

Documented Code For glossaries v4.41

Nicola L.C. Talbot

Dickimaw Books

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

2018-07-23

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 v4.41: 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	36
1.4	Xindy	46
1.5	Loops and conditionals	55
1.6	Defining new glossaries	61
1.7	Defining new entries	66
1.8	Resetting and unsetting entry flags	92
1.9	Keeping Track of How Many Times an Entry Has Been Unset	95
1.10	Loading files containing glossary entries	100
1.11	Using glossary entries in the text	100
1.12	Adding an entry to the glossary without generating text	159
1.13	Creating associated files	161
1.14	Writing information to associated files	180
1.15	Glossary Entry Cross-References	189
1.16	Displaying the glossary	191
1.17	Acronyms	219
1.18	Predefined acronym styles	223
1.19	Predefined Glossary Styles	255
1.20	Debugging Commands	256
1.21	Compatibility with version 2.07 and below	261
2	Prefix Support (glossaries-prefix Code)	262
3	Glossary Styles	269
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	269
3.2	In-line Style (glossary-inline.sty)	271
3.3	List Style (glossary-list.sty)	274
3.4	Glossary Styles using longtable (the glossary-long package)	277
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	283
3.6	Glossary Styles using longtable (the glossary-longragged package)	288
3.7	Glossary Styles using multicol (glossary-mcols.sty)	293
3.8	Glossary Styles using supertabular environment (glossary-super package)	299
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	306
3.10	Tree Styles (glossary-tree.sty)	312

4 Backwards Compatibility	322
4.1 glossaries-compatible-207	322
4.2 glossaries-compatible-307	328
5 Accessibility Support (glossaries-accsupp Code)	342
5.1 Defining Replacement Text	343
5.2 Accessing Replacement Text	346
5.3 Displaying the Glossary	362
5.4 Acronyms	363
5.5 Debugging Commands	378
6 Multi-Lingual Support	380
6.1 Polyglossia Captions	380
Glossary	382
Change History	383
Index	407

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2018/07/23 v4.41 (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`:

```
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@seeautonumberlist}{}

```

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

```

156 \@gls@declareoption{seeautonumberlist}{%
157   \renewcommand*{\@glo@seeautonumberlist}{%
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 }
```

glscheck@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 }
```

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

`glossariesNowarn` Define package option to suppress warnings

```

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

`glossariesMissingLangWarn` 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 }
```

`glossariesNoLangWarn` Suppress warning if language support not found.

```

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

`glossariesNonglossDefined` 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.

```
531 \newcommand*\glsnoidxstripaccents{%
532   \let\IeC\@firstofone
533   \let\' \@firstofone
534   \let\' \@firstofone
535   \let\~ \@firstofone
536   \let\" \@firstofone
537   \let\u \@firstofone
538   \let\t \@firstofone
539   \let\d \@firstofone
540   \let\r \@firstofone
541   \let\= \@firstofone
542   \let\.\@firstofone
543   \let\~ \@firstofone
544   \let\v \@firstofone
545   \let\H \@firstofone
546   \let\c \@firstofone
547   \let\b \@firstofone

548   \let\a \@secondoftwo
549   \def\AE{AE}%
550   \def\ae{ae}%
551   \def\OE{OE}%
552   \def\oe{oe}%
553   \def\AA{AA}%
554   \def\aa{aa}%
555   \def\L{L}%
556   \def\l{l}%
557   \def\O{O}%
558   \def\o{o}%
559   \def\SS{SS}%
```

```

560 \def\ss{ss}%
561 \def\th{th}%

562 \def\TH{TH}%
563 \def\dh{dh}%
564 \def\DH{DH}%
565 }

```

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.

```

566 \define@boolkey[glS]{sanitize}{description}[true]{%
567   \GlossariesWarning{sanitize={description} package option deprecated}%
568   \ifglS@sanitize@description
569     \glSsetnoexpandfield{desc}%
570     \glSsetnoexpandfield{descplural}%
571   \else
572     \glSsetexpandfield{desc}%
573     \glSsetexpandfield{descplural}%
574   \fi
575 }

576 \define@boolkey[glS]{sanitize}{name}[true]{%
577   \GlossariesWarning{sanitize={name} package option deprecated}%
578   \ifglS@sanitize@name
579     \glSsetnoexpandfield{name}%
580   \else
581     \glSsetexpandfield{name}%
582   \fi
583 }

584 \define@boolkey[glS]{sanitize}{symbol}[true]{%
585   \GlossariesWarning{sanitize={symbol} package option deprecated}%
586   \ifglS@sanitize@symbol
587     \glSsetnoexpandfield{symbol}%
588     \glSsetnoexpandfield{symbolplural}%
589   \else
590     \glSsetexpandfield{symbol}%
591     \glSsetexpandfield{symbolplural}%
592   \fi
593 }

```

sanitizesort

```

594 \define@boolkey{glossaries.sty}[glS]{sanitizesort}[true]{%
595   \ifglSsanitizesort
596     \glSsetnoexpandfield{sortvalue}%
597     \renewcommand*{\@glS@noidx@setsanitizesort}{%
598       \glSsanitizesorttrue
599       \glSsetnoexpandfield{sortvalue}%
600     }%
601   \else

```



```

602   \glsssetexpandfield{sortvalue}%
603   \renewcommand*{\@gls@noidx@setsanitizesort}{%
604     \glsssanitizesortfalse
605     \glsssetexpandfield{sortvalue}%
606   }%
607 \fi
608 }

```

Default setting:

```

609 \glsssanitizesorttrue
610 \glsssetnoexpandfield{sortvalue}%

```

`setsanitizesort` Default behaviour for `\makenoidxglossaries` is `sanitizesort=false`.

```

611 \newcommand*{\@gls@noidx@setsanitizesort}{%
612   \glsssanitizesortfalse
613   \glsssetexpandfield{sortvalue}%
614 }

615 \define@choicekey{gls}{sanitize}{sort}{true,false}[true]{%
616   \setbool{glsssanitizesort}{#1}%
617   \ifglsssanitizesort
618     \glsssetnoexpandfield{sortvalue}%
619   \else
620     \glsssetexpandfield{sortvalue}%
621   \fi
622   \GlossariesWarning{sanitize={sort} package option
623     deprecated. Use sanitizesort instead}%
624 }

```

`sanitize`

```

625 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
626   \ifthenelse{\equal{#1}{none}}{}%
627   {%
628     \GlossariesWarning{sanitize package option deprecated}%
629     \glsssetexpandfield{name}%
630     \glsssetexpandfield{symbol}%
631     \glsssetexpandfield{symbolplural}%
632     \glsssetexpandfield{desc}%
633     \glsssetexpandfield{descplural}%
634   }%
635   {%
636     \setkeys{gls}{sanitize}{#1}%
637   }%
638 }

