made robust	158
\gls@assign@desc@field: new	21	\Glsentrylongpl: made robust	158
\gls@assign@descplural@field: new	21	\Glsentryname: made robust	153
\gls@assign@field: new	74	\Glsentryplural: made robust	155
\gls@ifnotmeasuring: new	93	\Glsentryshort: made robust	158
\glsaddallunused: new	162	\Glsentryshortpl: made robust	158
\glsexpandfields: new	74	\Glsentrysymbol: made robust	155
\glsnoexpandfields: new	74	\Glsentrysymbolplural: made robust	155
\glsssee: made robust	192	\Glsentrytext: made robust	154
\glssseeformat: made robust	192	\Glsentryuseri: made robust	156
\glssseeitem: made robust	193	\Glsentryuserii: made robust	157
		\Glsentryuseriii: made robust	157
		\Glsentryuseriv: made robust	157
		\Glsentryuserv: made robust	157
		\Glsentryuservi: made robust	158

<code>\glstextup</code> : new	223	removed <code>\MakeUppercase</code> as now dealt with in <code>\glentryfmt</code>	131
<code>\ifglshassymbol</code> : changed test to check for <code>\@gls@default@symbol</code>	60	<code>\@Gls@</code> : add <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	128
3.10a (2013-09-28)		change to using <code>\glentryfmt</code> style commands	128
<code>\gls@assign@type@field</code> : new	21	removed <code>\makefirstuc</code> (now dealt with in <code>\glentryfmt</code>)	128
3.10a (2013-10-13)		<code>\@Glspl@</code> : add <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	130
<code>\@gls@keymap</code> : new	76	change to using <code>\glentryfmt</code> style commands	130
<code>\@gls@provide@newglossary</code> : new ...	62	removed <code>\makefirstuc</code> (now dealt with in <code>\glentryfmt</code>)	130
<code>\@gls@writedef</code> : new	76	<code>\@acrlong</code> : added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	364
<code>\@glsdefaultplural</code> : Obsolete	72	<code>\@acrshort</code> : added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	363
<code>\@glsnodelsc</code> : new	72	<code>\@gls@</code> : add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	127
<code>\@print@glossary</code> : Added providecommand code to aux file	197, 198	change to using <code>\glentryfmt</code> style commands	127
<code>\gls@defglossaryentry</code> : Changed to using <code>\@gls@default@value</code>	85	<code>\@gls@noexpand@fields</code> : Fixed bug expand replaced with <code>noexpand</code>	73
new	84	<code>\@glsdisp</code> : add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	131
<code>\glswritedefhook</code> : new	84	change to using <code>\glentryfmt</code> style commands	132
<code>\makeglossaries</code> : Added providecommand code to aux file ..	177	<code>\@glspl@</code> : add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	129
<code>\new@glossaryentry</code> : new	75	General: added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	145–151
<code>\ns@newglossary</code> : added <code>\@gls@provide@newglossary</code>	64	changed to just use <code>\Glsentrydescplural</code>	138
3.11a (2013-10-15)		changed to just use <code>\Glsentrydesc</code> .	137
<code>\@ACRlong</code> : added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	365	changed to just use <code>\glentrydesc</code> .	137
<code>\@ACRshort</code> : added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	363		
<code>\@Acrlong</code> : added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	364		
<code>\@Acrshort</code> : added <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glsinsert</code> and <code>\glscustomtext</code>	363		
<code>\@GLS@</code> : add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	128		
change to using <code>\glentryfmt</code> style commands	128		
removed <code>\MakeUppercase</code> (now moved to <code>\glentryfmt</code>)	128		
<code>\@Glspl</code> : add <code>\glslabel</code> , <code>\glsifplural</code> , <code>\glscapscase</code> , <code>\glscustomtext</code> and <code>\glsinsert</code>	131		
change to using <code>\glentryfmt</code> style commands	131		

changed to just use		<code>\gls@doentryfmt</code> : new	63
<code>\Glsentryfirstplural</code>	135	<code>\glsdisplay</code> : obsoleted	110
changed to just use		<code>\glsdisplayfirst</code> : obsoleted	110
<code>\glsentryfirstplural</code>	135	<code>\glsgenentryfmt</code> : new	105
changed to just use <code>\Glsentryfirst</code>	134	<code>\glsgetgrouptitle</code> : Added check in	
changed to just use		case non-Latin alphabet in use	216
<code>\glsentryfirst</code>	133, 134	<code>\glsglossarymark</code> : replaced	
changed to just use <code>\Glsentryname</code>	136	<code>\MakeUppercase</code> with	
changed to just use <code>\glsentryname</code>	136	<code>\mfirstucMakeUppercase</code>	44
changed to just use <code>\Glsentryplural</code>	134	<code>\glsnavigation</code> : switched to using	
changed to just use		<code>\@gls@getgrouptitle</code>	274
<code>\glsentryplural</code>	134, 135	<code>\ifglshasdesc</code> : replaced <code>\ifdefempty</code>	
changed to just use		with <code>\ifcsempy</code>	60
<code>\Glsentrysymbolplural</code>	139	<code>\ifglshaslong</code> : new	60
changed to just use		<code>\ifglshasshort</code> : new	60
<code>\glsentrysymbolplural</code>	139, 140	<code>\ifglshassymbol</code> : replaced	
changed to just use <code>\Glsentrysymbol</code>	138	<code>\ifdefempty</code> with <code>\ifcsempy</code>	60
changed to just use		<code>\ifglssused</code> : replaced <code>\ifthenelse</code> with	
<code>\glsentrysymbol</code>	138, 139	<code>\ifbool</code>	57
Changed to just use <code>\Glsentrytext</code>	133	<code>\longnewglossaryentry</code> : new	84
changed to just use <code>\glsentrytext</code>	132	<code>\ns@newglossary</code> : replaced	
changed to just use		<code>\glsdisplay</code> and	
<code>\Glsentryuseriii</code>	142	<code>\glsdisplayfirst</code> with	
changed to just use		<code>\glsentryfmt</code>	64
<code>\glsentryuseriii</code>	141, 142	compatible-3.07: <code>cnew</code>	31
changed to just use <code>\Glsentryuserii</code>	141	<code>\SetCustomDisplayStyle</code> : updated to	
changed to just use <code>\glsentryuserii</code>	141	use <code>\defglentryfmt</code>	256
changed to just use <code>\Glsentryuseriv</code>	142	<code>\SetDefaultAcronymDisplayStyle</code> :	
changed to just use		changed to use <code>\defglentryfmt</code>	241
<code>\glsentryuseriv</code>	142, 143	<code>\SetDescriptionAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuseri</code>	140	updated to use <code>\defglentryfmt</code>	247
changed to just use <code>\glsentryuseri</code>	140	<code>\SetDescriptionDUAAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuservi</code>	144	updated to use <code>\defglentryfmt</code>	245
changed to just use <code>\glsentryuservi</code>	144	<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuserv</code>	143	updated to use <code>\defglentryfmt</code>	243
changed to just use		<code>\SetDUADisplayStyle</code> : updated to use	
<code>\glsentryuserv</code>	143, 144	<code>\defglentryfmt</code>	254
Now requires <code>textcase</code>	4	<code>\SetFootnoteAcronymDisplayStyle</code> :	
acronymlists: replaced		updated to use <code>\defglentryfmt</code>	249
<code>\@addtoacronymlists</code> with		<code>\SetSmallAcronymDisplayStyle</code> :	
<code>\DeclareAcronymList</code>	19	updated to use <code>\defglentryfmt</code>	252
<code>\defgldisplay</code> : obsoleted	111	<code>\setupglossaries</code> : new	35
<code>\defgldisplayfirst</code> : obsoleted	111	<code>\showglong</code> : new	262
<code>\defglentryfmt</code> : new	63	<code>\showgshort</code> : new	262
<code>\forglentries</code> : replaced <code>\ifx</code> with		numbers: new	33
<code>\ifdefempty</code>	56	symbols: new	33
<code>\gls@assign@desc</code> : new	84		
<code>\gls@defglossaryentry</code> : Fixed default		<code>\gls@defglossaryentry</code> : added	
counter if none supplied	88	<code>\glslabel</code>	85

\glsaddkey: new	78	altsuper4colheaderborder: switched to \tabularnewline	309
3.13a (2013-11-05)		long: switched to \tabularnewline ..	281
\@gls@assign@symbol@field: changed to use \glssetnoexpandfield	22	long3col: switched to \tabularnewline	282
\@gls@assign@symbolplural@field: changed to use \glssetnoexpandfield	22	long3colheader: switched to \tabularnewline	283
\@gls@link: removed \relax	116	long3colheaderborder: switched to \tabularnewline	283
\@gls@notranslatorhook: new	26	long4col: switched to \tabularnewline	284
\@gls@setupsort@standard: moved \@gls@santizesort to \glsprestandardsort	13	long4colheader: switched to \tabularnewline	284
ucmark: added check for memoir	11	longheader: switched to \tabularnewline	282
see: added \gls@checkseeallowed ...	69	longheaderborder: switched to \tabularnewline	282
\glossarysection: changed \glossarymark to \gls glossarymark	44	\SetFootnoteAcronymDisplayStyle: fixed missing argument bug	249
\glossarystyle: fixed bug caused by using \ifdef instead of \ifcsdef .	218	super: switched to \tabularnewline .	303
\gls@assign@desc@field: changed to use \glssetnoexpandfield	21	super3col: switched to \tabularnewline	305
\gls@assign@descplural@field: changed to use \glssetnoexpandfield	21	super3colheader: switched to \tabularnewline	305
\gls@assign@name@field: changed to use \glssetnoexpandfield	22	super4col: switched to \tabularnewline	306
\gls@assign@type@field: changed to use \glssetexpandfield	21	super4colheader: switched to \tabularnewline	307
\gls@checkseeallowed: new	69	super4colheaderborder: switched to \tabularnewline	307
\glsaddallunused: set default to \@glo@types	162	superheader: switched to \tabularnewline	304
\Glsentryfull: changed to use \acrfullformat	159	superheaderborder: switched to \tabularnewline	304
\Glsentryfull: changed to use \acrfullformat	159	3.14a (2013-11-12)	
\Glsentryfullpl: changed to use \acrfullformat	159	\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles	183
\gls glossarymark: renamed \glossarymark to \gls glossarymark to avoid conflict with memoir	44	General: new	265
\glsprestandardsort: new	13	acronyms: new	17
\glssetexpandfield: new	21	\gls@def glossaryentry: added check for existence of default glossary	86
\glssetnoexpandfield: new	21	set the default for firstplural to be the value of plural	88
altsuper4colheader: switched to \tabularnewline	308	xindy gloss: new	30
		\longprovide glossaryentry: new ...	84

compatible-2.07: added check for 2.07 before setting 3.07 compatibility	31	xindynoglsnumbers: new	30
notranslate: new	26	sm-short-long: new	232
\provideglossaryentry: new	75	sm-short-long-desc: new	234
4.0 (2013-11-14)		index: new	34
\gls@defglossaryentry: added check for first key	88	\newacronymstyle: new	229
super: fixed typo in \subglossentry (\glossentrydesc)	303	long-sc-short: new	231
4.01 (2013-11-16)		long-sc-short-desc: new	233
General: fixed non-value options so that they can be passed to document class .	8	long-short: new	229
\CustomAcronymFields: inserted missing comma	257	long-short-desc: new	232
4.02 (2013-12-05)		long-sm-short: new	232
\@acrfull: now using \acrfullfmt . .	223	long-sm-short-desc: new	233
\@gls@indexdef: new	34	long-sp-short-desc: new	232
\@gls@numbersdef: new	34	footnote: new	236
\@gls@symbolsdef: new	33	footnote-desc: new	238
General: Removed \acronymfont .	148–152	footnote-sc: new	238
\ACRfullfmt: new	225	footnote-sc-desc: new	239
\Acrfullfmt: new	224	footnote-sm: new	238
\acrfullfmt: new	224	footnote-sm-desc: new	239
\ACRfullplfmt: new	226	\setacronymstyle: new	228
\Acrfullplfmt: new	225	\SetDescriptionAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	247
\acrfullplfmt: new	225	\SetDescriptionFootnoteAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	243
\acronymentry: new	228	\SetFootnoteAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	249
sanitize: fixed bug that caused an error here	25	\SetGenericNewAcronym: new	227
sc-short-long: new	232	\SetSmallAcronymDisplayStyle: Moved check for empty custom text to prevent unwanted parenthetical material	252
sc-short-long-desc: new	234	dua: new	234
\Genacrfullformat: new	110	dua-desc: new	236
\genacrfullformat: new	110	numberedsection: added nameref option	8
\GenericAcronymFields: new	228	4.02 (2013-13-05)	
\Genplacrfullformat: new	110	\makeglossaries: made preamble only	179
\Glsentryfull: bug fix: added missing \acronymfont	159	4.03 (2014-01-17)	
\glsentryfull: bug fix: added missing \acronymfont	159	General: changed default to \@empty instead of \relax	31
\Glsentryfullpl: bug fix: added missing \acronymfont	159	4.03 (2014-01-20)	
\glsentryfullpl: bug fix: added missing \acronymfont	159	\@do@esc@wrglossary: added	
\glsgenacfmt: new	108	\glsdetoklabel	190
\GlsUseAcrEntryDisplayStyle: new . . .	229		
\GlsUseAcrStyleDefs: new	229		
short-long: new	231		
short-long-desc: new	233		

\@do@noesc@wrglossary: added		\Genacrfullformat: redefined to use	
\glsdetoklabel	187	accessibility information	362
\@ACRlong: removed \glslabel		\genacrfullformat: redefined to use	
(defined in \@gls@link)	365	accessibility information	362
\@ACRshort: removed \glslabel		\Genplacrfullformat: redefined to use	
(defined in \@gls@link)	363	accessibility information	362
\@Acrlong: removed \glslabel		\genplacrfullformat: redefined to use	
(defined in \@gls@link)	364	accessibility information	362
\@Acrshort: removed \glslabel		\glossentryname: added	
(defined in \@gls@link)	363	\glsdetoklabel	213
\@GLS@: removed \glslabel (defined in		\gls@defglossaryentry: added	
\@gls@link)	128	\glsdetoklabel	84
\@GLSpl: removed \glslabel (defined		replaced #1 with \@gls@label	86
in \@gls@link)	131	replaced \ifthenelse with	
\@Gls@: removed \glslabel (defined in		\ifdefequal	87
\@gls@link)	128	\glsadd: added \glsdetoklabel	161
\@Gls@entry@field: new	152	\glsaddkey: switched to using	
\@Glspl@: removed \glslabel (defined		\@gls@field@link	79
in \@gls@link)	130	\glsdetoklabel: new	57
\@acrlong: removed \glslabel		\glsdisplaynumberlist: added	
(defined in \@gls@link)	364	\glsdetoklabel	160
\@acrshort: removed \glslabel		\glsdoifexistsorwarn: new	58
(defined in \@gls@link)	363	\glsentryaccess: switched to using	
\@gls@: removed \glslabel (defined in		\@gls@entry@field	349
\@gls@link)	127	\glsentrydescaccess: switched to	
\@gls@access@display: new	351	using \@gls@entry@field	350
\@gls@entry@field: new	152	\glsentrydescpluralaccess: switched	
\@gls@fetchfield: new	77	to using \@gls@entry@field	350
\@gls@field@link: new	132	\glsentryfirstaccess: switched to	
\@gls@link: added \glsdetoklabel .	115	using \@gls@entry@field	350
moved \@gls@link@opts and		\glsentryfirstplural: added	
\@gls@link@label to \@gls@link	115	\glsdetoklabel	156
\@gls@writedef: added		\glsentrylongaccess: switched to	
\glsdetoklabel	76	using \@gls@entry@field	351
\@glsdisp: removed \glslabel		\glsentrylongpluralaccess: switched	
(defined in \@gls@link)	131	to using \@gls@entry@field	351
\@Glspl@: removed \glslabel (defined		\glsentrypluralaccess: switched to	
in \@gls@link)	129	using \@gls@entry@field	350
\@printglossary: added		\glsentryshortaccess: switched to	
\glsdetoklabel	196	using \@gls@entry@field	350
General: removed \glslabel (defined in		\glsentryshortpluralaccess:	
\@gls@link)	145	switched to using	
sc-short-long-desc: redefined to use		\@gls@entry@field	350
accessibility information	369	\glsentrysymbolaccess: switched to	
\compatibleglossentry: added		using \@gls@entry@field	350
\glsdetoklabel	345	\glsentrysymbolpluralaccess:	
\compatiblesubglossentry: added		switched to using	
\glsdetoklabel	346	\@gls@entry@field	350