```

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

```

639 \newif\ifglstranslate

```

`otranslatorhook` `\@gls@notranslatorhook` has been removed.

s@usetranslator

```
640 \newcommand*{@gls@usetranslator}{%
    polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
    polyglossia as well.
641   \@ifpackageloaded{polyglossia}{%
642     {%
643       \let\glsifusetranslator\@secondoftwo
644     }%
645     {%
646       \@ifpackageloaded{babel}{%
647         {%
648           \IfFileExists{translator.sty}{%
649             {%
650               \RequirePackage{translator}%
651               \let\glsifusetranslator\@firstoftwo
652             }%
653           }%
654         }%
655       }%
656     }%
657 }
```

dtranslatordict Checks if given translator dictionary has been loaded.

```
658 \newcommand{\glsifusedtranslatordict}[3]{%
659   \glsifusetranslator
660   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
661   {#3}%
662 }
```

notranslate Provide a synonym for translate=false that can be passed via the document class.

```
663 \@gls@declareoption{notranslate}{%
664   \glstranslatefalse
665   \let@gls@usetranslator\relax
666   \let\glsifusetranslator\@secondoftwo
667 }
```

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

```
668 \define@choicekey{glossaries.sty}{translate}{%
669   [\gls@translate@val\gls@translate@nr}%
670   {true,false,babel}[true]%
671   {%
672     \ifcase\gls@translate@nr\relax
673     \glstranslatetrue
674     \renewcommand*{@gls@usetranslator}{%
675       \@ifpackageloaded{polyglossia}{%
676         {%
677           \let\glsifusetranslator\@secondoftwo
678         }%

```

```

679      {%
680      \@ifpackageloaded{babel}%
681      {%
682      \IfFileExists{translator.sty}%
683      {%
684      \RequirePackage{translator}%
685      \let\glsifusetranslator\@firstoftwo
686      }%
687      {}%
688      }%
689      {}%
690      }%
691      }%
692      \or
693      \glstranslatefalse
694      \let\@gls@usetranslator\relax
695      \let\glsifusetranslator\@secondoftwo
696      \or
697      \glstranslatetrue
698      \let\@gls@usetranslator\relax
699      \let\glsifusetranslator\@secondoftwo
700      \fi
701      }

```

Set the default value:

```

702 \glstranslatefalse
703 \let\glsifusetranslator\@secondoftwo
704 \@ifpackageloaded{translator}%
705 {%
706   \glstranslatetrue
707   \let\glsifusetranslator\@firstoftwo
708 }%
709 {%
710   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
711   {
712     \@ifpackageloaded{\gls@thissty}%
713     {%
714       \glstranslatetrue
715       \@endfortrue
716     }%
717     {}%
718   }
719 }

```

indexonlyfirst Set whether to only index on first use.

```

720 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
721 \glsindexonlyfirstfalse

```

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

```

722 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
723 \glshyperfirsttrue

gls@setacrstyle  Keep track of whether an acronym style has been set (for the benefit of \setupglossaries):
724 \newcommand*{\@gls@setacrstyle}{}

footnote  Set the long form of the acronym in footnote on first use.
725 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{%
726   \ifbool{glsacrdescription}%
727   {}%
728   {%
729     \renewcommand*{\@gls@sanitizedesc}{}%
730   }%
731   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
732 }

description  Allow acronyms to have a description (needs to be set using the description key in the optional
              argument of \newacronym).
733 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
734   \renewcommand*{\@gls@sanitizesymbol}{}%
735   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
736 }

smallcaps  Define \newacronym to set the short form in small capitals.
737 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
738   \renewcommand*{\@gls@sanitizesymbol}{}%
739   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
740 }

smaller  Define \newacronym to set the short form using \smaller which obviously needs to be de-
         fined by loading the appropriate package.
741 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
742   \renewcommand*{\@gls@sanitizesymbol}{}%
743   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
744 }

dua  Define \newacronym to always use the long forms (i.e. don't use acronyms)
745 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
746   \renewcommand*{\@gls@sanitizesymbol}{}%
747   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
748 }

shortcuts  Define acronym shortcuts.
749 \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.
750 \newcommand*{\glsorder}{word}

```

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

```
751 \newcommand*{\@glsorder}[1]{}

order
752 \define@choicekey{glossaries.sty}{order}{word,letter}{%
753   \def\glsorder{#1}}
```

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

```
754 \newif\ifglxindy

The default is makeindex:
755 \glxindyfalse
```

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

```
756 \@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.

```
757 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
758 \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.)

```
759 \def\@xdy@main@language{\language}%

Define key to set the language
760 \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.

```
761 \ifcsundef{inputencodingname}{%
762   \def\gls@codepage{}}{%
763   \def\gls@codepage{\inputencodingname}
764 }
```

Define a key to set the code page.

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

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

```
766 \define@key{glossaries.sty}{xindy}[]{%
767   \glxindytrue
768   \setkeys[gls]{xindy}{#1}%
769 }
```

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

```

770 \@gls@declareoption{xindygloss}{%
771   \glsxindytrue
772 }

```

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

```

773 \@gls@declareoption{xindynoglsnumbers}{%
774   \glsxindytrue
775   \gls@xindy@glsnumbersfalse
776 }

```

automake If this setting is on, automatically run **makeindex/xindy** at the end of the document. Must be used with `\makeglossaries`. Default is false.

```

777 \define@boolkey{glossaries.sty}[gls]{automake}[true]{%
778   \ifglsautomake
779     \renewcommand*{\@gls@doautomake}{%
780       \PackageError{glossaries}{You must use
781       \string\makeglossaries\space with automake=true}
782       {%
783         Either remove the automake=true setting or
784         add \string\makeglossaries\space to your document preamble.%
785       }%
786     }%
787   \else
788     \renewcommand*{\@gls@doautomake}{}%
789   \fi
790 }
791 \glsautomakefalse

```

@gls@doautomake

```

792 \newcommand*{\@gls@doautomake}{}
793 \AtEndDocument{\@gls@doautomake}

```

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

```

794 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
795   \ifglssavewrites
796     \renewcommand*{\glswritefiles}{\@glswritefiles}%
797   \else
798     \let\glswritefiles\@empty
799   \fi
800 }

```

Set default:

```

801 \glssavewritesfalse
802 \let\glswritefiles\@empty

```

compatible-3.07

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

compatible-2.07

```
805 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{}
Also set 3.07 compatibility if this option is set.
806 \ifbool{glscompatible-2.07}%
807 {%
808   \booltrue{glscompatible-3.07}%
809 }%
810 {}%
811 }
812 \boolfalse{glscompatible-2.07}
```

al@makeglossary Store the original definition.

```
813 \let\gls@original@makeglossary\makeglossary
```

iginal@glossary Store the original definition.

```
814 \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.)

```
815 \def\makeglossary{%
816   \GlossariesWarning{Use of \string\makeglossary\space with
817   glossaries.sty is \MessageBreak deprecated. Use \string\makeglossaries\space
818   instead. If you \MessageBreak need the original definition of
819   \string\makeglossary\space use \MessageBreak the package options
820   kernelglossredefs=false (to \MessageBreak restore the former definition of
821   \string\makeglossary) and \MessageBreak nomain (if the file extensions cause a
822   conflict)}}%
823   \makeglossaries
824 }
```

erride@glossary

```
825 \newcommand*{\@gls@override@glossary}[1][main]{%
826   \GlossariesWarning{Use of \string\glossary\space with
827   glossaries.sty is deprecated. \MessageBreak Indexing should be performed
828   with the user level \MessageBreak commands, such as \string\gls\space or
829   \string\glsadd. If you need the \MessageBreak original definition of
830   \string\glossary\space use the package \MessageBreak options
831   kernelglossredefs=false (to restore the \MessageBreak former definition of
832   \string\glossary) and nomain (if the \MessageBreak file extensions cause a
833   conflict)}}%
834   \gls@glossary{#1}%
835 }
```

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`

```
836 \if@gls@docloaded
837 \else
838   \def\glossary{\@gls@override@glossary}
839 \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.

```
840 \define@choicekey{glossaries.sty}{kernelglossredefs}%
841   [\gls@debug@val\gls@debug@nr]{true,false,nowarn}[true]%
842 {%
843   \ifcase\gls@debug@nr\relax
844     \def\glossary{\@gls@override@glossary}%
845     \def\makeglossary{%
846       \GlossariesWarning{Use of \string\makeglossary\space with
847         glossaries.sty is deprecated. Use \string\makeglossaries\space
848         instead. If you need the original definition of
849         \string\makeglossary\space use the package options
850         kernelglossredefs=false (to prevent redefinition of
851         \string\makeglossary) and nomain (if the file extensions cause a
852         conflict)}}%
853     \makeglossaries
854   }%
855   \or
856     \let\glossary\gls@original@glossary
857     \let\makeglossary\gls@original@makeglossary
858   \or
859     \def\makeglossary{\makeglossaries}%
860     \renewcommand*{\@gls@override@glossary}[1][main]{%
861       \gls@glossary{##1}%
862     }%
863   \fi
864 }
```

`symbols` Create a “symbols” glossary type

```
865 \@gls@declareoption{symbols}{%}
```



```

866 \let\@gls@do@symbolsdef\@gls@symbolsdef
867 }

```

Default is not to define the symbols glossary:

```

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

```

@gls@symbolsdef

```

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

Define hook to set the toc title when translator is in use.
872 \newcommand*{\gls@tr@set@symbols@toctitle}{%
873 \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
874 }%
875 }%

```

numbers Create a “symbols” glossary type

```

876 \@gls@declareoption{numbers}{%
877 \let\@gls@do@numbersdef\@gls@numbersdef
878 }

```

Default is not to define the numbers glossary:

```

879 \newcommand*{\@gls@do@numbersdef}{}

```

@gls@numbersdef

```

880 \newcommand*{\@gls@numbersdef}{%
881 \newglossary[nlg]{numbers}{nls}{nlo}{\glsnumbersgroupname}%
882 \newcommand*{\printnumbers}[1] [] {\printglossary[type=numbers,##1]}%

Define hook to set the toc title when translator is in use.
883 \newcommand*{\gls@tr@set@numbers@toctitle}{%
884 \translatelet{\glossarytoctitle}{Numbers (glossaries)}%
885 }%
886 }%

```

index Create an “index” glossary type

```

887 \@gls@declareoption{index}{%
888 \let\@gls@do@indexdef\@gls@indexdef
889 }

```

Default is not to define index glossary:

```

890 \newcommand*{\@gls@do@indexdef}{}

```

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

```

891 \newcommand*{\@gls@indexdef}{%
892 \newglossary[ilg]{index}{ind}{idx}{\indexname}%
893 \newcommand*{\printindex}[1] [] {\printglossary[type=index,##1]}%
894 \newcommand*{\newterm}[2] [] {%
895 \newglossaryentry{##2}%
896 {type={index},name={##2},description={\nopostdesc},##1}}
897 }%

```

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

```

898 \@for\CurrentOption := \@declaredoptions\do{%
899   \ifx\CurrentOption\@empty
900   \else
901     \@expandtwoargs
902     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
903     \ifin@
904     \@use@option
905     \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
906   \fi
907 \fi
908 }

```

Now process options passed to the package:

```

909 \ProcessOptionsX

```

Load backward compatibility stuff:

```

910 \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:

```

911 \disable@keys{glossaries.sty}{compatible-2.07,%
912 xindy,xindygloss,xindynoglsnumbers,makeindex,%
913 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,nomain}

```

Now define `\setupglossaries`:

```

914 \newcommand*{\setupglossaries}[1]{%
915   \renewcommand*{\@gls@setacrstyle}{}%
916   \ifglsacrshortcuts
917     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
918   \else
919     \def\@gls@setupshortcuts{%
920       \ifglsacrshortcuts
921         \DefineAcronymSynonyms
922       \fi
923     }%
924   \fi
925   \glsacrshortcutsfalse
926   \let\@gls@do@numbersdef\relax
927   \let\@gls@do@symbolssdef\relax
928   \let\@gls@do@indexdef\relax
929   \let\@gls@do@acronymsdef\relax
930   \ifglsentrycounter
931     \let\@gls@doentrycounterdef\relax
932   \else
933     \let\@gls@doentrycounterdef\@gls@define@glossaryentrycounter
934   \fi
935   \ifglssubentrycounter
936     \let\@gls@dosubentrycounterdef\relax

```

```

937 \else
938   \let\@gls@dosubentrycounterdef\@gls@define@glossarysubentrycounter
939 \fi
940 \setkeys{glossaries.sty}{#1}%
941 \@gls@setacrstyle
942 \@gls@setupshortcuts
943 \@gls@do@acronymsdef
944 \@gls@do@numbersdef
945 \@gls@do@symbolssdef
946 \@gls@do@indexdef
947 \@gls@doentrycounterdef
948 \@gls@dosingentrycounterdef
949 }