\glsentrytextaccess: switched to using \@gls@entry@field	350	replaced \ifcsempty with \ifdefempty and replaced \ifx with \ifdefequal	60
\glsгенacfmt: redefined to use accessibility information	360	\ifglused: added \glsdetoklabel ..	57
\glsгенentryfmt: redefined to use accessibility information	357	sm-short-long-desc: redefined to use accessibility information	369
\glshyperlink: added \glsdetoklabel	160	long-sc-short-desc: redefined to use accessibility information	368
\glslocalreset: added \glsdetoklabel	93	long-short: redefined to use accessibility information	366
\glslocalunset: added \glsdetoklabel	94	long-short-desc: redefined to use accessibility information	368
\glsmoveentry: added \glsdetoklabel	91	long-sm-short-desc: redefined to use accessibility information	368
replaced \ifthenelse with \ifdefequal	91	footnote: redefined to use accessibility information	372
\glsrefentry: added \glsdetoklabel	211	footnote-desc: redefined to use accessibility information	374
\glsreset: added \glsdetoklabel ...	93	footnote-sc: redefined to use accessibility information	374
\glsseelist: added \expandafter commands	193	footnote-sc-desc: redefined to use accessibility information	375
\glsstepentry: added \glsdetoklabel	211	footnote-sm: redefined to use accessibility information	374
\glsstepsubentry: added \glsdetoklabel	211	footnote-sm-desc: redefined to use accessibility information	375
\glsunset: added \glsdetoklabel ...	94	\renewacronymstyle: new	229
short-long: commented spurious EOL	231	\showglocounter: added \glsdetoklabel	260
redefined to use accessibility information	367	\showglodesc: added \glsdetoklabel	261
short-long-desc: redefined to use accessibility information	369	\showglodescaccess: added \glsdetoklabel	381
\ifglsdscsuppressed: added \glsdetoklabel	60	\showglodescplural: added \glsdetoklabel	261
fixed typo	60	\showglodescpluralaccess: added \glsdetoklabel	381
\ifglsentryexists: added \glsdetoklabel	57	\showglofirst: added \glsdetoklabel	259
\ifglshaschildren: added \glsdetoklabel	59	\showglofirstaccess: added \glsdetoklabel	381
\ifglshasdesc: added \glsdetoklabel	60	\showglofirsttpl: added \glsdetoklabel	259
\ifglshasfield: new	61	\showglofirstpluralaccess: added \glsdetoklabel	381
\ifglshaslong: added \glsdetoklabel	60	\showgloflag: added \glsdetoklabel	262
\ifglshasparent: added \glsdetoklabel	59	\showgloindex: added \glsdetoklabel	262
\ifglshasshort: added \glsdetoklabel	60	\showglolevel: added \glsdetoklabel	259
\ifglshassymbol: added \glsdetoklabel	60		

\showglolong: added \glsdetoklabel	262	redefined to use accessibility	
\showglolongaccess: added		information	370
\glsdetoklabel	382	dua-desc: commented spurious EOL ..	236
\showglolongpluralaccess: added		redefined to use accessibility	
\glsdetoklabel	382	information	372
\showglongname: added \glsdetoklabel	261	4.04 (2014-03-04)	
\showglongnameaccess: added		\@gls@getcounterprefix: added	
\glsdetoklabel	381	warning if no prefix can be formed ..	191
\showgloparent: added		4.04 (2014-03-06)	
\glsdetoklabel	258	\@@gls@noidx@nosanitizesort: new ..	23
\showgloplural: added		\@@gls@noidx@sanitizesort: new ...	22
\glsdetoklabel	259	\@@gls@nosanitizesort: new	22
\showglopluralaccess: added		\@@gls@sanitizesort: new	22
\glsdetoklabel	381	\@glo@addchildren: new	199
\showgloshort: added		\@glo@do@sortentries: new	199
\glsdetoklabel	262	\@glo@grabfirst: new	204
\showgloshortaccess: added		\@glo@sortedinsert: new	200
\glsdetoklabel	382	\@glo@sortentries: new	198
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new	201
\glsdetoklabel	382	\@glo@sorthandler@letter: new ...	200
\showglosort: added \glsdetoklabel	261	\@glo@sorthandler@nocase: new ...	201
\showglosymbol: added		\@glo@sorthandler@word: new	200
\glsdetoklabel	261	\@glo@sortmacro@case: new	202
\showglosymbolaccess: added		\@glo@sortmacro@def: new	202
\glsdetoklabel	381	\@glo@sortmacro@def@do: new	203
\showglosymbolplural: added		\@glo@sortmacro@letter: new	201
\glsdetoklabel	262	\@glo@sortmacro@nocase: new	202
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ...	202
\glsdetoklabel	381	\@glo@sortmacro@use: new	203
\showglotext: added \glsdetoklabel	259	\@glo@sortmacro@word: new	201
\showglotextaccess: added		\@gls@noidx@do: new	205
\glsdetoklabel	381	\@gls@noidx@getgrouptitle: new ..	216
\showglotype: added \glsdetoklabel	259	\@gls@noref@warn: new	182
\showglouserii: added		\@gls@reference: new	207
\glsdetoklabel	260	\@gls@warnonglossdefined: new	20
\showglouseriii: added		\@gls@warnontheGLOSSdefined: new ..	21
\glsdetoklabel	260	\@no@makeglossaries: new	182
\showglouseriiii: added		\@print@glossary: new	197
\glsdetoklabel	260	\@print@noidx@glossary: new	203
\showglouseriv: added		\@print@gloss@setsort: new	195
\glsdetoklabel	260	\@print@glossary: new	195
\showglouseriv: added		General: added sort key to printgloss	
\glsdetoklabel	260	group	209
\showglouseriv: added		\compatibleglossentry: changed	
\glsdetoklabel	261	\newcommand to \def as is may or	
dua: fixed bug in \acrfullfmt	235	may not be defined	345
fixed bug in \Acrfullplfmt	236	\compatiblesubglossentry: changed	
fixed bug in \acrfullplfmt	236	\newcommand to \def as is may or	
		may not be defined	346

\defglsdisplayfirst: fixed unwanted space	111	\Acrfullplfmt: fixed no case change bug	225
\glo@grabfirst: new	204	\glsletentryfield: new	152
\gls@defglossaryentry: replaced \ifx with \ifdefvoid	90	4.08 (2014-07-30)	
\glsnoidxdisplayloc: new	207	\@ACRlong: added	
\glsnoidxdisplaylocclishandler: new	206	\do@gls@link@checkfirsthyper	364
\glsnoidxloclist: new	206	\@ACRshort: added	
\glsnoidxlocclishandler: new	206	\do@gls@link@checkfirsthyper	363
\glsnoidxstripaccents: new	23	\@Acrlong: added	
alttree: moved hangindent and parindent assignments outside level test	322	\do@gls@link@checkfirsthyper	364
\makeglossaries: Moved definition of \glswrite to \makeglossaries ..	177	\@Acrshort: added	
\makenoidxglossaries: new	179	\do@gls@link@checkfirsthyper	363
\printglossary: changed to use new \@printglossary	194	\@GLS@: moved \glsifhyper	128
\printnoidxglossaries: new	195	moved check for first use to	
\printnoidxglossary: new	194	\@gls@link	128
\showgloclolist: new	262	\@GLSpl: moved \glsifhyper	131
\warn@noprintglossary: Activate warning in \makeglossaries	194	moved check for first use to	
\writeist: checked for definition of \glswrite	164, 168	\@gls@link	131
4.06 (2014-03-12)		\@GLS@: moved \glsifhyper	128
\@GLS@: added \glsifhyper	128	moved check for first use to	
\@GLSpl: added \glsifhyper	131	\@gls@link	128
\@Gls@: added \glsifhyper	128	\@GLspl@: moved \glsifhyper	130
\@GLspl@: added \glsifhyper	130	moved check for first use to	
\@gls@: added \glsifhyper	127	\@gls@link	130
\@gls@numbersdef: added hook to set toc title	34	\@acrlong: added	
\@gls@symbolsdef: added hook to set toc title	33	\do@gls@link@checkfirsthyper	364
\@glsdisp: added \glsifhyper	132	\@acrshort: added	
\@glspl@: added \glsifhyper	129	\do@gls@link@checkfirsthyper	362
General: added \glsifhyper	145–152	\@closegls: new	174
acronym: added hook to set toc title	17	\@gls@: moved \glsifhyper	127
acronyms: added hook to set toc title ...	17	moved check for first use to	
\glsdefmain: added hook to set toc title	16	\@gls@link	127
4.07 (2014-04-04)		\@gls@automake: new	174
\@glossarysection: added optional argument when using unstarred version	45	\@gls@doautomake: new	31
\@gls@noidx@do: added \global in case it's used in a tabular-like style	205	\@gls@field@link: added assignment of	
		\do@gls@link@checkfirsthyper	132
		\@gls@forbidtexext: new	63
		\@gls@hyp@opt: new	113
		\@gls@link: removed redundancy	115
		renamed \gls@type to \glstype ...	115
		\@gls@link@checkfirsthyper: new .	114
		\@glsdisp: moved \glsifhyper	132
		moved check for first use to	
		\@gls@link	132
		\@glspl@: moved \glsifhyper	129
		moved check for first use to	
		\@gls@link	129
		\@ignored@glossaries: new	66

General: added entrycounter option to		removed \@sGLStext	133
printgloss family	209	removed \@sGlstext	133
added nopostdot option to		removed \@sglstext	132
printgloss family	209	removed \@sGLSuseriii	142
added subentrycounter option to		removed \@sGlsuseriii	142
printgloss family	209	removed \@sglsuseriii	141
explicitly initialise hyper key	112	removed \@sGLSuserii	141
moved \glsifhyper	145–152	removed \@sGlsuserii	141
removed \@sACRlongpl	151	removed \@sglsuserii	141
removed \@sAcrlongpl	151	removed \@sGLSuseriv	143
removed \@sacrlongpl	150	removed \@sGlsuseriv	142
removed \@sACRlong	149	removed \@sglsuseriv	142
removed \@sAcrlong	149	removed \@sGLSuseri	140
removed \@sacrlong	148	removed \@sGlsuseri	140
removed \@sACRshortpl	147	removed \@sglsuseri	140
removed \@sAcrshortpl	147	removed \@sGLSsuservi	144
removed \@sacrshortpl	146	removed \@sGlsuservi	144
removed \@sACRshort	146	removed \@sglsuservi	144
removed \@sAcrshort	145	removed \@sGLSsuserv	143
removed \@sacrshort	145	removed \@sGlsuserv	143
removed \@sgls@link	114	removed \@sglsuserv	143
removed \@sGLSdescplural	138	removed \@sGLS	128
removed \@sGlsdescplural	138	removed \@sGls	127
removed \@sglsdescplural	137	removed \@sgls	126
removed \@sGLSdesc	137	removed \@thirdofthree (defined in	
removed \@sGlsdesc	137	kernel)	126
removed \@sglsdesc	136	removed sPGLS	270
removed \@sglsdisp	131	removed sPgls	268
removed \@sGLSfirstplural	135	removed spgls	267
removed \@sGlsfirstplural	135	removed sPGLSpl	271
removed \@sglsfirstplural	135	removed sPglspl	269
removed \@sGLSfirst	134	removed spglspl	268
removed \@sGlsfirst	133	\ACRfull: removed \s@ACRfull	224
removed \@sglsfirst	133	switched to using \@gls@hyp@opt ..	224
removed \@sGLSname	136	\Acrfull: removed \s@Acrfull	224
removed \@sGlsname	136	switched to using \@gls@hyp@opt ..	224
removed \@sglsname	136	\acrfull: removed \@sacrfull	223
removed \@sGLSplural	135	switched to using \@gls@hyp@opt ..	223
removed \@sGlsplural	134	\ACRfullpl: removed \s@ACRfullpl ..	226
removed \@sglsplural	134	switched to using \@gls@hyp@opt ..	226
removed \@sGLSpl	130	\Acrfullpl: removed \s@Acrfullpl ..	225
removed \@sGlspl	130	switched to using \@gls@hyp@opt ..	225
removed \@sglspl	129	\acrfullpl: removed \s@acrfullpl ..	225
removed \@sGLSsymbolplural	139	switched to using \@gls@hyp@opt ..	225
removed \@sGlsymbolplural	139	\ACRlong: switched to using	
removed \@sglsymbolplural	139	\@gls@hyp@opt	149
removed \@sGLSsymbol	139	\Acrlong: switched to using	
removed \@sGlsymbol	138	\@gls@hyp@opt	149
removed \@sglsymbol	138		

\acrlong: switched to using \@gls@hyp@opt	148	\glsdohyperlink: new	125
\ACRlongpl: switched to using \@gls@hyp@opt	151	\glsdohypertarget: new	125
\Acrlongpl: switched to using \@gls@hyp@opt	151	\glsenablehyper: added \KV@glslink@hypertrue to definition	126
\acrlongpl: switched to using \@gls@hyp@opt	150	\GLSfirst: switched to using \@gls@hyp@opt	134
\ACRshort: switched to using \@gls@hyp@opt	146	\Glsfirst: switched to using \@gls@hyp@opt	133
\Acrshort: switched to using \@gls@hyp@opt	145	\glsfirst: switched to using \@gls@hyp@opt	133
\acrshort: switched to using \@gls@hyp@opt	144	\GLSfirstplural: switched to using \@gls@hyp@opt	135
\ACRshortpl: switched to using \@gls@hyp@opt	147	\Glsfirstplural: switched to using \@gls@hyp@opt	135
\Acrshortpl: switched to using \@gls@hyp@opt	147	\glsfirstplural: switched to using \@gls@hyp@opt	135
\acrshortpl: switched to using \@gls@hyp@opt	146	\glsifhyper: deprecated	113
\forallacronyms: new	56	\glslink: switched to using \@gls@hyp@opt	114
\GLS: switched to using \@gls@hyp@opt	128	\glslinkcheckfirsthyperhook: new	115
\Gls: switched to using \@gls@hyp@opt	127	\glslinkvar: new	113
\gls: switched to using \@gls@hyp@opt	126	\GLSname: switched to using \@gls@hyp@opt	136
\glsdefglossaryentry: added check for ignored glossary	86	\Glsname: switched to using \@gls@hyp@opt	136
\gls@istfilebase: new	41	\glsname: switched to using \@gls@hyp@opt	136
\glsaddkey: removed \@sGLS@user@<key>	80	\GLSpl: switched to using \@gls@hyp@opt	130
removed \@sGls@user@<key>	80	\Glspl: switched to using \@gls@hyp@opt	130
removed \@sgls@user@<key>	79	\glspl: switched to using \@gls@hyp@opt	129
switched to using \@gls@hyp@opt	79, 80	\GLSplural: switched to using \@gls@hyp@opt	134
\GLSdesc: switched to using \@gls@hyp@opt	137	\Glsplural: switched to using \@gls@hyp@opt	134
\Glsdesc: switched to using \@gls@hyp@opt	137	\glsplural: switched to using \@gls@hyp@opt	134
\glsdesc: switched to using \@gls@hyp@opt	136	\glsspace: new	224
\GLSdescplural: switched to using \@gls@hyp@opt	138	\GLSsymbol: switched to using \@gls@hyp@opt	139
\Glsdescplural: switched to using \@gls@hyp@opt	137	\Glsymbol: switched to using \@gls@hyp@opt	138
\glsdescplural: switched to using \@gls@hyp@opt	137	\glssymbol: switched to using \@gls@hyp@opt	138
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	126	\GLSsymbolplural: switched to using \@gls@hyp@opt	139
\glsdisp: switched to using \@gls@hyp@opt	131		