```

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

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

```

950 \ifthenelse{\equal{\glscounter}{section}}{%
951 {%
952   \ifcsundef{chapter}{}%
953   {%
954     \let\@gls@old@chapter\@chapter
955     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
956       \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}}}%
957   }%
958 }%
959 {}

```

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

```

960 \newcommand*\@gls@onlypremakeg{}

```

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

```

961 \newcommand*\@onlypremakeg}[1]{%
962   \ifx\@gls@onlypremakeg\@empty
963     \def\@gls@onlypremakeg{#1}%
964   \else
965     \expandafter\toks@ \expandafter{\@gls@onlypremakeg}%
966     \edef\@gls@onlypremakeg{\the\toks@, \noexpand#1}%
967   \fi
968 }

```

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

```

969 \newcommand*{\@disable@onlypremakeg}{%
970 \@for\@thiscs:=\@gls@onlypremakeg\do{%
971   \expandafter\@disable@premakecs\@thiscs%
972 }}

```

`\sable@premakecs` Disables the given command.

```

973 \newcommand*{\@disable@premakecs}[1]{%
974   \def#1{\PackageError{glossaries}{\string#1\space may only be
975     used before \string\makeglossaries}{You can't use
976     \string#1\space after \string\makeglossaries}}}%
977 }

```

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`

```

978 \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`

```

979 \providecommand*\acronymname{Acronyms}

```

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

```

980 \newcommand*{\glssettoctitle}[1]{%
981   \def\glossarytoctitle{\csname @gls@#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`

```

982 \providecommand*\entryname{Notation}

```

`\descriptionname`

```

983 \providecommand*\descriptionname{Description}

```

`\symbolname`

```

984 \providecommand*\symbolname{Symbol}

```

`\pagelistname`

```

985 \providecommand*\pagelistname{Page List}

```

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

ymbolsgroupname

```
986 \providecommand*\glssymbolsgroupname{\Symbols}
```

umbersgroupname

```
987 \providecommand*\glsnnumbersgroupname{\Numbers}
```

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

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

acrpluralsuffix Default plural suffix for acronyms

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

acrpluralsuffix

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

\seename

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

\andname

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

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

eGlossariesLang

```
993 \newcommand*\RequireGlossariesLang[1]{%
994   \@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}%
995 }
```

sGlossariesLang

```
996 \newcommand*\ProvidesGlossariesLang[1]{%
997   \ProvidesFile{glossaries-#1.ldf}%
998 }
```

ssarytocaptions Does nothing if translator hasn't been loaded.

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

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

```
1000 \ifglstranslate
```

Load `tracklang`

```
1001 \RequirePackage{tracklang}
```

Load translator if required.

```
1002 \@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.)

```
1003 \ifpackageloaded{translator}
1004 {%
```

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.

```
1005 \ifboolexpr
1006 {
1007   test {\ifdefstring{\trans@languages}{English}}
1008   and not
1009   test {\ifdefstring{bbl@loaded}{english}}
1010 }
1011 {%
1012 \let\glsifusetranslator\@secondoftwo
1013 }%
1014 {%
1015   \usedictionary{glossaries-dictionary}%
1016   \renewcommand*{\addglossarytocaptions}[1]{%
1017     \ifcsundef{captions#1}{}%
1018     {%
1019       \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
1020       \expandafter\toks@\expandafter{\@gls@tmp
1021         \renewcommand*{\glossaryname}{\translate{Glossary}}}%
1022       }%
1023       \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
1024     }%
1025   }%
1026 }%
1027 }%
1028 {}%
```

Check for tracked languages

```
1029 \AnyTrackedLanguages
1030 {%
1031   \ForEachTrackedDialect{\this@dialect}{%
1032     \IfTrackedLanguageFileExists{\this@dialect}%
1033     {glossaries-}% prefix
1034     {.ldf}%
1035     {%
1036       \RequireGlossariesLang{\CurrentTrackedTag}%
1037     }%
1038     {%
1039       \@gls@missinglang@warn\this@dialect\CurrentTrackedLanguage
1040     }%
1041   }%
```

```

1042 }%
1043 {}%
    if using translator use translator interface.
1044 \glsifusetranslator
1045 {%
1046   \renewcommand*{\glssettoctitle}[1]{%
1047     \ifcsdef{gls@tr@set@#1@toctitle}%
1048       {%
1049         \csuse{gls@tr@set@#1@toctitle}%
1050       }%
1051       {%
1052         \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
1053       }%
1054     }%
1055   \renewcommand*{\glossaryname}{\translate{Glossary}}}%
1056   \renewcommand*{\acronymname}{\translate{Acronyms}}}%
1057   \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
1058   \renewcommand*{\descriptionname}{%
1059     \translate{Description (glossaries)}}}%
1060   \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
1061   \renewcommand*{\pagelistname}{%
1062     \translate{Page List (glossaries)}}}%
1063   \renewcommand*{\glssymbolsgroupname}{%
1064     \translate{Symbols (glossaries)}}}%
1065   \renewcommand*{\glsnumbersgroupname}{%
1066     \translate{Numbers (glossaries)}}}%
1067 }{}%
1068 \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.

```
1069 \DeclareRobustCommand*{\nopostdesc}{}
```

`\@nopostdesc` Suppress next description terminator.

```

1070 \newcommand*{\@nopostdesc}{%
1071   \let\org@glspostdescription\glspostdescription
1072   \def\glspostdescription{%
1073     \let\glspostdescription\org@glspostdescription}%
1074 }

```

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

```
1075 \newcommand*{\@no@post@desc}{\nopostdesc}
```

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

```
1076 \newcommand{\glspar}{\par}
```

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

```

1077 \newcommand{\setStyleFile}[1]{%
1078   \renewcommand*{\gls@istfilebase}{#1}%

```

Just in case `\istfilename` has been modified.

```
1079 \ifglxindy
1080   \def\istfilename{\gls@istfilebase.xdy}
1081 \else
1082   \def\istfilename{\gls@istfilebase.ist}
1083 \fi
1084 }
```

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

```
1085 \@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`

```
1086 \ifglxindy
1087   \def\istfilename{\gls@istfilebase.xdy}
1088 \else
1089   \def\istfilename{\gls@istfilebase.ist}
1090 \fi
```

`gls@istfilebase`

```
1091 \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`

```
1092 \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`

```
1093 \newcommand*{\glscompositor}{.}
```

`lsSetCompositor` Sets the compositor.

```
1094 \newcommand*{\glsSetCompositor}[1]{%
1095   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
1096 \@onlypremakeg\glsSetCompositor
```


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

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

```
1097 \newcommand*{\@glsAlphacompositor}{\glscompositor}
```

AlphaCompositor Sets the alpha compositor.

```
1098 \ifglsxindy
1099   \newcommand*\glsSetAlphaCompositor[1]{%
1100     \renewcommand*\@glsAlphacompositor{#1}}
1101 \else
1102   \newcommand*\glsSetAlphaCompositor[1]{%
1103     \glsnoxywarning\glsSetAlphaCompositor}
1104 \fi
```

Can only be used before `\makeglossaries`

```
1105 \@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.

```
1106 \newcommand*{\gls@suffixF}{}
```

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

```
1107 \newcommand*{\glsSetSuffixF}[1]{%
1108   \renewcommand*{\gls@suffixF}{#1}}
```

Only has an effect when used before `\makeglossaries`

```
1109 \@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.

```
1110 \newcommand*{\gls@suffixFF}{}
```

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

```
1111 \newcommand*{\glsSetSuffixFF}[1]{%
1112   \renewcommand*{\gls@suffixFF}{#1}%
1113 }
```

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 `\glsnumber`, otherwise it will simply display its argument “as is”.

```

1114 \ifcsundef{hyperlink}%
1115 {%
1116   \newcommand*{\glsnumberformat}[1]{#1}%
1117 }%
1118 {%
1119   \newcommand*{\glsnumberformat}[1]{\glsnumberformat{#1}}%
1120 }

```

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`

```

1121 \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`

```

1122 \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`

```

1123 \newcommand*{\glossarypreamble}{%
1124   \csuse{@glossarypreamble@\currentglossary}%
1125 }

```

`\setglossarypreamble`

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

Code provided by Michael Pock.

```

1126 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
1127   \ifglossaryexists{#1}{%
1128     \csgdef{@glossarypreamble@#1}{#2}%
1129   }{%
1130     \GlossariesWarning{%
1131       Glossary '#1' is not defined%
1132     }%
1133   }%
1134 }

```

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

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

`\glossarypostamble`

```
1135 \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`.

```
1136 \newcommand*{\glossarysection}[2][\@gls@title]{%
1137   \def\@gls@title{#2}%
1138   \ifcsundef{phantomsection}%
1139     {%
1140       \@glossarysection{#1}{#2}%
1141     }%
1142   {%
1143     \p@glossarysection{#1}{#2}%
1144   }%

1145   \glsglossarymark{\glossarytoctitle}%
1146 }
```

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

```
1147 \ifcsundef{glossarymark}%
1148 {%
1149   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1150 }%
1151 {%
1152   \@ifclassloaded{memoir}
1153   {%
1154     \newcommand{\glsglossarymark}[1]{%
1155       \ifglsucmark
1156         \markboth{\memUHead{#1}}{\memUHead{#1}}%
1157       \else
1158         \markboth{#1}{#1}%
1159       \fi
1160     }
1161   }%
1162   {%
1163     \newcommand{\glsglossarymark}[1]{%
1164       \ifglsucmark
1165         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1166       \else
1167         \markboth{#1}{#1}%
1168       \fi
1169     }
1170   }%
1171 }
```

```

1166     \else
1167     \mkboth{#1}{#1}%
1168     \fi
1169   }
1170 }
1171 }

```

`\glossarymark` Provided for backward compatibility:

```

1172 \providecommand{\glossarymark}[1]{%
1173   \ifglsucmark
1174     \mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1175   \else
1176     \mkboth{#1}{#1}%
1177   \fi
1178 }

```

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`

```

1179 \newcommand*{\setglossarysection}[1]{%
1180 \setkeys{glossaries.sty}{section=#1}}

```

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

`\glossarysection`

```

1181 \newcommand*{\@glossarysection}[2]{%
1182   \ifdefempty\@glossarysecstar
1183   {%
1184     \csname\@glossarysec\endcsname[#1]{#2}%
1185   }%
1186   {%
1187     \csname\@glossarysec\endcsname*{#2}%
1188     \@gls@toc{#1}{\@glossarysec}%
1189   }%

```

Do automatic labelling if required

```

1190   \@glossaryseclabel
1191 }

```

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`

```

1192 \newcommand*{\@pglossarysection}[2]{%
1193   \glsclearpage

```

```

1194 \phantomsection
1195 \ifdefempty\@glossarysecstar
1196 {%
1197   \csname\@glossarysec\endcsname{#2}%
1198 }%
1199 {%
1200   \@gls@toc{#1}{\@glossarysec}%
1201   \csname\@glossarysec\endcsname*{#2}%
1202 }%

```

Do automatic labelling if required

```

1203 \@glossaryseclabel
1204 }

```

`\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.

```

1205 \newcommand*{\gls@doclearpage}{%
1206   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1207     {%
1208       \ifcsundef{cleardoublepage}%
1209       {%
1210         \clearpage
1211       }%
1212     }%
1213     \ifcsdef{if@openright}%
1214     {%
1215       \if@openright
1216         \cleardoublepage
1217       \else
1218         \clearpage
1219       \fi
1220     }%
1221     {%
1222       \cleardoublepage
1223     }%
1224   }%
1225 }%
1226 {}%
1227 }

```

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

```

1228 \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`

```
1229 \newcommand*{\@gls@toc}[2]{%
1230   \ifglstoc
1231     \ifglsnumberline
1232       \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1233     \else
1234       \addcontentsline{toc}{#2}{#1}%
1235     \fi
1236   \fi
1237 }
```

1.4 Xindy

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

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

```
1238 \newcommand*{\glsnoindexwarning}[1]{%
1239   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1240 }
```

`\glsnoindexwarning` Reverse for commands that may only be used with makeindex.

```
1241 \newcommand*{\glsnomakeindexwarning}[1]{%
1242   \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1243 }
```

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

```
1244 \ifglsxindy
1245   \edef\@xdyattributes{\string"default\string"}%
1246 \fi
```

`\@xdyattributelist` Comma-separated list of attributes.

```
1247 \ifglsxindy
1248   \edef\@xdyattributelist{}%
1249 \fi
```

`\@xdylocref` Define list of markup location references.