\Glsymbolplural: switched to using \@gls@hyp@opt	139	altlongragged4col: fixed bug that displayed description instead of symbol	295
\glssymbolplural: switched to using \@gls@hyp@opt	139	\newglossary: added starred version ..	63
\GLStext: switched to using \@gls@hyp@opt	133	\newignoredglossary: new	66
\Glstext: switched to using \@gls@hyp@opt	133	\ns@newglossary: added \@glotype@<name>@log	64
\glstext: switched to using \@gls@hyp@opt	132	new	64
\glstreenamefmt: new	315	\p@gls@hyp@opt: new	113
\GLSuseri: switched to using \@gls@hyp@opt	140	\PGLS: changed to use \@gls@hyp@opt	270
\Glsuseri: switched to using \@gls@hyp@opt	140	\PglS: changed to use \@gls@hyp@opt	268
\glsuseri: switched to using \@gls@hyp@opt	140	\pgls: changed to use \@gls@hyp@opt	267
\GLSuserii: switched to using \@gls@hyp@opt	141	\PGLSpl: changed to use \@gls@hyp@opt	271
\Glsuserii: switched to using \@gls@hyp@opt	141	\PglSpl: changed to use \@gls@hyp@opt	269
\glsuserii: switched to using \@gls@hyp@opt	140	\pglSpl: changed to use \@gls@hyp@opt	268
\GLSuseriii: switched to using \@gls@hyp@opt	142	\s@gls@hyp@opt: new	113
\Glsuseriii: switched to using \@gls@hyp@opt	142	\s@newglossary: new	64
\glsuseriii: switched to using \@gls@hyp@opt	141	automake: new	30
\GLSuseriv: switched to using \@gls@hyp@opt	143	4.09 (2014-08-12)	
\Glsuseriv: switched to using \@gls@hyp@opt	142	\glsaddkey: fixed bug in user commands	79
\glsuseriv: switched to using \@gls@hyp@opt	142	4.10 (2014-08-27)	
\GLSuserv: switched to using \@gls@hyp@opt	143	\@Gls@acentryname: new	153
\Glsuserv: switched to using \@gls@hyp@opt	143	\@Gls@entryname: new	153
\glSuserv: switched to using \@gls@hyp@opt	143	\@gls@glossary: Renamed \@glossary to \@gls@glossary	184
\glsuservi: switched to using \@gls@hyp@opt	144	\glSpercentchar: new	163
\glSuservi: switched to using \@gls@hyp@opt	144	\glstildechar: new	163
\glSuservi: switched to using \@gls@hyp@opt	144	alttree: moved space after symbol	322, 323
\ifignoredglossary: new	66	4.11 (2014-09-01)	
		\@do@esc@wrglossary: added hook ..	189
		sanitize: none option	25
		\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	185
		\glsaddprotectedpagefmt: new	186
		\glsbackslash: new	163
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute	50
		\@gls@adjustmode: new	161
		\@gls@nottranslatorhook: removed ...	26
		\@gls@toc: added \protect to \numberline	46
		\@gls@usetranslator: new	26
		\glsacrpluralsuffix: new	38
		\glsadd: added check for vertical mode	161

\glsaddallunused: replaced @gobble with glsignore	162	\glsunset: switched to \@glsunset ...	94
\glsifusedtranslatordict: new	26	4.15 (2015-03-16)	
\glsignore: new	162	General: bug fix replaced \@glo@type with \glstype	151
\glsupacrpluralsuffix: new	38	4.16 (2015-06-18)	
\ProvidesGlossariesLang: new	38	\glsaddstoragekey: new	77
\RequireGlossariesLang: new	38	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	365
\indexspace: new	277, 297, 315	\@ACRshort: added \glspostlinkhook	364
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	364
\@@glslocalreset: new	95	\@Acrshort: added \glspostlinkhook	363
\@@glslocalunset: new	94	\@GLS@: added \glspostlinkhook ...	129
\@@glsreset: new	95	\@GLSpl: added \glspostlinkhook ..	131
\@@glsunset: new	94	\@Gls@: added \glspostlinkhook ...	128
\@@newglossaryentry@defcounters: new	96	\@Glspl@: added \glspostlinkhook .	130
\@cGls: new	99	\@acrlong: added \glspostlinkhook	364
\@cGls@: new	99	\@acrshort: added \glspostlinkhook	363
\@cGlspl@: new	100	\@gls@: added \glspostlinkhook ...	127
\@cgls: new	99	\@gls@@link: added \glspostlinkhook	114
\@cgls@: new	99	\@gls@field@link: added \glspostlinkhook	132
\@cglspl: new	100	\@gls@link: moved definition of \glsifhyperon outside of this macro	116
\@cglspl@: new	100	\@glsdisp: added \glspostlinkhook	132
\@gls@entry@count: new	99	\@glspl@: added \glspostlinkhook .	129
\@gls@increment@currcount: new ...	98	General: added \glspostlinkhook	145–152
\@gls@local@increment@currcount: new	98	\glsacspace: new	231
\@gls@write@entrycounts: new	98	\glsadd: changed \@do@wrglossary to \@do@wrglossary	161
\@glslocalreset: new	95	\glsfielddef: new	82
\@glslocalunset: new	94	\glsfieldedef: new	81
\@glsreset: new	95	\glsfieldfetch: new	82
\@glsunset: new	94	\glsfieldgdef: new	81
\@newglossaryentry@defcounters: new	90	\glsfieldxdef: new	81
\cGls: new	99	\glsifhyperon: moved definition of \glsifhyperon	115
\cgls: new	99	\glslinkpostsetkeys: new	115
\cGlsformat: new	100	\glspostlinkhook: new	114
\cglsformat: new	99	\glswriteentry: new	185
\cGlspl: new	100	\ifglsfieldcseq: new	83
\cglspl: new	100	\ifglsfielddefeq: new	83
\cGlsplformat: new	101	\ifglsfieldeq: new	82
\cglsplformat: new	100	long-sp-short: new	230
\gls@defdocnewglossaryentry: new .	74	\showglofield: new	263
\glsenableentrycount: new	96	4.18 (2015-09-09)	
\glslocalreset: switched to \@glslocalreset	93	General: split mfirstuc into separate bundle	4
\glslocalunset: switched to \@glslocalunset	94		
\glsreset: switched to \@glsreset ...	93		

4.19 (2015-10-31)	\glstreenamebox: new	321	\gls@arabicpage: new	186
4.19 (2015-11-22)	\@gls@link@nocheckfirsthyper: new	132	\gls@protected@pagefmts: added	
	\@gls@preglossaryhook: new	195	arabic to list	185
	\@printglossary: added		\glstrytitlecase: new	156
	\@gls@preglossaryhook	196	\glsfindwidesttoplevelname: new	321
	\do@glsglisablehyperinlist: new	115	\glslistgroupheaderfmt: new	277
	\doifglossarynoexistsordo: new	59	\glslistnavigationitem: new	277
	\gls@gobbleopt: new	63	\glstreegroupheaderfmt: new	315
	\glsdoifexistsordo: new	58	\glstreenavigationfmt: new	315
4.20 (2015-11-30)			\ifglswrallowprimitivemods: new	187
	\@gls@link: added		list: fixed missing space before	
	\@gls@setdefault@glslink@opts	115	description	277
	added \glsdonohyperlink when		long: fixed typo in \glossentrydesc	281
	hyperlink is suppressed	116	super4col: fixed bug in \glossentry	306
	\@gls@setdefault@glslink@opts:		4.23 (2016-04-30)	
	new	115	\glscurrentfieldvalue: new	62
	\gls@checkseeallowed@preambleonly:		\ifglshasfield: added	
	new	69	\glscurrentfieldvalue	61, 62
	\glsdonohyperlink: new	125	altlongragged4col: check for	
4.21 (2016-01-24)			nogroupskip changed	295
	\@printglossary: warn if no style has		altsuperragged4col: check for	
	been set	195	nogroupskip changed	314
	General: changed checkfirsthyper		long: check for nogroupskip changed	281
	assignment	145–151	long-booktabs: check for nogroupskip	
	\glossarystyle: set default style if not		changed	287
	already set	218	long3col: check for nogroupskip	
	\glsltpenaltycheck: new	290	changed	283
	\glspatchLToutput: new	290	long3col-booktabs: check for	
	\glspenaltygroupskip: new	290	nogroupskip changed	288
	altlong4col-booktabs: new	288	long4col: check for nogroupskip	
	altlongragged4col-booktabs: new	289	changed	284
	long-booktabs: new	287	long4col-booktabs: check for	
	long3col-booktabs: new	287	nogroupskip changed	288
	long4col-booktabs: new	288	longragged: check for nogroupskip	
	longragged-booktabs: new	289	changed	292
	longragged3col-booktabs: new	289	longragged3col: check for nogroupskip	
	\setglossarystyle: set default style if		changed	293
	not already set	217	super: check for nogroupskip changed	303
4.22 (2016-04-19)			super3col: check for nogroupskip	
	\@do@esc@wrglossary: added check		changed	305
	for \@arabic	189	super4col: check for nogroupskip	
	added test to allow temporary primitive		changed	307
	modifications and added arabic case	189	superragged: check for nogroupskip	
	mcolalttreespannav: new	302	changed	310
	mcolindexspannav: new	298	superragged3col: check for	
	mcoltreononamespannav: new	300	nogroupskip changed	312
	mcoltreespannav: new	299	4.24 (2016-05-27)	
			\@gls@extramakeindexopts: new	173

\@gls@glossary: added check for debug mode	184	\gls@set@xr@key: new	69
\@gls@see@noindex: new	6	\gls@xr@key: new	69
debug: new	5	\GlsAddXdyLocation: bug fix: changed #1 to #2	53
seenoinindex: new	6	\glsnoidxstripaccents: added \a ...	23
\glsnomakeindexwarning: new	47	added \TH, \dh and \DH	24
\GlsSetQuote: new	170	4.31 (2017-08-10)	
\GlsSetWriteIstHook: new	170	nolist: added check for “list” style	10
4.25 (2016-06-09)		4.31 (2017-09-10)	
\@gls@enablesavenonumberlist: new	70	style: changed \renewcommand to \def .	8
\@gls@initnonumberlist: new	70	4.32 (2017-08-24)	
\@gls@savenonumberlist: new	70	\@glsnavhypertarget: new	272
4.26 (2016-10-12)		\@glsshowtarget: new	6
\@glossary@default@style: added		\glsshowtarget: new	6
check for classicthesis	8	4.33 (2017-09-20)	
mcolindex: replaced \@idxitem with		\@do@esc@wrglossary: added	
\glstreeitem	297	\gls@the and \gls@number	189
mcolindexspannav: replaced \@idxitem		renamed from	
with \glstreeitem	298	\@do@esc@wrglossary	188
\glstreechildpredesc: new	316	\@do@noesc@wrglossary: new	187
\glstreeitem: new	315	\@do@wrglossary: changed to check	
\glstreepredesc: new	316	for esclocations	187
\glstreesubitem: new	316	\@gls@missinglang@warn: new	20
\glstreesubsubitem: new	316	\GlsSetXdyFirstLetterAfterDigits:	
4.28 (2017-01-07)		added starred version	163
\glspatchtabularx: new	93	\GlsSetXdyNumberGroupOrder: new .	163
4.29 (2017-01-19)		esclocations: new	9
\@gls@noidx@do: current letter group		4.34 (2017-11-03)	
assignment made global	206	mcolalttreespannav: removed spurious	
\@print@noidx@glossary: moved		space	302
definition of		\glsshowtarget: modified to check for	
\@gls@currentlettergroup outside		math mode and inner	6
of theglossary environment	204	4.35 (2017-11-14)	
General: added check for		\glsadd: added \@gls@setsort (in case	
\@glstr@doaccsupp	345	of sort=use)	161
\glsnavhyperlinkname: new	272	4.36 (2018-03-07)	
4.30 (2017-06-11)		\@gls@glossary: removed \index ...	184
\@glo@autosee: new	90	4.37 (2018-04-07)	
\@glo@autoseehook: new	90	\gls@begindocdefs: new	75
\@glo@check@sortallowed: new	13	4.38 (2018-05-10)	
\@gls@noidx@do: letter group		\@gls@define@glossaryentrycounter:	
assignment made global	206	added check for existence of	
\@gls@setupsort@def: added check for		glossaryentry counter	11
register	14	new	11
\@gls@setupsort@none: new	15	\@gls@define@glossarysubentrycounter:	
\@xdycrossrefhook: new	52	new	12
\@xdylocationclassorder: bug fix:		prepended \currentglossary. to	
changed \edef to \def	53	\theHglossarysubentry and	
\glosortentrieswarning: new	20	removed spurious eol space	12

\glsaccsupp: added braces around actual text argument	351	kernelglossredefs: new	32
\glsentrycounterlabel: bug fix: move conditional inside command	211	\glossary: added warning	32
\GlsEntryCounterLabelPrefix: new	209	\gls@original@glossary: new	31
\glsentryitem: bug fix: move conditional inside command	212	\gls@original@makeglossary: new ..	31
\glsrefentry: bug fix: move conditional inside command	211	\makeglossaries: removed redefinition of \makeglossary	177
\glsresetsubentrycounter: bug fix: move conditional inside command	210, 211	\makeglossary: added warning	32
\glsstepentry: bug fix: move conditional inside command	211	nonumberlist: changed \val and \nr to \gls@nonumberlist@val and \gls@nonumberlist@nr	69
\glsstepsubentry: bug fix: move conditional inside command	211	translate: changed \val and \nr to \gls@translate@val and \gls@translate@nr	27
\glssubentrycounterlabel: bug fix: move conditional inside command .	211	numberedsection: changed \val and \nr to \gls@numberedsection@val and \gls@numberedsection@nr	8
\glssubentryitem: bug fix: move conditional inside command	212	4.42 (2019-01-06)	
\showglongnameaccess: bug fix: corrected field (was showing text access field)	381	\@gls@@automake@immediate: new ..	176
4.40 (2018-06-01)		\@gls@automake@immediate: new ...	175
\istfile: changed \def to \providecommand	183	\gls@automake@nr: new	30
\makenoidxglossaries: false	179	\glsfieldedef: changed from \edef to \protected@csedef	81
4.41 (2018-07-23)		\glsfieldxdef: changed from \edef to \protected@csxdef	81
\@gls@override@glossary: new	32	\ifglsautomake: now defined explicitly instead of through boolean key	30
General: changed \val and \nr to \gls@numberedsection@val and \gls@numberedsection@nr	208	noglossaryindex: new	34
debug: changed \val and \nr to \gls@debug@val and \gls@debug@nr	5	automake: switch from boolean to choice	30
seenoindex: changed \val and \nr to \gls@seenoindex@val and \gls@seenoindex@nr	6	4.42 (??)	
		altlong4col-booktabs: removed superfluous \glspatchLToutput .	288
		4.43 (2019-09-28)	
		\glsnoidxstripaccents: add check for LaTeX version 2019/10/01	24
		4.44 (2019-12-06)	
		\@glsprefix@record@hook: new	267

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	
\!	121, 122
\"	23, 119–122, 124
\#	166, 167
\%	163, 169, 329, 330
\&	38, 160
\'	23
\.	11, 23
\=	23
\?	119–121, 171
\@	75
\@@delimN	220
\@@do@@wrglossary	179, 187, 190
\@@do@esc@wrglossary	187
\@@do@noesc@wrglossary	187
\@@do@wrglossary	161, 185
\@@glo@assign@sortkey	180
\@@glo@list	56
\@@glo@sort	23
\@@glo@type	194, 195
\@@glossarysec	7, 45, 46
\@@glossaryseclabel	8, 45, 46, 208
\@@glossarysecstar	8, 45, 208
\@@gls@checkactual	123
\@@gls@checkbar	122
\@@gls@checkescactual	121
\@@gls@checkescbar	121
\@@gls@checkesclevel	122
\@@gls@checkescquote	120, 172, 173
\@@gls@checklevel	123
\@@gls@checkquote	119, 120, 170, 171
\@@gls@default@entryfmt	102, 111
\@@gls@expand@field	21, 73, 74, 78, 79, 242, 244, 246, 248, 250, 253, 255, 376–380
\@@gls@extramakeindexopts	170, 177
\@@gls@fixbraces	192
\@@gls@noexpand@field	21, 72, 73
\@@gls@noidx@no@sanitizesort	23
\@@gls@noidx@nosanitizesort	182
\@@gls@nosanitizesort	22, 182
\@@gls@sanitizesort	22, 182
\@@gls@xdycheckbackslash	124, 125
\@@gls@xdycheckquote	124
\@@gls@localreset	95, 97
\@@gls@localunset	94, 97
\@@glsreset	95, 97
\@@glsunset	94, 97
\@@newglossaryentry@defcounters	96
\@@this@glo@	57
\@ACRfull	224
\@ACRfullpl	226
\@ACRlong	149, 225
\@ACRlongpl	151, 226
\@ACRshort	146, 225
\@ACRshortpl	147, 148, 226
\@Acrfull	224
\@Acrfullpl	225
\@Acrlong	149, 224
\@Acrlongpl	151, 225
\@Acrshort	145
\@Acrshortpl	147
\@Alph	186, 189
\@GLS	128
\@GLS@	128, 270
\@GLSdesc	137
\@GLSdesc@	137
\@GLSdescplural	138
\@GLSdescplural@	138
\@GLSfirst	134
\@GLSfirst@	134
\@GLSfirstplural	135
\@GLSfirstplural@	135
\@GLSname	136
\@GLSname@	136

\@GLSpl	130	\@Glsuseri	140
\@GLSpl@	130, 131, 271	\@Glsuseri@	140
\@GLSplplural	134, 135	\@Glsuserii	141
\@GLSplplural@	135	\@Glsuserii@	141
\@GLSsymbol	139	\@Glsuseriii	142
\@GLSsymbol@	139	\@Glsuseriii@	142
\@GLSsymbolplural	139	\@Glsuseriv	142
\@GLSsymbolplural@	139, 140	\@Glsuseriv@	142
\@GLStext	133	\@Glsuserv	143
\@GLStext@	133	\@Glsuserv@	143
\@GLSuseri	140	\@Glsuservi	144
\@GLSuseri@	140	\@Glsuservi@	144
\@GLSuserii	141	\@Mi	290
\@GLSuserii@	141	\@PGLS	270
\@GLSuseriii	142	\@PGLS@	270
\@GLSuseriii@	142	\@PGLSpl	271
\@GLSuseriv	143	\@PGLSpl@	271
\@GLSuseriv@	143	\@Pgl	268
\@GLSuserv	143	\@Pgl@	269
\@GLSuserv@	143, 144	\@Pglspl	269
\@GLSuservi	144	\@Pglspl@	269
\@GLSuservi@	144	\@Roman	186, 189
\@Gls	127	\@acrfull	223
\@Gls@	97, 99, 127, 269	\@acrfullpl	225
\@Gls@acrentryname	227	\@acrlong	148, 224
\@Gls@entry@field	79, 153–158	\@acrlongpl	150, 225
\@Gls@entryname	153, 227	\@acrshort	145, 224
\@GlsSetXdyFirstLetterAfterDigits	163	\@acrshortpl	146, 225
\@GlsSetXdyNumberGroupOrder	163, 164	\@addtoacronymlists	18
\@Glsdesc	137	\@after	18
\@Glsdesc@	137	\@afterheading	278, 333
\@Glsdescplural	137, 138	\@alph	186, 189
\@Glsdescplural@	138	\@arabic	186, 189
\@Glsfirst	133	\@auxout	62,
\@Glsfirst@	133, 134		64, 98, 177, 179, 180, 182, 194, 197, 198, 273
\@Glsfirstplural	135	\@backslashchar	118, 124, 125
\@Glsfirstplural@	135	\@before	18
\@Glsname	136	\@bsphack	185
\@Glsname@	136	\@cGls	99
\@Glspl	130	\@cGls@	97, 99
\@Glspl@	98, 100, 130, 270	\@cGlspl	100
\@Glsplural	134	\@cGlspl@	98, 100
\@Glsplural@	134	\@cclv	290, 291
\@Glssymbol	138	\@cgls	99
\@Glssymbol@	138	\@cgls@	97, 99
\@Glssymbolplural	139	\@cglspl	100
\@Glssymbolplural@	139	\@cglspl@	97, 100
\@Glstext	133	\@chapter	36
\@Glstext@	133	\@classoptionslist	34

<code>\@closegls</code>	174, 175	<code>\@glo@autoseehook</code>	90
<code>\@colht</code>	290	<code>\@glo@check@mkidxrangechar</code>	117, 118, 190, 326, 327
<code>\@colroom</code>	290, 291	<code>\@glo@check@sortallowed</code> ..	13–16, 178, 182
<code>\@currentlabelname</code>	8, 208	<code>\@glo@childlist</code>	199
<code>\@curroptions</code>	34	<code>\@glo@counter</code>	69, 85, 88
<code>\@declaredoptions</code>	34	<code>\@glo@counterprefix</code>	183, 187, 190, 191, 217, 220
<code>\@delimN</code>	220	<code>\@glo@default@sorttype</code> ..	13, 180, 201, 202
<code>\@delimR</code>	219	<code>\@glo@defaultcounter</code>	88
<code>\@disable@onlypremakeg</code>	178	<code>\@glo@desc</code>	67, 84–86, 89
<code>\@disable@premakecs</code>	36	<code>\@glo@descaccess</code>	347–349
<code>\@disabled@gl saddxdycounters</code>	49	<code>\@glo@descplural</code>	67, 84, 85
<code>\@do@addcounter</code>	48	<code>\@glo@descpluralaccess</code>	347–349
<code>\@do@auxoutstuff</code>	197, 198	<code>\@glo@do@sortentries</code>	198
<code>\@do@glossentry</code>	213, 345, 346	<code>\@glo@entry</code>	162
<code>\@do@gls@getcounterprefix</code>	187, 190	<code>\@glo@entryprefix</code>	265
<code>\@do@gls@islistofacronyms</code>	18	<code>\@glo@entryprefixfirst</code>	265
<code>\@do@glssee</code>	90	<code>\@glo@entryprefixfirstplural</code> ..	265, 266
<code>\@do@ifinlist</code>	47, 48	<code>\@glo@entryprefixplural</code>	265
<code>\@do@newglossaryentry</code>	227, 241–248, 250, 252–255, 257, 376–380	<code>\@glo@esclabel</code>	91, 92
<code>\@do@seeglossary</code>	179, 192	<code>\@glo@etext</code>	102–104
<code>\@do@subglossentry</code>	214, 346	<code>\@glo@first</code>	68, 85, 88, 89, 253, 380
<code>\@do@wrglossary</code>	116	<code>\@glo@firstaccess</code>	346, 348, 349
<code>\@do@writeaux@info</code>	193, 194	<code>\@glo@firstplural</code>	68, 85, 88, 380
<code>\@ehc</code>	290	<code>\@glo@firstpluralaccess</code>	347–349
<code>\@empty</code>	11, 15, 18, 31, 34–36, 47, 49, 52, 55, 56, 86, 87, 92, 116, 117, 127–131, 145–151, 164, 167, 169, 170, 174–176, 183, 185, 191, 217, 242, 244, 246–250, 252, 253, 255, 257, 327, 329, 331, 363–365	<code>\@glo@grabfirst</code>	204
<code>\@end@fixbraces</code>	192	<code>\@glo@label</code>	71, 72, 78, 79, 81–83, 85–90, 96, 160, 265, 266, 321, 349
<code>\@endfortrue</code>	28, 59, 77, 273	<code>\@glo@list</code>	90
<code>\@esphack</code>	185	<code>\@glo@long</code>	60, 71, 86, 88
<code>\@expandtwoargs</code>	34	<code>\@glo@longaccess</code>	347–349
<code>\@firstofone</code>	23	<code>\@glo@longpl</code>	71, 86, 88, 241, 244, 245, 248, 250, 253–255, 376
<code>\@firstofthree</code>	113, 126, 127, 129, 131, 145, 147, 148, 150, 363–365	<code>\@glo@longpluralaccess</code>	347–349
<code>\@firstoftwo</code>	26, 27, 76, 77, 113, 129–131, 147, 148, 150, 151	<code>\@glo@name</code>	13, 67, 72, 85, 87–89
<code>\@for</code>	28, 34, 36, 48, 49, 56, 76, 77, 118, 164–166, 176–178, 186, 193, 198, 199, 228, 242, 245, 246, 249, 251, 254, 256, 258, 273, 274, 327	<code>\@glo@no@assign@sortkey</code>	178
<code>\@glo@@desc</code>	89	<code>\@glo@nonumberlist</code>	70
<code>\@glo@@symbol</code>	89	<code>\@glo@numfmt</code>	191, 327
<code>\@glo@access</code>	346, 348, 349, 351	<code>\@glo@parent</code> ..	15, 69, 85, 87, 88, 92, 199, 200
<code>\@glo@addchildren</code>	198, 203	<code>\@glo@plural</code>	67, 85, 87, 88, 378
<code>\@glo@assign@sortkey</code>	178, 180, 209	<code>\@glo@pluralaccess</code>	347–349
<code>\@glo@autosee</code>	90	<code>\@glo@prefix</code>	9, 69, 70, 85, 92, 117, 118, 190, 326, 327
		<code>\@glo@range</code>	190, 326, 327
		<code>\@glo@see</code>	69, 85, 90
		<code>\@glo@seeautonumberlist</code>	9, 69
		<code>\@glo@short</code>	60, 71, 86, 88, 379
		<code>\@glo@shortaccess</code>	347–349, 376–379

<code>\@glo@shortpl</code>	71, 86, 88, 241, 243–245, 248, 250, 253, 255, 376, 379	<code>\@gls@@automake@immediate</code>	177
<code>\@glo@shortpluralaccess</code>	347–349	<code>\@gls@@link</code>	114
<code>\@glo@sort</code>	13, 16, 22, 23, 67, 85, 88, 92	<code>\@gls@Hcounter</code>	116, 117
<code>\@glo@sortedinsert</code>	199, 200	<code>\@gls@ReturnAfterFi</code>	221
<code>\@glo@sortentries</code>	201, 202	<code>\@gls@access@display</code>	351–353
<code>\@glo@sorthandler@case</code>	202	<code>\@gls@actualchar</code> .	92, 121, 123, 168, 169, 330
<code>\@glo@sorthandler@letter</code>	201	<code>\@gls@addpredefinedattributes</code> .	164, 173
<code>\@glo@sorthandler@nocase</code>	202	<code>\@gls@adjustmode</code>	161
<code>\@glo@sorthandler@word</code>	201	<code>\@gls@automake</code>	178
<code>\@glo@sortinghandler</code>	198, 200	<code>\@gls@automake@immediate</code>	176
<code>\@glo@sortinglist</code>	198, 200, 202, 203	<code>\@gls@between</code>	274
<code>\@glo@sorttype</code>	180, 203, 205, 210	<code>\@gls@body</code>	153
<code>\@glo@storeentry</code>	13–15	<code>\@gls@checkactual</code>	119, 171
<code>\@glo@suffix</code>	117, 118, 190, 327	<code>\@gls@checkbar</code>	119, 171
<code>\@glo@symbol</code>	60, 68, 85, 89, 247, 252, 348	<code>\@gls@checkedmkidx</code>	118–124, 170–172
<code>\@glo@symbolaccess</code>	347–349, 379	<code>\@gls@checkescactual</code>	119, 171
<code>\@glo@symbolplural</code>	68, 85, 89	<code>\@gls@checkescbar</code>	119, 172
<code>\@glo@symbolpluralaccess</code>	347–349	<code>\@gls@checkescquote</code>	119, 171, 172
<code>\@glo@text</code>	67, 85, 88, 89, 127–132, 152–154, 248, 266, 378	<code>\@gls@checklevel</code>	119, 172
<code>\@glo@textaccess</code> ...	346, 348, 349, 376–379	<code>\@gls@checkmkidxchars</code> 91, 92, 117, 171, 179, 190, 192, 326, 327
<code>\@glo@thislabel</code>	91	<code>\@gls@checkquote</code>	119, 170, 171
<code>\@glo@thislettergrp</code>	204–206	<code>\@gls@classI</code>	165
<code>\@glo@thisvalue</code>	61, 62	<code>\@gls@classII</code>	165
<code>\@glo@tmp</code>	78, 79, 191	<code>\@gls@codepage</code>	198
<code>\@glo@type</code>	8, 15, 68, 85, 86, 88–90, 161, 162, 177, 183, 195– 199, 203, 204, 207, 208, 227, 242, 244, 246, 248, 250, 251, 253, 255, 257, 272, 274	<code>\@gls@counter</code> 112, 115–117, 161, 183, 190, 191, 327
<code>\@glo@types</code>	56, 57, 64, 95, 96, 161, 162, 176–178, 263, 321	<code>\@gls@counterwithin</code>	11, 12
<code>\@glo@useri</code>	70, 85, 88	<code>\@gls@ctr</code>	48
<code>\@glo@userii</code>	70, 85, 88	<code>\@gls@currentlettergroup</code>	204–206
<code>\@glo@useriii</code>	71, 85, 88	<code>\@gls@debugfalse</code>	5
<code>\@glo@useriv</code>	71, 85, 88	<code>\@gls@debugtrue</code>	5
<code>\@glo@userv</code>	71, 85, 88	<code>\@gls@declareoption</code> 9, 10, 16, 17, 20, 21, 26, 29, 30, 33, 34
<code>\@glo@uservi</code>	71, 85, 88	<code>\@gls@default</code>	101
<code>\@glodesc</code>	89	<code>\@gls@default@value</code> 60–62, 72, 73, 85, 87–89, 251, 265
<code>\@glolist@</code>	86	<code>\@gls@deffile</code>	75, 76
<code>\@gloname</code>	89	<code>\@gls@define@glossaryentrycounter</code> .	.. 12, 35, 209, 210
<code>\@glossary@default@style</code> 8, 10, 195, 217, 218, 258	<code>\@gls@define@glossarysubentrycounter</code>	35, 209, 210
<code>\@glossaryentryfield</code>	92	<code>\@gls@defsort</code>	13–15, 89
<code>\@glossarysection</code>	44	<code>\@gls@defsortcount</code>	13–15, 64
<code>\@glossarystyle</code>	195, 196, 208	<code>\@gls@do@acronymsdef</code>	17, 35, 66
<code>\@glossarysubentryfield</code>	92	<code>\@gls@do@indexdef</code>	34–36, 66
<code>\@gls</code>	126	<code>\@gls@do@numbersdef</code>	33–35, 66
<code>\@gls@</code>	97, 99, 126, 268, 269	<code>\@gls@do@symbolsdef</code>	33, 66
		<code>\@gls@do@symbolssdef</code>	35, 36