```
1250 \ifglsxindy
1251   \def\@xdylocref{}
1252 \fi
```

`\@gls@ifinlist`

```
1253 \newcommand*{\@gls@ifinlist}[4]{%
1254   \def\@do@ifinlist##1,#1,##2\end@do@ifinlist{%
1255     \def\@gls@listsuffix{##2}%
1256     \ifx\@gls@listsuffix\@empty
```

```

1257      #4%
1258    \else
1259      #3%
1260    \fi
1261  }%
1262  \@do@ifinlist,#2,#1,\end@do@ifinlist
1263 }

```

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.

```

1264 \ifglxsindy
1265   \newcommand*{\@xdycounters}{\@glscounter}
1266   \newcommand*\GlsAddXdyCounters[1]{%
1267     \@for\@gls@ctr:=#1\do{%

```

Check if already in list before adding.

```

1268       \edef\@do@addcounter{%
1269         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1270         {%
1271           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1272             \noexpand\@gls@ctr}%
1273         }%
1274       }%
1275       \@do@addcounter
1276     }
1277   }

```

Only has an effect before `\writeist`:

```

1278   \@onlypremakeg\GlsAddXdyCounters
1279 \else
1280   \newcommand*\GlsAddXdyCounters[1]{%
1281     \glsnoxindywarning\GlsAddXdyAttribute
1282   }
1283 \fi

```

saddxdycounters Counters must all be identified before adding attributes.

```

1284 \newcommand*\@disabled@glssaddxdycounters{%
1285   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1286     can't be used after \string\GlsAddXdyAttribute}{Move all
1287     occurrences of \string\GlsAddXdyCounters\space before the first
1288     instance of \string\GlsAddXdyAttribute}%
1289 }

```

AddXdyAttribute Adds an attribute.

```

1290 \ifglxsindy

```

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

```

1291   \newcommand*\@glssaddxdyattribute[2]{%

```

Add to xindy attribute list

```
1292 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1293 \string"#2#1\string"}%
```

Add to xindy markup location.

```
1294 \expandafter\toks@\expandafter{\@xdylocref}%
1295 \edef\@xdylocref{\the\toks@ ^^J%
1296 (markup-locref
1297 :open \string"glstildechar n%
1298 \expandafter\string\csname glsX#2X#1\endcsname
1299 \string" ^^J
1300 :close \string"\string" ^^J
1301 :attr \string"#2#1\string")}%
```

Define associated attribute command `\glsX<counter>X<attribute>{\<Hprefix>}{\<n>}`

```
1302 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1303 \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1304 }%
1305 }
```

High-level command:

```
1306 \newcommand*\GlsAddXdyAttribute[1]{%
```

Add to comma-separated attribute list

```
1307 \ifx\@xdyattributelist\@empty
1308 \edef\@xdyattributelist{#1}%
1309 \else
1310 \edef\@xdyattributelist{\@xdyattributelist,#1}%
1311 \fi
```

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

```
1312 \@for\@this@counter:=\@xdycounters\do{%
1313 \protected@edef\gls@do@addxdyattribute{%
1314 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1315 }
1316 \gls@do@addxdyattribute
1317 }%
```

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

```
1318 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1319 }
```

Only has an effect before `\writeist`:

```
1320 \@onlypremakeg\GlsAddXdyAttribute
1321 \else
1322 \newcommand*\GlsAddXdyAttribute[1]{%
1323 \glsnoxindywarning\GlsAddXdyAttribute}
1324 \fi
```

`definedattributes` Add known attributes for all defined counters

```
1325 \ifglsxindy
1326 \newcommand*\@gls@addpredefinedattributes{%
```



```

1327 \GlsAddXdyAttribute{glsnumberformat}
1328 \GlsAddXdyAttribute{textrm}
1329 \GlsAddXdyAttribute{textsf}
1330 \GlsAddXdyAttribute{texttt}
1331 \GlsAddXdyAttribute{textbf}
1332 \GlsAddXdyAttribute{textmd}
1333 \GlsAddXdyAttribute{textit}
1334 \GlsAddXdyAttribute{textup}
1335 \GlsAddXdyAttribute{textsl}
1336 \GlsAddXdyAttribute{textsc}
1337 \GlsAddXdyAttribute{emph}
1338 \GlsAddXdyAttribute{glshypernumber}
1339 \GlsAddXdyAttribute{hyperrm}
1340 \GlsAddXdyAttribute{hypersf}
1341 \GlsAddXdyAttribute{hypertt}
1342 \GlsAddXdyAttribute{hyperbf}
1343 \GlsAddXdyAttribute{hypermd}
1344 \GlsAddXdyAttribute{hyperit}
1345 \GlsAddXdyAttribute{hyperup}
1346 \GlsAddXdyAttribute{hypersl}
1347 \GlsAddXdyAttribute{hypersc}
1348 \GlsAddXdyAttribute{hyperemph}

1349 \GlsAddXdyAttribute{glsignore}
1350 }
1351 \else
1352 \let\@gls@addpredefinedattributes\relax
1353 \fi

```

`\dyuseralphabets` List of additional alphabets

```

1354 \def\@xdyuseralphabets{}

```

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

```

1355 \ifglsxindy
1356 \newcommand*{\GlsAddXdyAlphabet}[2]{%
1357 \edef\@xdyuseralphabets{%
1358 \@xdyuseralphabets ^^J
1359 (define-alphabet "#1" (#2))}}
1360 \else
1361 \newcommand*{\GlsAddXdyAlphabet}[2]{%
1362 \glsnoxindywarning\GlsAddXdyAlphabet}
1363 \fi

```

This code is only required for xindy:

```

1364 \ifglsxindy

```

`\dy@locationlist` List of predefined location names.

```

1365 \newcommand*{\@gls@xdy@locationlist}{%

```

```

1366     roman-page-numbers,%
1367     Roman-page-numbers,%
1368     arabic-page-numbers,%
1369     alpha-page-numbers,%
1370     Alpha-page-numbers,%
1371     Appendix-page-numbers,%
1372     arabic-section-numbers%
1373 }

```

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.