<code>\@gls@doautomake</code>	30, 31, 176, 178	<code>\@gls@langmod</code>	174, 175
<code>\@gls@docheckquotedef</code>	170–173	<code>\@gls@levelchar</code> ..	92, 122, 123, 168, 169, 330
<code>\@gls@docloadedfalse</code>	4	<code>\@gls@link</code> ..	114, 127–132, 145–152, 363–365
<code>\@gls@docloadedtrue</code>	4	<code>\@gls@link@checkfirsthyper</code>	127–131
<code>\@gls@dodedeflistparser</code>	178	<code>\@gls@link@label</code>	115, 243, 249
<code>\@gls@doentrycounterdef</code>	35, 36	<code>\@gls@link@nocheckfirsthyper</code> 132, 145–151	
<code>\@gls@doentrydef</code>	111	<code>\@gls@link@opts</code>	115, 243, 249
<code>\@gls@dolast</code>	192, 193	<code>\@gls@list</code>	273, 274
<code>\@gls@donext</code>	193	<code>\@gls@listsuffix</code>	47
<code>\@gls@donext@def</code>	160	<code>\@gls@loadlist</code>	10, 258
<code>\@gls@dosubentrycounterdef</code>	35, 36	<code>\@gls@loadlong</code>	10, 258
<code>\@gls@dotothiswrite</code>	174–176	<code>\@gls@loadsuper</code>	10, 258
<code>\@gls@elem</code>	273	<code>\@gls@loadtree</code>	10, 258
<code>\@gls@enablesavenonumberlist</code>	75	<code>\@gls@local@increment@currcount</code>	97
<code>\@gls@encapchar</code>		<code>\@gls@loclist</code>	180, 181, 205, 206
.....	121, 122, 168, 169, 191, 192, 327, 330	<code>\@gls@map</code>	76, 77
<code>\@gls@entry@count</code>	98	<code>\@gls@missinglang@warn</code>	20, 39
<code>\@gls@entry@field</code>		<code>\@gls@missingnumberlist</code>	89
.....	78, 79, 96, 97, 153–159, 349–351	<code>\@gls@noaccess</code>	351
<code>\@gls@escbsdq</code>	119, 169, 331	<code>\@gls@noexpand@fields</code>	74
<code>\@gls@expand@fields</code>	73, 74	<code>\@gls@nohyperlist</code>	19, 66, 115
<code>\@gls@expandonce</code>	74	<code>\@gls@noidx@do</code>	204
<code>\@gls@extramakeindexopts</code>	177	<code>\@gls@noidx@getgrouptitle</code>	179
<code>\@gls@fetchfield</code>	61	<code>\@gls@noidx@sanitizesort</code>	22, 182
<code>\@gls@field@link</code>	79, 80, 132–144	<code>\@gls@noidx@setsanitizesort</code>	25, 182
<code>\@gls@firsttok</code>	204	<code>\@gls@noidx@loclist@finalsep</code>	181
<code>\@gls@fixbraces</code>	90	<code>\@gls@noidx@loclist@prev</code>	181, 206, 207
<code>\@gls@forbidtexext</code>	64	<code>\@gls@noidx@loclist@sep</code>	181, 206, 207
<code>\@gls@get@counterprefix</code>	191	<code>\@gls@noref@warn</code>	179, 204
<code>\@gls@getbody</code>	153	<code>\@gls@numberlink</code>	220
<code>\@gls@getcounterprefix</code>	187, 190	<code>\@gls@numbersdef</code>	33
<code>\@gls@getgrouptitle</code>	179, 216, 274	<code>\@gls@numlist@lastsep</code>	160
<code>\@gls@glossary</code>	184, 185	<code>\@gls@numlist@nextsep</code>	160
<code>\@gls@gobbleopt</code>	63	<code>\@gls@numlist@sep</code>	160
<code>\@gls@grptitle</code>	216, 272, 274	<code>\@gls@old@chapter</code>	36
<code>\@gls@hyp@opt</code>	80,	<code>\@gls@oldnewglossaryentryposthook</code> .	348
99, 100, 114, 126–151, 223–226, 267–271		<code>\@gls@oldnewglossaryentryprehook</code> ..	348
<code>\@gls@hyp@opt@cs</code>	113	<code>\@gls@onlypremakeg</code>	36
<code>\@gls@hypergroup</code>	273	<code>\@gls@order</code>	174–176
<code>\@gls@ifinlist</code>	48	<code>\@gls@org@LT@output</code>	290
<code>\@gls@ifnotmeasuring</code>	93	<code>\@gls@org@glsnoidx@displayloc</code>	181
<code>\@gls@igtype</code>	66	<code>\@gls@org@glssseeformat</code>	181
<code>\@gls@increment@currcount</code>	97	<code>\@gls@override@glossary</code>	32, 33
<code>\@gls@indexdef</code>	34	<code>\@gls@patchtabularx</code>	93
<code>\@gls@initnonumberlist</code>	70, 85	<code>\@gls@preglossaryhook</code>	196
<code>\@gls@islistofacronyms</code>	18	<code>\@gls@prevlevel</code> .	301, 302, 321–324, 339, 340
<code>\@gls@keylist</code>	375	<code>\@gls@provide@newglossary</code>	64
<code>\@gls@keymap</code>	70, 76–79, 265, 348	<code>\@gls@quotechar</code>	120–123, 168–172, 330
<code>\@gls@label</code>	179, 183, 187, 190, 191	<code>\@gls@reference</code>	179, 180, 182

\@gls@removespaces	220, 221	\@glsAlphacompositor	42, 52, 328
\@gls@renewglossary	174	\@glsHlocref	187, 190
\@gls@replacementtext	351	\@glsacronymlists	18, 19, 56, 227, 228, 242, 244–246, 248–251, 253–258, 263
\@gls@rest	153	\@glsaddkey	78, 79
\@gls@restoreat	75, 76	\@glsaddstoragekey	77, 78
\@gls@roman	51, 327, 328	\@glsaddxdyattribute	48, 49
\@gls@sanitized@tmp	118	\@glsdefaultsort	13
\@gls@sanitizedesc	28	\@glsdesc	136
\@gls@sanitizesort	13	\@glsdesc@	136, 137
\@gls@sanitizesymbol	28, 29	\@glsdescplural	137
\@gls@saveentrycounter	116, 161	\@glsdescplural@	137
\@gls@savenonumberlist	70	\@glsdisp	131
\@gls@see@noindex	7, 69	\@glsentry	95, 96, 98
\@gls@setacrstyle	28, 29, 35	\@glsentrytitlecase	156
\@gls@setcounter	65	\@glsfirst	133
\@gls@setdefault@glslink@opts	115	\@glsfirst@	133
\@gls@setsort	13–16, 116, 161	\@glsfirstletter	163
\@gls@setupshortcuts	35	\@glsfirstplural	135
\@gls@sort	205	\@glsfirstplural@	135
\@gls@sort@A	200, 201	\@glshypernumber	219
\@gls@sort@B	200, 201	\@glsisacronymlistfalse	19
\@gls@startswithexpandonce	73	\@glsisacronymlisttrue	19
\@gls@storenonumberlist	70, 89	\@glslink	116, 126, 160, 272
\@gls@symbolsdef	33	\@glslocalreset	94, 97
\@gls@this	186	\@glslocalunset	94, 97
\@gls@thisHloc	191	\@glslocref	183, 187, 190, 191, 326, 327
\@gls@thisfield	61	\@glsminrange	164, 165, 328
\@gls@thislabel	59, 193, 202, 203	\@glsname	136
\@gls@thislist	160	\@glsname@	136
\@gls@thisloc	191	\@glsnavhypertarget	272
\@gls@thisval	77	\@glsnextpages	196
\@gls@title	44	\@glsnodesc	85, 86, 89
\@gls@tmp	15, 39, 52, 74, 118, 185, 273, 274	\@glsnoname	85, 87, 89
\@gls@tmpb	119–124, 170–172	\@glsnonextpages	196
\@gls@toc	45, 46	\@glsnumberformat	112, 115, 161, 183, 190, 191, 326, 327
\@gls@type	176, 178, 228, 242, 245, 246, 249, 251, 254, 256, 258, 321	\@glsopenfile	174, 183
\@gls@updatechecked	118, 119, 171, 172	\@glsorder	177
\@gls@usetranslator	27, 38	\@glspl	129
\@gls@value	73, 156	\@glspl@	97, 100, 129, 268, 270
\@gls@warnonglossdefined	21, 194	\@glsplural	134
\@gls@warnonthe glossdefined	21, 212	\@glsplural@	134
\@gls@write@entrycounts	98	\@glsprefix@record@hook	268–271
\@gls@writedef	75	\@glsreset	93, 97
\@gls@writeisthook	168, 170	\@glssee	90, 192
\@gls@xdy@locationlist	165	\@glsshowtarget	5, 6, 125
\@gls@xdycheckbackslash	118	\@glsymbol	138
\@gls@xdycheckquote	118	\@glsymbol@	138
\@gls@xref	192		

\@glssymbolplural	139	\@no@makeglossaries	178, 180
\@glssymbolplural@	139	\@no@post@desc	332
\@glstarget	126, 212, 273	\@nopostdesc	196
\@glstext	132	\@onelevel@sanitize	22, 51, 76, 92, 118, 163, 164, 167, 192, 194, 204, 328, 329
\@glstext@	132	\@onlypreamble	65, 75, 84, 98, 101, 179, 182
\@glunset	94, 97	\@onlypremakeg	41, 42, 48, 49, 53, 65, 170
\@gluseri	140	\@org@glossaryentrynumbers	195, 196
\@gluseri@	140	\@org@gl@assign@descplural	242, 250, 253, 255, 376, 379, 380
\@gluserii	140, 141	\@org@gl@assign@firstpl	241, 242, 244, 246, 248, 250, 253, 255, 376–380
\@gluserii@	141	\@org@gl@assign@plural	242, 244, 246, 248, 250, 253, 255, 376–380
\@gluseriii	141	\@org@gl@assign@symbolplural	242, 244, 246, 248, 253, 255, 377, 378, 380
\@gluseriii@	141	\@org@gl@numberformat	160
\@gluseriv	142	\@org@newglossaryentryprehook	84
\@gluseriv@	142	\@outputpage	290, 291
\@gluserv	143	\@p@glossarysection	44
\@gluserv@	143	\@pgls	267
\@gluservi	144	\@pgls@	267
\@gluservi@	144	\@pglspl	268
\@glswidestname	321–323, 339	\@pglspl@	268
\@glswritefiles	31	\@plus	277, 297, 315
\@glxtr@doaccsupp	345	\@print@glossary	194
\@glxtr@record	267	\@print@noidx@glossary	194
\@gobble	5, 13–15, 76, 93, 118, 162, 163, 166, 167, 179, 325, 329, 330	\@printgloss@setsort	178, 180, 195
\@idxitem	315	\@printglossary	194
\@ifclassloaded	4, 11, 44	\@roman	51, 327
\@ifl@t@r	24	\@secondofthree	113, 126, 128, 130, 145, 147, 149, 151, 363
\@ifnextchar	65, 113	\@secondoftwo	23, 26, 27, 39, 76, 77, 125– 128, 131, 145, 146, 148–150, 363–365, 383
\@ifpackageloaded	4, 8, 26–28, 38, 55, 93, 159, 170, 345	\@set@glo@numformat	191, 327
\@ifstar	63, 77, 78, 113, 163, 222	\@sglsaddkey	78
\@ifundefined	38, 273, 280, 291, 302, 309, 322, 323, 339, 353	\@sglsaddstoragekey	77, 78
\@ignored@glossaries	66	\@tabacckludge	23
\@input@	197	\@text@composite@x	23
\@istfilename	177	\@thirdofthree	113, 128, 131, 146, 148, 150, 151, 363
\@makecol	290, 291	\@this@attr	166, 167
\@makeglossary	177	\@this@childlabel	199
\@minus	277, 297, 315	\@this@counter	49
\@mkboth	44, 45	\@this@ctr	166
\@newglossary	62, 64	\@this@key	77
\@newglossaryentry@defcounters	90, 96	\@this@label	198
\@newglossaryentryposthook	78, 79, 90, 265, 348	\@this@cs	36
\@newglossaryentryprehook	78, 79, 84, 86, 265, 348	\@tmp	51, 328
\@nil	18, 90, 117–119, 153, 171, 172, 190, 192, 204, 205, 219–221, 326, 327		
\@nnil	18, 193		

<code>\bfseries</code>	282–285, 287, 288, 292–294, 296, 304–309, 311–315	<code>\csshow</code>	263
<code>\bgroup</code>	23, 84, 159, 195, 199	<code>\csuse</code>	40, 43, 63, 73, 79, 80, 111, 174–176, 199, 202, 203, 205, 207, 209, 218, 229, 266, 332–344
<code>bib2gls</code>	23, 188	<code>\csxdef</code>	89, 98
<code>booktabs</code> package	286–289	<code>\currentglossary</code>	11, 12, 43, 196
<code>\boolean</code>	256	<code>\currentglssubentry</code>	12, 211
<code>\boolfalse</code>	31	<code>\CurrentOption</code>	34, 35, 265, 345
<code>\booltrue</code>	31	<code>\CurrentTrackedLanguage</code>	39, 383, 384
<code>\bottomrule</code>	287, 288	<code>\CurrentTrackedTag</code>	39, 383, 384
<code>\box</code>	290	<code>\CustomAcronymFields</code>	257
C		<code>\CustomNewAcronymDef</code>	258
<code>\c</code>	23	D	
<code>\c@equation</code>	116	<code>\d</code>	23
<code>\c@glossaryentry</code>	11	<code>datatool</code> package	200
<code>\c@glossarysubentry</code>	12	<code>datatool-base</code> package	4
<code>\c@page</code>	186, 187, 189	<code>\day</code>	164, 168, 327, 330
<code>\catcode</code>	75	<code>\DeclareAcronymList</code>	17, 19, 227, 228, 242, 244, 246, 248, 251, 253, 255, 257
<code>\cGls</code>	99	<code>\DeclareListParser</code>	178
<code>\cgl</code>	99	<code>\DeclareOption</code>	9, 265, 345
<code>\cGlsformat</code>	97	<code>\DeclareOptionX</code>	9
<code>\cglformat</code>	97	<code>\DeclareRobustCommand</code>	40, 192, 193, 251, 351–353
<code>\cGlspl</code>	100	<code>\def</code>	8, 9, 11–16, 18, 22–24, 29, 30, 32, 33, 35–37, 40, 41, 44, 47, 50–55, 59, 63–65, 67–71, 74, 82, 84–89, 91, 93, 97–101, 111, 112, 115–125, 127–152, 160, 161, 164, 168, 170–172, 174–176, 178, 180, 181, 183, 186, 187, 189–192, 195, 198, 202, 204– 210, 216–227, 242, 244, 246–248, 250, 252, 253, 255, 257, 265, 268–271, 274– 276, 301, 302, 321–324, 326, 327, 330, 332, 339, 340, 345–348, 362–365, 376–380
<code>\cGlsplformat</code>	98	<code>\def@glx@dycheckbackslash</code>	124, 125
<code>\cglspformat</code>	97	<code>\DefaultNewAcronymDef</code>	242
<code>\char</code>	217	<code>\defglentryfmt</code>	64, 66, 111, 228, 241, 243, 245, 247, 249, 252, 254, 257
<code>classicthesis</code> package	8	<code>\define@boolkey</code>	7, 9, 11, 12, 17, 24, 25, 28, 29, 31, 112, 209
<code>\cleardoublepage</code>	46	<code>\define@choicekey</code>	5–8, 13, 25, 27, 29, 30, 33, 69, 208, 209
<code>\clearpage</code>	46	<code>\define@key</code>	8, 12, 19, 25, 29, 30, 67– 71, 78, 79, 112, 161, 207–209, 265, 346, 347
<code>\closeout</code>	75, 168, 170, 174, 183	<code>\DefineAcronymSynonyms</code>	35, 241
<code>\compatglossarystyle</code>	332–344	<code>\delimN</code>	167, 178, 206, 220, 329
<code>\compatibleglossentry</code>	214	<code>\delimR</code>	167, 219, 220, 329
<code>\compatiblesubglossentry</code>	214		
<code>\copy</code>	290, 291		
<code>\count@</code>	204, 205		
<code>\csdef</code>	21, 78–80, 90, 91, 96, 97, 198, 199, 219, 229, 331		
<code>\csedef</code>	98, 186		
<code>\csgdef</code>	43, 63, 66, 97, 98, 194, 207		
<code>\cslet</code>	70, 84, 91, 203		
<code>\csname</code>	13– 16, 23, 35, 37, 39, 40, 45, 46, 49, 51, 52, 55, 56, 59, 64, 65, 72, 73, 78–80, 82, 83, 86–92, 94, 95, 111, 115–117, 127–132, 145–152, 160, 161, 165–167, 171–174, 179, 183, 185, 186, 190–192, 195–197, 199, 208, 213, 214, 217, 218, 222, 258– 266, 273, 274, 321–323, 325–327, 339, 345, 346, 349, 350, 353, 363–365, 381, 382		

284, 287, 288, 290–292, 294, 295, 303, 305, 307, 310, 312, 314, 317, 318, 320– 323, 325–329, 331, 332, 337–340, 345, 351	glossaries-accsupp package 91, 345 glossaries-extra package 75, 166, 267, 345 \GlossariesWarning 5, 7, 20, 21, 24, 25, 32, 33, 43, 47, 58, 62, 69, 72, 99, 100, 111, 113, 159, 175, 176, 179– 182, 185, 191, 195, 196, 215, 218, 325, 345 \GlossariesWarningNoLine 5, 6, 20, 178, 180, 183, 198, 273 \glossary 31–33, 326, 327 glossary package 1, 32, 222 glossary styles: altlist 278, 279, 333 altlistgroup 279, 333 altlisthypergroup 279, 333 altlong4col 285, 286, 294, 335 altlong4col-booktabs 288, 290 altlong4colborder 286, 335 altlong4colheader 286, 288, 335 altlong4colheaderborder 286, 335 altlongragged4col 289, 294–296, 336 altlongragged4col-booktabs 289 altlongragged4colborder 296, 336 altlongragged4colheader 295, 336 altlongragged4colheaderborder 296, 337 altsuper4col 308, 313, 344 altsuper4colborder 308, 344 altsuper4colheader 308, 344 altsuper4colheaderborder 308, 344 altsuperragged4col 313, 314, 342 altsuperragged4colborder 314, 342 altsuperragged4colheader 314, 342 altsuperragged4colheaderborder 314, 342 alttree 301, 316, 321, 323, 339 alttreegroup 324, 340 alttreehypergroup 324, 340 index 8, 297, 315–318, 337 indexgroup 317, 337 indexhypergroup 317, 337 inline 332 list 8, 10, 277–279, 332 listdotted 279, 280, 333 listgroup 278, 332 listhypergroup 278, 333 long 281, 282, 287, 291, 333, 335 long-booktabs 287, 289 long3col 282, 283, 287, 334 long3col-booktabs 287, 289 long3colborder 283, 333
file types .aux 197 .glo 91 .ist 162, 173 .toc 46 .xdy 41 glo 264 \firstacronymfont . . . 110, 229–231, 237, 243, 247, 249, 252, 362, 366–368, 372, 373 \fmtversion 24 \footnote 237, 243, 373 \FootnoteNewAcronymDef 251 \forallglossaries . . . 57, 183, 194, 195, 321 \forallglsentries 95, 96, 98, 162 \ForEachTrackedDialect 39, 383, 384 \forglsentries 57, 59, 91, 202, 321 \forlistcsloop 198, 204 \forlistloop 181, 206	
G	
garamondx package 223 \gdef 15, 49, 64, 82, 87, 88, 185, 210, 273 \Genacrfullformat 109, 227–231, 237, 362, 366, 367, 373 \genacrfullformat 109, 110, 227–231, 237, 361, 362, 366, 367, 372 \GenericAcronymFields 227, 229–239, 366–369, 371, 372, 375 \Genplacrfullformat 109, 228, 230, 231, 237, 361, 367, 373 \genplacrfullformat 109, 110, 227, 228, 230, 231, 237, 361, 367, 373 \glo@desc 332 \glo@do@compare 200, 201 \glo@grabfirst 205 \glo@label 59, 91 \glo@list 91 \glo@name 213 \glo@parent 59 \glo@type 91 \glo@value 76 \global 14–16, 72, 75, 84, 89, 94, 95, 185, 196, 197, 205, 206, 210, 290, 291 \glolinkprefix 116, 160, 212 \glosortentrieswarning 20, 198 glossaries package 32, 34, 54, 55, 164, 258, 265, 277, 325, 345	

long3colheader	283 , 287 , 334	superragged3colheaderborder	313 , 342
long3colheaderborder	283 , 334	superraggedborder	310 , 341
long4col	284 , 285 , 288 , 334	superraggedheader	311 , 341
long4col-booktabs	288	superraggedheaderborder	311 , 342
long4colborder	285 , 335	tree	298 , 318 , 319 , 321 , 337
long4colheader	284 , 288 , 335	treegroup	299 , 319 , 338
long4colheaderborder	285 , 335	treehypergroup	319 , 338
longborder	281 , 334	treenoname	299 , 316 , 319 , 320 , 338
longheader	281 , 287 , 334	treenonamegroup	320 , 339
longheaderborder	282 , 334	treenonamehypergroup	320 , 339
longragged	289 , 291 – 293	glossary-hypernav package	162
longragged-booktabs	289	glossary-list package	8 , 10 , 277
longragged3col	289 , 293 , 294 , 336	glossary-long package ...	10 , 280 , 294 , 302 , 303
longragged3col-booktabs	289	glossary-longragged package	291
longragged3colborder	294 , 336	glossary-mcols package	296
longragged3colheader	294 , 336	glossary-super package ..	10 , 280 , 302 , 309 , 313
longragged3colheaderborder .	294 , 336	glossary-superragged package	309
longraggedborder	292 , 335	glossary-tree package	10 , 315
longraggedheader	292 , 336	\glossaryentry	191 , 192 , 327
longraggedheaderborder	293 , 336	glossaryentry (counter)	11 , 12 , 211 , 212
mcolalttree	301 , 341	\glossaryentryfield	
mcolalttreegroup	301 , 341	213 , 332 – 339 , 341 – 344 , 366
mcolalttreehypergroup ...	301 , 302 , 341	\glossaryentrynumbers	
mcolindex	297 , 340	..	9 , 167 , 195 , 196 , 205 , 206 , 209 , 210 , 329
mcolindexgroup	297 , 340	\glossaryheader	
mcolindexhypergroup	297 , 298 , 340	167 , 204 , 275 , 277 – 279 , 281 – 285 , 287 , 288 , 291 – 297 , 299 – 301 , 303 , 304 , 306 , 310 , 311 , 313 , 316 – 321 , 324 , 329
mcoltree	298 , 340	\glossarymark	44
mcoltreegroup	340	\glossaryname	16 , 39 , 40
mcoltreehypergroup	299 , 340	\glossarypostamble	167 , 204 , 329
mcoltreenoname	300 , 341	\glossarypreamble	166 , 204 , 329
mcoltreenonamegroup	300 , 341	\glossarysection	166 , 204 , 329
mcoltreenonamehypergroup ...	300 , 341	glossarysubentry (counter)	12 , 210 – 212
sublistdotted	333	\glossarysubentryfield	
super	303 , 304 , 311 , 343	214 , 332 – 339 , 341 – 344 , 366
super3col	304 – 306 , 343	\glossarytitle ..	166 , 195 , 196 , 204 , 208 , 329
super3colborder	305 , 343	\glossarytoctitle	8 , 16 , 17 , 33 , 34 , 37 , 40 , 44 , 166 , 195 , 204 , 208 , 329
super3colheader	305 , 343	\glossentry	91 , 196 , 206 , 214 , 215 , 275 , 277 – 282 , 284 , 292 , 293 , 295 , 303 , 305 , 306 , 310 , 311 , 313 , 316 , 318 , 319 , 322
super3colheaderborder	306 , 343	\Glossentrydesc	365
super4col	306 – 308 , 344	\glossentrydesc	
super4colborder	307 , 344	..	275 – 282 , 284 , 292 , 293 , 295 , 303 , 305 , 306 , 310 – 313 , 316 – 318 , 320 , 322 , 323 , 365
super4colheader	307 , 344	\glossentryname	
super4colheaderborder	307 , 344	..	275 – 282 , 284 , 292 , 293 , 295 , 303 , 305 , 306 , 310 , 311 , 313 , 316 – 319 , 322 , 323 , 365
superborder	303 , 343		
superheader	304 , 343		
superheaderborder	304 , 343		
superragged	309 , 311 , 341		
superragged3col	311 – 313 , 342		
superragged3colborder	312 , 342		
superragged3colheader	312 , 342		

\Glossentrysymbol	366	\gls@noidxglossary	179
\glossentrysymbol	275, 276, 284, 295, 306, 313, 316–318, 320, 322, 323, 366	\gls@nonumberlist@nr	69
\Gls	99, 222, 240	\gls@nonumberlist@val	69
\gls	32, 99, 179, 211, 222, 240	\gls@nosetquote	85, 168, 170, 173
\gls@Alphpage	186, 189	\gls@number	189
\gls@alphpage	186, 189	\gls@numberedsection@nr	8, 208
\gls@arabicpage	186, 189	\gls@numberedsection@val	8, 208
\gls@assign@desc	84, 89	\gls@numberpage	186, 189
\gls@assign@descplural	242, 250, 253, 255, 376, 379, 380	\gls@org@glossaryentryfield	196
\gls@assign@field	74, 78, 79, 84, 86, 88, 89, 265, 266	\gls@org@glossarysubentryfield	196
\gls@assign@firstpl	241, 242, 244, 246, 248, 250, 253, 255, 376–380	\gls@org@insert	247, 249, 252
\gls@assign@plural	242, 244, 246, 248, 250, 253, 255, 376–380	\gls@orgAlph	189
\gls@assign@symbolplural	242, 244, 246, 248, 253, 255, 377, 378, 380	\gls@orgalph	189
\gls@automake@nr	30, 176	\gls@orgarabic	189
\gls@automake@val	30	\gls@orgnumber	189
\gls@begindocdefs	75	\gls@orgRoman	189
\gls@checkisacronymlist	114	\gls@orgromannumeral	189
\gls@checkseeallowed	69, 75, 178, 179	\gls@orgthe	189
\gls@checkseeallowed@preambleonly ..	75	\gls@original@glossary	33
\gls@codepage	55, 174, 175, 198	\gls@original@makeglossary	33
\gls@debug@nr	5, 33	\gls@protected@pagefmts	118, 186
\gls@debug@val	5, 33	\gls@Romanpage	186, 189
\gls@defdocnewglossaryentry	76, 96	\gls@romanpage	186, 189
\gls@defglossaryentry	74–76, 84	\gls@save@numberlist	9
\gls@disablepagerefexpansion ..	185, 189	\gls@seen@index@nr	7
\gls@do@addxdyattribute	49	\gls@seen@index@val	7
\gls@docclearpage	46	\gls@set@xr@key	69
\gls@dosubst	118	\gls@suffixF	42, 167, 169, 329, 331
\gls@dotoc@title	195, 196, 208	\gls@suffixFF	42, 167–170, 329, 331
\gls@end@sanitizesort	22, 23	\gls@text	110
\gls@endcheck	73, 74	\gls@the	189
\gls@glossary	32, 33, 190–192	\gls@thissty	28
\gls@gobbleopt	65	\gls@tmp	183, 251
\gls@grplabel	272	\gls@tmplen	125, 321–323, 339, 340
\gls@hypergroup@prerun	273	\gls@tr@set@acronym@toctitle	17
\gls@ifnotmeasuring	93, 94	\gls@tr@set@main@toctitle	16
\gls@inlinepostchild	275, 276, 332	\gls@tr@set@numbers@toctitle	34
\gls@inlinesep	275, 332	\gls@tr@set@symbols@toctitle	33
\gls@inlinesubsep	275, 276, 332	\gls@translate@nr	27
\gls@islistofacronyms	18	\gls@translate@val	27
\gls@istfilebase	40, 41, 174, 175	\gls@wrglossary	185
\gls@label	222	\gls@xdystring	118
\gls@level	87, 88, 205	\gls@xindy@glsnumbersfalse	30
		\gls@xindy@glsnumberstrue	29
		\gls@xr@key	6, 7, 69
		\gls@sacccsupp	351
		\gls@acronymtrue	17
		\gls@acrplural@suffix	38, 223, 232, 236–238, 242

<code>\glsacrshortcutsfalse</code>	35	<code>\glsdoifexistsordo</code>	114, 152
<code>\glsacrshortcutstrue</code>	35	<code>\glsdoifexistsorwarn</code>	207, 213, 214
<code>\glsacspace</code>	230, 233	<code>\glsdoifnoexists</code>	74, 84
<code>\glsadd</code>	32, 162	<code>\glsdonohyperlink</code>	116, 125, 126
<code>\glsadd options</code>		<code>\glsdosanitizesort</code>	13
<code>counter</code>	161	<code>\glsentryaccess</code>	351
<code>format</code>	161, 219	<code>\glsentrycounter</code>	217, 220
<code>\glsaddall options</code>		<code>\glsentrycounterfalse</code>	12
<code>types</code>	161	<code>\glsentrycounterlabel</code>	212
<code>\GlsAddXdyAttribute</code>	48–50, 325, 326	<code>\GlsEntryCounterLabelPrefix</code>	211
<code>\GlsAddXdyCounters</code>	48, 49, 65	<code>\glsentrycountertrue</code>	12
<code>\glsautomakefalse</code>	30, 31, 176	<code>\glsentrycurrcount</code>	96, 98
<code>\glsautomaketrue</code>	30	<code>\Glsentrydesc</code>	137, 214, 365
<code>\glsautoprefix</code>	8, 208	<code>\glsentrydesc</code> 104, 105, 137, 213, 355–357, 365	
<code>\glscapspace</code>	102, 103, 106–109, 127–131, 145–151, 234, 235, 247, 252, 353, 355, 357, 358, 360, 361, 363–365, 370	<code>\glsentrydescaccess</code>	352
<code>\glsclearpage</code>	45	<code>\Glsentrydescplural</code>	138
<code>\glsclosebrace</code>	52, 53, 167, 168, 329, 330	<code>\glsentrydescplural</code>	102, 103, 137, 138, 353–355
<code>\glscompositor</code>	41, 42, 52, 169, 328, 331	<code>\glsentrydescpluralaccess</code>	352
<code>\glscounter</code>	19, 36, 48, 65, 88, 116, 325	<code>\Glsentryfirst</code> ..	100, 104, 107, 134, 356, 359
<code>\glscurrententrylabel</code>	194, 196	<code>\glsentryfirst</code>	99, 104, 105, 107, 133, 134, 355, 356, 359
<code>\glscurrentfieldvalue</code>	61, 62	<code>\glsentryfirstaccess</code>	352
<code>\glscustomtext</code>		<code>\Glsentryfirstplural</code>	
....	102, 105, 107, 108, 110, 127–131, 145–152, 234, 235, 243, 247, 249, 250, 252, 353, 357, 359, 360, 362–365, 370, 371	<code>\glsentryfirstplural</code>	101, 103, 106, 135, 354, 358
<code>\GlsDeclareNoHyperList</code>	19	<code>\glsentryfirstplural</code>	100, 102, 103, 106, 135, 353, 355, 357, 358
<code>\glsdefaulttype</code>	16, 43, 55, 56, 63, 64, 86, 101, 111, 183, 194, 195	<code>\glsentryfirstpluralaccess</code>	352
<code>\glsdefmain</code>	16, 65	<code>\glsentryfmt</code>	64, 66
<code>\glsdescriptionaccessdisplay</code>		<code>\Glsentryfull</code>	228, 236, 238, 372, 374
.....	355–357, 365, 366	<code>\glsentryfull</code>	228, 236, 238, 371, 374
<code>\glsdescriptionpluralaccessdisplay</code>		<code>\Glsentryfullpl</code>	228, 236, 238, 372, 374
.....	353–355	<code>\glsentryfullpl</code>	228, 236, 238, 372, 374
<code>\glsdescwidth</code>	281– 283, 285, 286, 289–296, 303–306, 308–315	<code>\glsentryitem</code>	275, 277–282, 284, 292, 293, 295, 303, 305, 306, 310, 311, 313, 316, 318, 319, 322, 332–339, 341–344
<code>\glsdetoklabel</code>	57–61, 70, 76, 81–85, 91, 94–98, 115, 152, 153, 160, 161, 179– 181, 187, 190, 196, 199–201, 205, 207, 211, 213, 258–263, 321, 345, 346, 381, 382	<code>\Glsentrylong</code>	100, 149, 153, 159, 230, 235, 236, 362, 364, 367, 370–372
<code>\glsdisplay</code>	102, 111	<code>\glsentrylong</code>	99, 110, 148, 150, 153, 159, 229–239, 249, 362, 364–375
<code>\glsdisplayfirst</code>	102, 111	<code>\glsentrylongaccess</code>	352
<code>\glsdisplaynumberlist</code>	181	<code>\Glsentrylongpl</code>	101, 151, 159, 230, 235, 236, 362, 367, 370–372
<code>\glsdohyperlink</code>	125, 126	<code>\glsentrylongpl</code>	100, 110, 150, 152, 159, 230, 231, 235–238, 249, 257, 362, 367, 368, 370–374
<code>\glsdohypertarget</code>	126	<code>\glsentrylongpluralaccess</code>	353
<code>\glsdoifexists</code>		<code>\Glsentryname</code>	136, 213, 365
..	59, 61, 81–83, 93, 94, 127–132, 145– 151, 159, 161, 180, 181, 268–271, 362–366	<code>\glsentryname</code>	136, 321, 365