```

1374 \protected@edef\@gls@roman{\@roman{0}\string"
1375     \string"roman-numbers-lowercase\string" :sep \string"}}%
1376 \@onelevel@sanitize\@gls@roman
1377 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1378     :sep \string"}%
1379 \@onelevel@sanitize\@tmp
1380 \ifx\@tmp\@gls@roman
1381     \expandafter
1382     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1383         \string"roman-numbers-lowercase\string"%
1384     }%
1385 \else
1386     \expandafter
1387     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1388         :sep \string"\@gls@roman\string"%
1389     }%
1390 \fi

```

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

```

1391 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1392     \string"roman-numbers-uppercase\string"%
1393 }%

```

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

```

1394 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1395     \string"arabic-numbers\string"%
1396 }%

```

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

```

1397 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1398     \string"alpha\string"%
1399 }%

```

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

```

1400 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1401 \string"ALPHA\string"%
1402 }%

```

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

```

1403 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1404 \string"ALPHA\string"
1405 :sep \string"\glsAlphacompositor\string"
1406 \string"arabic-numbers\string"%
1407 }

```

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

```

1408 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1409 \string"arabic-numbers\string"
1410 :sep \string"\glscompositor\string"
1411 \string"arabic-numbers\string"%
1412 }%

```

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

```

1413 \def\@xdyuserlocationdefs{}

```

userlocationnames List of additional user location names

```

1414 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

```

1415 \fi

```

xdycrossrefhook Hook used after writing cross-reference class information.

```

1416 \ifglsxindy
1417 \newcommand\@xdycrossrefhook{}
1418 \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.)

```

1419 \ifglsxindy
1420 \newcommand*\GlsAddXdyLocation{[3][[]]{%
1421 \def\@gls@tmp{#1}%
1422 \ifx\@gls@tmp\@empty
1423 \edef\@xdyuserlocationdefs{%
1424 \@xdyuserlocationdefs ^^J%
1425 (define-location-class \string"#2\string"^^J\space\space
1426 \space(:sep \string"{}\glsopenbrace\string" #3
1427 :sep \string"\glsclosebrace\string"))
1428 }%

```

```

1429 \else
1430 \edef\@xdyuserlocationdefs{%
1431 \@xdyuserlocationdefs ^^J%
1432 (define-location-class \string"#2\string"^^J\space\space
1433 \space(:sep "\glsoopenbrace"
1434 #1
1435 :sep "\glsclosebrace\glsoopenbrace" #3
1436 :sep "\glsclosebrace"))
1437 }%
1438 \fi

1439 \edef\@xdyuserlocationnames{%
1440 \@xdyuserlocationnames^^J\space\space\space
1441 \string"#2\string"}%
1442 }

```

Only has an effect before \writeist:

```

1443 \@onlypremakeg\GlsAddXdyLocation
1444 \else
1445 \newcommand*\{GlsAddXdyLocation}[2]{%
1446 \glsnnoxindywarning\GlsAddXdyLocation}
1447 \fi

```

ationclassorder Define location class order

```

1448 \ifglxindy
1449 \def\@xdylocationclassorder{^^J\space\space\space
1450 \string"roman-page-numbers\string"^^J\space\space\space
1451 \string"arabic-page-numbers\string"^^J\space\space\space
1452 \string"arabic-section-numbers\string"^^J\space\space\space
1453 \string"alpha-page-numbers\string"^^J\space\space\space
1454 \string"Roman-page-numbers\string"^^J\space\space\space
1455 \string"Alpha-page-numbers\string"^^J\space\space\space
1456 \string"Appendix-page-numbers\string"
1457 \@xdyuserlocationnames^^J\space\space\space
1458 \string"see\string"
1459 }
1460 \fi

```

Change the location order.

ationClassOrder

```

1461 \ifglxindy
1462 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1463 \def\@xdylocationclassorder{#1}}
1464 \else
1465 \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1466 \glsnnoxindywarning\GlsSetXdyLocationClassOrder}
1467 \fi

```

\@xdysortrules Define sort rules

```

1468 \ifglxindy
1469   \def\@xdysortrules{}
1470 \fi

```

`\GlsAddSortRule` Add a sort rule

```

1471 \ifglxindy
1472   \newcommand*\GlsAddSortRule[2]{%
1473     \expandafter\toks@\expandafter{\@xdysortrules}%
1474     \protected@edef\@xdysortrules{\the\toks@ ^^J
1475       (sort-rule \string"#1\string" \string"#2\string"))}%
1476   }
1477 \else
1478   \newcommand*\GlsAddSortRule[2]{%
1479     \glsnxindywarning\GlsAddSortRule}
1480 \fi

```

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

```

1481 \ifglxindy
1482   \def\@xdyrequiredstyles{tex}
1483 \fi

```

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

```

1484 \ifglxindy
1485   \newcommand*\GlsAddXdyStyle[1]{%
1486     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1487 \else
1488   \newcommand*\GlsAddXdyStyle[1]{%
1489     \glsnxindywarning\GlsAddXdyStyle}
1490 \fi

```

`\GlsSetXdyStyles` Reset the list of required styles

```

1491 \ifglxindy
1492   \newcommand*\GlsSetXdyStyles[1]{%
1493     \edef\@xdyrequiredstyles{#1}}
1494 \else
1495   \newcommand*\GlsSetXdyStyles[1]{%
1496     \glsnxindywarning\GlsSetXdyStyles}
1497 \fi

```

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

```

1498 \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.

```

1499 \def\@xdylanguage#1#2{}

```

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

```

1500 \ifglxindy
1501   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1502     \ifglossaryexists{#1}{%
1503       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1504     }{%
1505       \PackageError{glossaries}{Can't set language type for
1506         glossary type '#1' --- no such glossary}{%
1507         You have specified a glossary type that doesn't exist}}
1508 \else
1509   \newcommand*\GlsSetXdyLanguage[2][]{%
1510     \glsnoxywarning\GlsSetXdyLanguage}
1511 \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.

```

1512 \def\@gls@codepage#1#2{}

```

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

```

1513 \ifglxindy
1514   \newcommand*\GlsSetXdyCodePage[1]{%
1515     \renewcommand*\@gls@codepage{#1}%
1516   }

```

Suggested by egreg:

```

1517 \AtBeginDocument{%
1518   \ifx\@gls@codepage\@empty
1519     \ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{}%
1520   \fi
1521 }
1522 \else
1523   \newcommand*\GlsSetXdyCodePage[1]{%
1524     \glsnoxywarning\GlsSetXdyCodePage}
1525 \fi

```

`\@xdylettergroups` Store letter group definitions.

```

1526 \ifglxindy
1527   \ifglxindy@glsnumbers
1528     \def\@xdylettergroups{(define-letter-group
1529       \string"glxnumbers\string"^^J\space\space\space
1530       :prefixes (\string"0\string" \string"1\string"
1531         \string"2\string" \string"3\string" \string"4\string"
1532         \string"5\string" \string"6\string" \string"7\string"
1533         \string"8\string" \string"9\string")^^J\space\space\space
1534       \@xdynumbergrouporder)}
1535   \else
1536     \def\@xdylettergroups{}

```

```

1537 \fi
1538 \fi

```

sAddLetterGroup 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.