<code>\glsentrynumberlist</code>	159, 180	<code>\glsentryuservi</code>	144
<code>\Glsentryplural</code>	103, 106, 134, 354, 358	<code>\glsesclocationsfalse</code>	179
<code>\glsentryplural</code>	102, 103, 106, 134, 135, 353, 354, 357, 358	<code>\glsesclocationstrue</code>	9
<code>\glsentrypluralaccess</code>	352	<code>\glsfieldfetch</code>	156
<code>\Glsentryprefix</code>	269	<code>\glsfirstaccessdisplay</code>	355, 356, 359
<code>\glsentryprefix</code>	268, 270	<code>\glsfirstpluralaccessdisplay</code>	353, 354, 357, 358
<code>\Glsentryprefixfirst</code>	269	<code>\glsfirstpluralaccessdisplay</code>	358
<code>\glsentryprefixfirst</code>	268, 270	<code>\glsgenacfmt</code>	229–231, 237, 366, 367, 372
<code>\Glsentryprefixfirstplural</code>	270	<code>\glsgenentryfmt</code>	229–231, 235, 237, 241, 243, 245, 247, 249, 252, 254, 257, 366, 367, 371, 372
<code>\glsentryprefixfirstplural</code>	268, 271	<code>\glsgetgroupitle</code>	274, 278, 279, 297–302, 317–320, 324
<code>\Glsentryprefixplural</code>	270	<code>\gls glossarymark</code>	44
<code>\glsentryprefixplural</code>	268, 271	<code>\gls groupheading</code> 168, 206, 275, 277–279, 281, 282, 284, 292, 293, 295, 297–304, 306, 310, 311, 313, 316–321, 323, 324, 330	
<code>\glsentryprevcount</code>	96–98	<code>\gls groupskip</code> 167, 168, 206, 276, 278, 281, 283, 284, 287, 288, 292–295, 303, 305, 307, 310, 312, 314, 317, 318, 320, 323, 329	
<code>\Glsentryshort</code>	108, 145, 153, 231, 237, 238, 361–363, 367, 373, 374	<code>\glshyperfirstfalse</code>	237, 372
<code>\glsentryshort</code> ..	108, 110, 145, 146, 153, 159, 228–239, 360–363, 366–369, 371–375	<code>\glshyperfirsttrue</code>	28
<code>\glsentryshortaccess</code>	352	<code>\glshyperlink</code>	193
<code>\Glsentryshortpl</code>	108, 147, 231, 237, 238, 360, 367, 373, 374	<code>\glshypernavsep</code>	274
<code>\glsentryshortpl</code>	108, 110, 147, 148, 159, 230, 231, 236–238, 257, 360, 362, 367, 371–374	<code>\glshypernumber</code>	43, 221
<code>\glsentryshortpluralaccess</code>	352	<code>\glsifhyperon</code>	113
<code>\Glsentrysymbol</code>	138, 214, 366	<code>\glsIfListOfAcronyms</code>	18, 19
<code>\glsentrysymbol</code>	104, 105, 138, 139, 214, 243, 247, 252, 355–357, 366	<code>\glsifplural</code>	102, 105, 108, 109, 127–131, 145–151, 234, 243, 247, 249, 252, 353, 357, 360, 361, 363–365, 370
<code>\glsentrysymbolaccess</code>	352	<code>\glsifusetranslator</code>	26, 27, 39, 383
<code>\Glsentrysymbolplural</code>	139	<code>\glsindexonlyfirstfalse</code>	28
<code>\glsentrysymbolplural</code>	102, 103, 139, 140, 243, 247, 252, 353–355	<code>\glsinlinedescformat</code>	275, 332
<code>\glsentrysymbolpluralaccess</code>	352	<code>\glsinlinedopostchild</code>	275, 332
<code>\Glsentrytext</code>	104, 107, 133, 355, 359	<code>\glsinlineemptydescformat</code>	275, 332
<code>\glsentrytext</code>	104, 105, 107, 132, 133, 160, 193, 355, 356, 358, 359	<code>\glsinlineformat</code>	275, 332
<code>\glsentrytextaccess</code>	351	<code>\glsinlineparentchildseparator</code>	275, 332
<code>\glsentrytype</code>	86	<code>\glsinlinepostchild</code>	275, 332
<code>\Glsentryuseri</code>	140	<code>\glsinlineseparator</code>	275, 332
<code>\glsentryuseri</code>	140	<code>\glsinlinesubdescformat</code>	276, 332
<code>\Glsentryuserii</code>	141	<code>\glsinlinesubnameformat</code>	275, 332
<code>\glsentryuserii</code>	141	<code>\glsinlinesubseparator</code>	276, 332
<code>\Glsentryuseriii</code>	142	<code>\glsinsert</code> 102–109, 127–131, 145–151, 235, 243, 247, 249, 250, 252, 353–365, 370, 371	
<code>\glsentryuseriii</code>	141, 142	<code>\glskeylisttok</code>	227, 241, 242, 244– 246, 248, 250, 251, 253–255, 257, 375–380
<code>\Glsentryuseriv</code>	142		
<code>\glsentryuseriv</code>	142, 143		
<code>\Glsentryuserv</code>	143		
<code>\glsentryuserv</code>	143, 144		
<code>\Glsentryuservi</code>	144		

<code>\glslabel</code>	85, 102–109, 114–116, 145–151, 229–231, 234, 235, 237, 243, 247, 249, 252, 353–362, 366, 367, 370–372
<code>\glslabeltok</code>	227, 241–248, 250–252, 254, 255, 257, 376–379
<code>\glslink</code>	227, 228, 235–238, 243, 371, 373
<code>\glslink options</code>	
counter	112, 126, 264
format	112, 126, 219
hyper	112, 114, 115, 126
local	112
<code>\glslinkcheckfirsthyperhook</code>	114
<code>\glslinkpostsetkeys</code>	115
<code>\glslinkvar</code>	113
<code>\glslistdottedwidth</code>	279, 280, 333
<code>\glslistgroupheaderfmt</code>	278, 279
<code>\glslistnavigationitem</code>	278, 279
<code>\glslocalreset</code>	95
<code>\glslocalunset</code>	96, 127–132
<code>\glslongaccessdisplay</code>	362, 364–376
<code>\glslongkey</code>	380
<code>\glslongpluralaccessdisplay</code>	
.....	362, 367, 368, 370–374, 376
<code>\glslongpluralkey</code>	380
<code>\glslongtok</code>	227–231, 235, 237, 241, 242, 244–246, 248, 250, 251, 253– 255, 257, 258, 366, 367, 371, 372, 375–380
<code>\glsLTpenaltycheck</code>	290, 291
<code>\glsmcols</code>	297–302
<code>\glsnameaccessdisplay</code>	365, 366
<code>\glsnamefont</code>	213, 214, 345, 346, 365
<code>\glsnavhyperlink</code>	274
<code>\glsnavhyperlinkname</code>	272, 273
<code>\glsnavhypertarget</code>	
.....	278, 279, 298–302, 318–320, 324
<code>\glsnavigation</code>	
....	278, 279, 297–302, 317, 319, 320, 324
<code>\glsnextpages</code>	9, 70, 196
<code>\glsnogroupskipfalse</code>	11
<code>\glsnoidxdisplayloc</code>	181, 183
<code>\glsnoidxdisplayloclisthandler</code>	181
<code>\glsnoidxloclist</code>	180, 205, 206
<code>\glsnoidxloclisthandler</code>	206
<code>\glsnoidxnumberlistloophandler</code>	181
<code>\glsnoidxstripaccents</code>	23
<code>\glsnomakeindexwarning</code>	170
<code>\glsnonextpages</code>	69, 196
<code>\glsnopostdotfalse</code>	11
<code>\glsnoxindywarning</code> 42, 48–50, 53–55, 163, 164	
<code>\glsnumberformat</code>	160
<code>\glsnumberlistloop</code>	181
<code>\glsnumbersgroupname</code>	34, 40, 217
<code>\glsnumlistlastsep</code>	160, 181
<code>\glsnumlistparser</code>	160, 178
<code>\glsnumlistsep</code>	160, 181
<code>\glsopenbrace</code>	52, 53, 167, 168, 329, 330
<code>\glsorder</code>	29, 174–177, 202
<code>\glsorg@endtheglossary</code>	5
<code>\glsorg@PrintChanges</code>	5
<code>\glsorg@theglossary</code>	5
<code>\glspagelistwidth</code> 282, 283, 285, 286, 289, 290, 293–296, 304–306, 308, 309, 311–315	
<code>\glspatchLtoutput</code>	287–289
<code>\glspenaltygroupskip</code>	287, 288
<code>\glspercentchar</code>	76, 167, 168
<code>\Glspl</code>	100, 241
<code>\glspl</code>	100, 241
<code>\glspluralaccessdisplay</code> 353, 354, 357, 358	
<code>\glspluralsuffix</code>	
.....	38, 88, 230, 231, 367, 368, 372–374
<code>\glspostdescription</code>	
. 40, 276–279, 281, 292, 303, 310, 316– 318, 320, 322, 323, 332–335, 337–341, 343	
<code>\glspostinline</code>	275
<code>\glspostlinkhook</code>	
.....	114, 127–132, 145–152, 363–365
<code>\glsprestandardsort</code>	13
<code>\glsreset</code>	95
<code>\glsresetentrycounter</code>	211
<code>\glsresetentrylist</code>	167, 204, 210, 329
<code>\glsresetsubentrycounter</code> ...	212, 275, 332
<code>\glsanitizesortfalse</code>	25
<code>\glsanitizesorttrue</code>	25
<code>\glsavenumberlistfalse</code>	9
<code>\glsavewritesfalse</code>	31
<code>\glsseeformat</code>	166, 179, 181, 329
<code>\glsseeitem</code>	193
<code>\glsseeitemformat</code>	193
<code>\glsseelastsep</code>	193
<code>\glsseelist</code>	192
<code>\glsseesep</code>	193
<code>\glssetexpandfield</code>	21, 24–26
<code>\glssetnoexpandfield</code>	21, 22, 24, 25
<code>\GlsSetQuote</code>	85, 168
<code>\glssettoctitle</code>	39, 195
<code>\glsshortaccessdisplay</code>	
.....	360–363, 366–369, 371–376
<code>\glsshortkey</code>	380

<code>\glsshortpluralaccessdisplay</code>	360, 362, 367, 371–374, 376
<code>\glsshortpluralkey</code>	380
<code>\glsshorttok</code>	227, 241–246, 248, 250–255, 257, 376–380
<code>\glsshowtarget</code>	6
<code>\glssortnumberfmt</code>	14, 15
<code>\glsspace</code>	224
<code>\glssstepentry</code>	212
<code>\glssstepsubentry</code>	212
<code>\glssubentrycounterfalse</code>	12
<code>\glssubentrycounterlabel</code>	212
<code>\glssubentryitem</code> 276, 277, 279–282, 284, 292, 293, 295, 303, 305, 306, 310, 312, 313, 317, 318, 320, 322, 332–339, 341–344	
<code>\glssymbolaccessdisplay</code>	355–357, 366
<code>\glssymbolpluralaccessdisplay</code> .	353–355
<code>\glssymbolsgroupname</code>	33, 40, 217
<code>\glstarget</code>	215, 276–282, 284, 292, 293, 295, 303, 305, 306, 310–313, 316–320, 322, 323, 332–344
<code>\glstextaccessdisplay</code> ..	355, 356, 358, 359
<code>\glstextformat</code>	114, 116
<code>\glstextup</code>	38, 374
<code>\glstildechar</code>	49, 167, 168
<code>\glstranslatefalse</code>	27
<code>\glstranslatetrue</code>	27, 28
<code>\glstreechildpredesc</code>	317, 318
<code>\glstreegroupheaderfmt</code>	297–302, 317, 319, 320, 324
<code>\glstreeindent</code> ..	318, 320, 322, 323, 338–340
<code>\glstreeitem</code>	297, 298, 316
<code>\glstreenamebox</code>	322, 323
<code>\glstreenamefmt</code>	315–319, 321–323
<code>\glstreenavigationfmt</code>	297–302, 317, 319, 320, 324
<code>\glstreepredesc</code>	316, 318, 320
<code>\glstreesubitem</code>	297, 316
<code>\glstreesubsubitem</code>	297, 316
<code>\glstype</code> .	114, 115, 127–132, 145–152, 363–365
<code>\glsucmarkfalse</code>	11
<code>\glsucmarktrue</code>	11
<code>\glsunset</code>	93, 95, 97, 98, 127–132
<code>\glsupacrpluralsuffix</code>	231, 232, 238, 245, 249, 251, 254
<code>\GlsUseAcrEntryDispStyle</code> ...	228, 231– 234, 236, 238, 239, 368, 369, 372, 374, 375
<code>\GlsUseAcrStyleDefs</code>	228, 231– 234, 236, 238, 239, 368, 369, 372, 374, 375
<code>\glswrallowprimitivemodstrue</code>	188
<code>\glswrite</code>	164–170, 177, 183, 327–331
<code>\glswritedefhook</code>	76
<code>\glswriteentry</code>	185
<code>\glswritefiles</code>	31, 183
<code>\glsxindyfalse</code>	29
<code>\glsxindytrue</code>	30
H	
<code>\H</code>	23
<code>\hangindent</code>	301, 302, 315, 318–320, 322–324, 337–340
<code>\hbox</code>	93, 279, 280, 333
<code>\hfill</code>	279, 280, 333
<code>\hline</code> ...	281–283, 285, 292–294, 296, 303–315
<code>\hsize</code>	280, 291, 302, 309
<code>\hspace</code>	316
<code>\hss</code>	279, 280, 333
<code>\ht</code>	290
<code>\hyperdef</code>	36
<code>\hyperlink</code>	112, 125, 220
<code>hyperref package</code>	191, 194, 219, 264
<code>\hypertarget</code>	125
I	
<code>\IeC</code>	23
<code>\if</code>	118, 190, 326
<code>\if@endfor</code>	273
<code>\if@gl@debug</code>	5, 20, 184
<code>\if@gl@docloaded</code>	4, 16, 32
<code>\if@gl@isacronymlist</code>	114
<code>\if@openright</code>	46
<code>\ifbool</code>	17, 28, 31, 58, 102–104
<code>\ifboolexpr</code>	39, 63, 216
<code>\ifcase</code>	5, 7, 8, 27, 33, 69, 208, 317, 337
<code>\ifcsdef</code>	26, 40, 46, 72, 73, 79–83, 111, 185, 198, 199, 202, 203, 218, 229
<code>\ifcsempy</code>	59, 60, 267
<code>\ifcsequal</code>	60
<code>\ifcsstrequal</code>	83
<code>\ifcsstring</code>	83
<code>\ifcsundef</code> .	7, 14, 30, 36, 39, 42, 44, 46, 57, 64, 66, 68, 86, 88, 89, 96, 97, 112, 116, 117, 125, 174, 175, 183, 194, 197, 199, 207, 208, 212, 216–219, 222, 228, 274, 331
<code>\ifdef</code>	61, 70, 75, 76, 93, 112, 152, 156, 180, 181, 223, 267, 315, 316
<code>\ifdefempty</code> ..	19, 45, 56, 60, 61, 66, 102, 105, 108, 176, 178, 204, 205, 227, 228, 234, 243, 247, 249, 251, 252, 353, 357, 360, 370

<code>\ifdefequal</code>	59–62, 72–74, 77, 87, 91, 205	<code>\ifglstranslate</code>	38
<code>\ifdefstrequal</code>	83	<code>\ifglsucmark</code>	44, 45
<code>\ifdefstring</code>		<code>\ifglused</code>	98, 102–108, 114, 162, 185, 243, 247, 249, 252, 268–271, 353–360
<code>\ifdefvoid</code>	22, 23, 90, 205, 206	<code>\ifglswrallowprimitivemods</code>	189
<code>\ifdim</code>	231, 290, 321	<code>\ifglsxindy</code>	40– 42, 47–50, 52–55, 65, 91, 92, 119, 163, 164, 170, 174, 175, 190, 192, 197, 325–327
<code>\iffalse</code>	89, 95	<code>\ifignoredglossary</code>	86, 90, 185
<code>\IfFileExists</code>	10, 26, 27, 175, 176, 197	<code>\ifin@</code>	34
<code>\ifglossaryexists</code> ..	43, 55, 59, 173–176, 195	<code>\ifinlistcs</code>	203, 207
<code>\ifgl@sanitize@description</code>	24	<code>\ifinner</code>	6
<code>\ifgl@sanitize@name</code>	24	<code>\ifKV@glslink@hyper</code>	115, 116
<code>\ifgl@sanitize@symbol</code>	24	<code>\ifKV@glslink@local</code>	127–132
<code>\ifgl@xindy@glslnumbers</code>	55	<code>\ifmeasuring@</code>	93
<code>\ifgl@acrdescription</code>	256	<code>\ifmmode</code>	6
<code>\ifgl@acrdua</code>	245, 251, 254, 256	<code>\ifnum</code>	14, 30, 97, 98, 176, 205, 290, 318, 320, 322, 338, 339
<code>\ifgl@acrfootnote</code>	114, 256	<code>\ifstreempty</code>	332
<code>\ifgl@acrshorthand</code>	17	<code>\ifstrequal</code>	216
<code>\ifgl@acrshortcuts</code>	35, 241	<code>\ifthenelse</code> ..	25, 36, 46, 116, 177, 217, 256, 273
<code>\ifgl@acrsmalls</code> ..	245, 246, 249, 251, 254	<code>\IfTrackedLanguage</code>	170
<code>\ifgl@acrsmalls</code> ..	245, 246, 249, 251	<code>\IfTrackedLanguageFileExists</code> ..	39, 383, 384
<code>\ifgl@autotake</code>	30, 178	<code>\iftrue</code>	90, 94
<code>\ifgl@descsuppressed</code>	275	<code>\ifundef</code>	11, 12, 64, 75, 86, 164, 168, 177
<code>\ifgl@entrycounter</code>	11, 12, 35, 211, 212	<code>\ifvmode</code>	161
<code>\ifgl@entryexists</code>	58, 75, 76, 84, 87	<code>\ifvoid</code>	290
<code>\ifgl@esclocations</code>	187	<code>\ifx</code>	11, 13, 15, 18, 34, 36, 47, 49, 51, 52, 55, 56, 86–89, 92, 116, 117, 119–124, 153, 164, 167, 169–172, 183, 187, 189–193, 195, 196, 217–220, 242, 244, 246, 248, 250, 251, 253, 255, 257, 258, 327–329, 331, 332, 337–340, 345, 351
<code>\ifgl@haschildren</code>	275, 332	<code>\immediate</code> ..	75, 76, 98, 174–176, 183, 197, 198
<code>\ifgl@hasdesc</code>	275	<code>\in@</code>	34
<code>\ifgl@haslong</code>	99–101, 153, 229–231, 234, 237, 249, 366, 367, 370, 372	<code>\indexname</code>	34
<code>\ifgl@hasparent</code>	199, 205, 321	<code>\indexspace</code> ..	278, 297–302, 317–320, 323, 324
<code>\ifgl@hasprefix</code>	269	<code>\input</code>	38, 101
<code>\ifgl@hasprefixfirst</code>	269	<code>\inputencodingname</code>	30
<code>\ifgl@hasprefixfirstplural</code>	270	<code>\InputIfFileExists</code>	75
<code>\ifgl@hasprefixplural</code>	270	<code>\istfilename</code> ..	40, 164, 168, 175–177, 327, 330
<code>\ifgl@hassymbol</code>		<code>\item</code> ..	277–280, 297, 298, 316, 317, 332, 333, 337
....	243, 247, 252, 316–318, 320, 322, 323		
<code>\ifglshyperfirst</code>	114		
<code>\ifgl@indexonlyfirst</code>	185		
<code>\ifgl@nogroupskip</code>	278, 281, 283, 284, 287, 288, 292, 293, 295, 303, 305, 307, 310, 312, 314, 317, 318, 320, 323		
<code>\ifgl@nonumberlist</code>	209		
<code>\ifgl@nopostdot</code>	11		
<code>\ifgl@numberline</code>	47		
<code>\ifgl@sanitizesort</code>	22, 25		
<code>\ifgl@savenumberlist</code>	72, 178, 193		
<code>\ifgl@savewrites</code>	31, 173, 185		
<code>\ifgl@subentrycounter</code>	12, 35, 210–212		
<code>\ifglstoc</code>	46		

J

`\jobname` 41, 75, 164, 168, 174–176, 197, 327, 330

K

`\key@ifundefined` 78, 79 || `\KV@glslink@hyperfalse` .. | 112, 114, 115, 126 |
| `\KV@glslink@hypertrue` | 112, 126 |

L	
<code>\L</code>	24
<code>\l</code>	24
<code>\label</code>	8, 208, 211
<code>\language</code>	29
<code>\leaders</code>	279, 280, 333
<code>\leavevmode</code>	84, 115
<code>\let</code>	5, 10, 13–17, 23, 24, 26, 27, 31, 33–36, 39, 40, 49, 50, 61–63, 72, 74, 75, 84–87, 89, 90, 93–96, 98, 101, 113–116, 118, 125–132, 145– 151, 153, 160, 168, 170, 173–182, 185, 186, 189, 192, 193, 195–197, 206, 208, 210, 214, 227, 239–242, 244, 246–250, 252, 253, 255, 265, 273, 274, 290, 297, 298, 316, 331, 348, 363–365, 376–380, 383
<code>\letcs</code>	59– 61, 76, 78, 79, 82, 88, 152, 153, 174, 175, 180, 181, 199–201, 205, 213, 216, 217, 321
link text	101
<code>\listcsadd</code>	203
<code>\listcsgadd</code>	207
<code>\listcsxadd</code>	198, 199
<code>\listead</code>	203
<code>\loadglsentries</code>	101
<code>\long</code>	84, 221
<code>\longnewglossaryentry</code>	84
longtable package	280, 287, 291
<code>\LT@end@pen</code>	290
<code>\LT@err</code>	290
<code>\LT@foot</code>	290, 291
<code>\LT@head</code>	290, 291
<code>\LT@lastfoot</code>	290
<code>\LT@output</code>	290
M	
<code>\makeatletter</code>	75, 197
<code>\makebox</code>	279, 280, 321–323, 333, 339, 340
makeglossaries	29, 41, 54, 55, 63, 170, 177, 197
<code>\makeglossaries</code>	6, 7, 30–33, 37, 69, 174, 180, 182, 198
<code>\makeglossary</code>	31, 33
makeindex	385
makeindex	9, 13, 29, 30, 37, 41, 43, 47, 63, 65, 67, 92, 117, 120, 162, 166, 168, 170, 173, 184, 188–191, 215, 216, 326, 327
delim_n	43
delim_r	43
page_compositor	41
special characters	119, 162
<code>\makenoidxglossaries</code>	6, 7, 69, 178, 182
<code>\MakeTextUppercase</code>	4
<code>\MakeUppercase</code>	354, 356, 363, 365
<code>\marginpar</code>	6
<code>\markboth</code>	44
<code>\mbox</code>	161, 278, 301, 302, 321, 333
memoir class	184
<code>\memUHead</code>	44
<code>\MessageBreak</code> ..	20, 32, 63, 195, 345, 383, 384
mfirstuc package	1
<code>\mfirstucMakeUppercase</code>	4, 44, 45, 80, 103, 105–109, 133–144, 146, 148, 150, 152, 227, 228, 235–238, 247, 252, 270, 271, 358–362, 370, 371, 373, 374
<code>\midrule</code>	287, 288
<code>\month</code>	164, 168, 327, 330
multicol package	296
N	
<code>\n</code>	169, 330
<code>\NeedsTeXFormat</code> ...	4, 265, 325, 331, 345, 383
<code>\new@glossaryentry</code>	75, 180
<code>\new@ifnextchar</code>	63, 79, 80, 99, 100, 126–130, 132–151, 223–226, 267–271
<code>\newacronym</code>	222, 227, 242, 244, 246, 248, 250, 253, 255, 257
<code>\newacronymhook</code>	227, 242, 244, 246, 248, 251, 254, 255, 258, 375
<code>\newacronymstyle</code>	229–234, 236–239
<code>\newcommand</code>	6–23, 25, 26, 28–38, 40–50, 52–66, 68–84, 90, 91, 93– 96, 98–102, 105, 108, 110, 111, 113–116, 118, 119, 125–164, 170, 173–177, 179, 182–187, 189–195, 197–203, 205–207, 209–219, 221–229, 231, 239, 241–250, 252–264, 266–274, 276, 277, 290, 297, 315, 316, 321, 331, 349–351, 366, 380–382
<code>\newcount</code>	14, 15, 72
<code>\newcounter</code>	11, 12
<code>\newenvironment</code>	212
<code>\newglossary</code>	16, 17, 33, 34, 65, 177
<code>\newglossaryentry</code>	6, 34, 71, 72, 75, 96, 227, 241, 243, 245, 247, 250, 252, 254, 257, 376–379
<code>\newglossaryentry options</code>	
access	348, 349
counter	68
description	28, 67, 72, 74, 85, 136, 154, 223, 250, 347

`\org@glossaryentrynumbers` 196, 210
`\org@glossarytitle` 195, 196
`\org@glspostdescription` 40
`\org@ifKV@glslink@hyper` 115, 116
`\outputpenalty` 290

P

`\p@` 277, 297, 315, 316
`\p@glshyp@opt` 113
 package options:
 `acronym` 16, 17, 37, 194, 222
 `true` 17
 `counter` 19
 `debug`
 `showtargets` 6
 `description` 248, 249
 `dua` 247–249
 `entrycounter` 11, 209, 210
 `true` 12
 `esclocations` 408
 `false` 9
 `footnote` 127–132, 244, 247, 248, 250
 `hyperfirst`
 `false` 127–132
 `index` 34
 `indexonlyfirst` 392
 `kernelglossredefs`
 `nowarn` 32
 `makeindex` 166, 264
 `nogroupskip` 281, 283, 284, 287, 288,
 292, 293, 295, 303, 305, 307, 310, 312, 314
 `nolist` 258
 `nolong` 258, 280
 `nomain` 16
 `nonumberlist` 9
 `nosuper` 258
 `notree` 258
 `nowarn` 5
 `numberline` 7
 `record` 267
 `sanitize` 24, 67, 153, 154
 `sanitizesort` 21
 `savewrites` 31, 389
 `false` 173
 `true` 177, 183
 `section` 7, 45
 `sort`
 `def` 13, 14
 `none` 13

`standard` 13
 `use` 13, 14, 408
 `style` 8, 258
 `subentrycounter` 12, 209, 210
 `toc` 7
 `true` 7
 `translate` 27
 `false` 26
 `translator` 26
 `xindy` 29, 30, 166, 264
`\PackageError` 6,
 7, 16, 30, 37, 48, 55, 58, 59, 63, 68, 71,
 72, 78–83, 86, 87, 96, 112, 152, 173, 174,
 177, 180, 182, 201–203, 208, 209, 217–
 219, 228, 229, 245, 246, 251, 254, 331, 353
`\PackageInfo` 5, 6, 174, 184
`\PackageWarning` 5, 6, 20
`\PackageWarningNoLine` ... 5, 6, 20, 383, 384
`\pagegoal` 290
`\pagelistname` 40, 283,
 285, 287, 288, 294, 296, 305–309, 312–315
`\par` 40, 215, 277–279, 297,
 299–302, 315, 316, 318–324, 333, 338–340
`\parindent`
 297–302, 316, 318–320, 322–324, 338–340
`\parskip` 297–300, 316, 318, 319
`\PassOptionsToPackage` 265, 345
`\penalty` 290
`\phantomsection` 45
`polyglossia package` 26, 38
`\printglossaries` 178
`\printglossary` .. 17, 20, 33, 34, 178, 194, 209
`\printglossary options`
 `entrycounter` 209
 `nogroupskip` 208
 `nonumberlist` 209
 `nopostdot` 209
 `numberedsection` 208
 `style` 208
 `subentrycounter` 209
 `title` 208
 `toctitle` 208
 `type` 16, 193, 207
`\printindex` 34
`\printnoidxglossaries` 180
`\printnoidxglossary`
 179, 180, 182, 195, 201, 202, 209
`\printnoidxglossary options`
 `sort` 209

<code>\printhnumbers</code>	34	<code>\RequirePackage</code>	
<code>\printsymbols</code>	33	.. 4, 9, 10, 26, 27, 35, 38, 258, 264, 265,	
<code>\ProcessOptions</code>	265, 345	280, 286, 287, 291, 297, 302, 309, 346, 383	
<code>\ProcessOptionsX</code>	35	<code>\restorecounters@</code>	117
<code>\protect</code>	47, 110, 229–231,	<code>\romannumeral</code>	186, 189, 321–323, 339
237, 243, 247, 249, 362, 366, 367, 372, 373			
<code>\protected@csedef</code>	81		
<code>\protected@csxdef</code>	81		
<code>\protected@edef</code>	8, 49, 51, 54, 56,		
86, 90, 92, 102–104, 110, 117, 160, 185,			
187, 190, 208, 213, 214, 217, 218, 227,			
251, 257, 266, 272, 326, 327, 345, 346, 351			
<code>\protected@write</code>	62, 64,		
165, 166, 177, 179, 182, 185, 194, 273, 327			
<code>\protected@xdef</code> 13–15, 18, 23, 73, 92, 190, 349			
<code>\providecommand</code>	17,		
37, 38, 45, 62, 98, 126, 166, 177, 180,			
183, 197, 198, 213, 214, 267, 277, 297, 315			
<code>\ProvidesFile</code>	38		
<code>\ProvidesPackage</code>			
..... 4, 265, 272, 274, 277, 280, 286,			
291, 296, 302, 309, 315, 325, 331, 345, 383			
	R		
<code>\r</code>	23		
<code>\raggedright</code>	289–296, 310–315		
<code>\raisebox</code>	125		
<code>\ref</code>	211		
<code>\refstepcounter</code>	211		
<code>\relax</code>	5, 8, 10, 14–		
17, 27, 30, 33–35, 50, 63, 68, 69, 73, 75,			
87, 89, 93, 97, 98, 113, 114, 116, 118–			
124, 153, 167, 168, 170, 172, 173, 176–			
181, 186, 190, 192, 193, 195, 197, 204,			
205, 208, 217, 218, 258, 273, 277, 290,			
297, 301, 302, 315, 317–324, 326, 329–			
331, 337–340, 348, 363, 364, 376–378, 380			
<code>\renewacronymstyle</code> .	366–370, 372, 374, 375		
<code>\renewcommand</code>	4–10, 12, 13,		
16, 17, 19–21, 25, 27–31, 33, 35, 39–42,			
55, 66, 69, 70, 84, 96–98, 160, 161, 163,			
164, 170, 171, 176–181, 196, 198, 208,			
227–239, 242, 244–246, 248–251, 253–			
255, 257, 275–285, 287, 288, 290–307,			
310–314, 316–326, 331–339, 341–344,			
348, 353, 357, 360, 362, 365–369, 371–379			
<code>\renewenvironment</code>	212,		
274, 277, 281–286, 289–316, 318, 319, 321			
<code>\RequireGlossariesLang</code>	39, 383, 384		
		S	
		<code>\s@glshyp@opt</code>	113
		<code>\s@GlsSetXdyFirstLetterAfterDigits</code>	163
		<code>\s@GlsSetXdyNumberGroupOrder</code> ..	163, 164
		<code>\s@newglossary</code>	63
		<code>\savecounters@</code>	116
		<code>\seenname</code>	192
		<code>\SetAcronymStyle</code>	28, 29
		<code>\setbool</code>	25
		<code>\setbox</code>	290, 291
		<code>\setcounter</code>	210, 211
		<code>\SetCustomDisplayStyle</code>	257, 258
		<code>\SetDefaultAcronymDisplayStyle</code> ...	242
		<code>\SetDefaultAcronymStyle</code>	256
		<code>\SetDescriptionAcronymDisplayStyle</code>	
	 248, 249	
		<code>\SetDescriptionAcronymStyle</code>	256
		<code>\SetDescriptionDUAAcronymDisplayStyle</code>	
	 246	
		<code>\SetDescriptionDUAAcronymStyle</code> ...	256
		<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>	
	 244, 245	
		<code>\SetDescriptionFootnoteAcronymStyle</code>	256
		<code>\SetDUADisplayStyle</code>	255, 256
		<code>\SetDUASStyle</code>	256
		<code>\setentrycounter</code>	49, 166, 207, 325
		<code>\SetFootnoteAcronymDisplayStyle</code> ...	251
		<code>\SetFootnoteAcronymStyle</code>	256
		<code>\SetGenericNewAcronym</code>	228
		<code>\setglossarystyle</code>	195, 218,
		258, 278–290, 292–314, 317–320, 323, 324	
		<code>\setglossentrycompatibility</code> ...	208, 218
		<code>\setkeys</code> 26, 30, 35, 45, 86, 115, 161, 162, 195,	
		227, 242, 244, 246, 248, 251, 253, 255, 257	
		<code>\setlength</code>	280, 291, 297–
		300, 302, 309, 316, 318, 319, 323, 339, 340	
		<code>\SetSmallAcronymDisplayStyle</code> ..	253, 254
		<code>\SetSmallAcronymStyle</code>	256
		<code>\settoheight</code>	125
		<code>\settowidth</code>	231, 321–323, 339
		<code>\sfcode</code>	11
		<code>\show</code>	258–264, 381, 382
		<code>\small</code>	6

W	
\warn@nomakeglossaries	178–180
\warn@noprintglossary	178–180, 197
\write	76, 98, 164– 170, 174–176, 180, 183, 197, 198, 327–331
\writeist	173, 174, 331
X	
\x	220
\xatlevel@	116
\xcapitalisewords	156
\xdef	86–89, 196, 273
\xglaccsupp	351
\xifinlistcs	198, 199, 203
xindy	385
xindy	9, 13, 23, 29, 30, 41, 42, 47, 50, 52, 54–56, 92, 123, 124, 163, 164, 166, 184, 188, 190, 197, 215, 264, 326
\xmakefirstuc ...	102–104, 110, 152, 154, 266
\xspace	222
xspace package	4, 222
Y	
\year	164, 168, 327, 330
Z	
\z@	290