```

1539 \newcommand*\GlsAddLetterGroup[2]{%
1540   \expandafter\toks@\expandafter{\@xdylettergroups}%
1541   \protected@edef\@xdylettergroups{\the\toks@^^J%
1542   (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1543 }%

```

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.

```

1544 \newcommand*\forallglossaries[3][\@glo@types]{%
1545   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1546 }

```

\forallacronyms

```

1547 \newcommand*\forallacronyms[2]{%
1548   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1549 }

```

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

```
\forglsentries[⟨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.

```

1550 \newcommand*\forglsentries[3][\glsdefaulttype]{%
1551   \edef\@glo@list{\csname glolist@#1\endcsname}%
1552   \@for#2:=\@glo@list\do
1553   {%
1554     \ifdefempty{#2}{-}{#3}%
1555   }%
1556 }

```

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⟩}
```

```

1557 \newcommand*{\forallglsentries}[3][\@glo@types]{%
1558   \expandafter\forallglossaries\expandafter[#1]{\@@this@glo@}%
1559   {%
1560     \forglsentries[\@@this@glo@]{#2}{#3}%
1561   }%
1562 }

```

$$\text{\ifglossaryexists}\langle type \rangle\{\langle true-text \rangle\}\{\langle false-text \rangle\}$$

```

1563 \newcommand{\ifglossaryexists}[3]{%
1564   \ifcsundef{@glotype@#1@out}{#3}{#2}%
1565 }

```

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

\glsdetoklabel

```
1566 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

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

```

1567 \newcommand{\ifglentryexists}[3]{%
1568   \ifcsundef{glo@glsetoklabel{#1}@name}{#3}{#2}%
1569 }

```

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

$$\text{\ifglused{\langle label \rangle}\{\langle true text \rangle\}\{\langle false text \rangle\}}$$

where $\langle label \rangle$ is the entry's label. If true it will do $\langle true\ text \rangle$ otherwise it will do $\langle false\ text \rangle$.


```

1570 \newcommand*{\ifglsused}[3]{%
1571   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1572 }

```

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 exist, otherwise do *<code>*.

```

1573 \newcommand{\glsdoifexists}[2]{%
1574   \ifglsentryexists{#1}{#2}{%
1575     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1576       has not been defined}{You need to define a glossary entry before you
1577       can use it.}}%
1578 }

```

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

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

```

1579 \newcommand{\glsdoifnoexists}[2]{%
1580   \ifglsentryexists{#1}{%
1581     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1582       been defined}{}}{#2}%
1583 }

```

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

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

```

1584 \newcommand{\glsdoifexistsorwarn}[2]{%
1585   \ifglsentryexists{#1}{#2}{%
1586     \GlossariesWarning{Glossary entry '\glsdetoklabel{#1}'
1587       has not been defined}%
1588   }%
1589 }

```

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

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

```

1590 \newcommand{\glsdoifexistsordo}[3]{%
1591   \ifglsentryexists{#1}{#2}{%
1592     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1593       has not been defined}{You need to define a glossary entry before you
1594       can use it.}}{#3}%

```

```

1595     #3%
1596   }%
1597 }

```

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

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

```

1598 \newcommand{\doifglossarynoexistsordo}[3]{%
1599   \ifglossaryexists{#1}%
1600   {%
1601     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1602     #3%
1603   }%
1604   {#2}%
1605 }

```

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

```

1606 \newcommand{\ifglshaschildren}[3]{%
1607   \glstoifexists{#1}%
1608   {%
1609     \def\do@glshaschildren{#3}%
1610     \edef\@gls@thislabel{\glstoifexists{#1}}%
1611     \expandafter\for@glstentries\expandafter
1612     [\csname glo@\@gls@thislabel @type\endcsname]
1613     {\glo@label}%
1614     {%
1615       \letcs\glo@parent{glo@\glo@label @parent}%
1616       \ifdefequal\@gls@thislabel\glo@parent
1617       {%
1618         \def\do@glshaschildren{#2}%
1619         \@endfortrue
1620       }%
1621     }%
1622   }%
1623   \do@glshaschildren
1624 }%
1625 }

```

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

```

1626 \newcommand{\ifglshasparent}[3]{%
1627   \glstoifexists{#1}%
1628   {%
1629     \ifcempty{glo@\glstoifexists{#1}@parent}{#3}{#2}%

```

```

1630 }%
1631 }

\ifglshasdesc \ifglshasdesc{<label>}{<true part>}{<false part>}
1632 \newcommand*{\ifglshasdesc}[3]{%
1633 \ifcsequal{glo@glstdetoklabel{#1}@desc}%
1634 {#3}%
1635 {#2}%
1636 }

sdescsuppressed \ifglstdescsuppressed{<label>}{<true part>}{<false part>} Does <true part> if the descrip-
tion is just \nopostdesc otherwise does <false part>.
1637 \newcommand*{\ifglstdescsuppressed}[3]{%
1638 \ifcsequal{glo@glstdetoklabel{#1}@desc}{@no@post@desc}%
1639 {#2}%
1640 {#3}%
1641 }

\ifglshassymbol \ifglshassymbol{<label>}{<true part>}{<false part>}
1642 \newcommand*{\ifglshassymbol}[3]{%
1643 \letcs{\@glo@symbol}{glo@glstdetoklabel{#1}@symbol}%
1644 \ifdefempty\@glo@symbol
1645 {#3}%
1646 {%
1647 \ifdefequal\@glo@symbol\@gls@default@value
1648 {#3}%
1649 {#2}%
1650 }%
1651 }

\ifglshaslong \ifglshaslong{<label>}{<true part>}{<false part>}
1652 \newcommand*{\ifglshaslong}[3]{%
1653 \letcs{\@glo@long}{glo@glstdetoklabel{#1}@long}%
1654 \ifdefempty\@glo@long
1655 {#3}%
1656 {%
1657 \ifdefequal\@glo@long\@gls@default@value
1658 {#3}%
1659 {#2}%
1660 }%
1661 }

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1662 \newcommand*{\ifglshasshort}[3]{%
1663 \letcs{\@glo@short}{glo@glstdetoklabel{#1}@short}%
1664 \ifdefempty\@glo@short
1665 {#3}%
1666 {%

```

```

1667 \ifdefequal\@glo@short\@gls@default@value
1668   {#3}%
1669   {#2}%
1670 }%
1671 }

```

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

```

1672 \newcommand*{\ifglshasfield}[4]{%
1673   \glsdoifexists{#2}%
1674   {%
1675     \letcs{\@glo@thisvalue}{glo\@glsdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1676   \ifdef\@glo@thisvalue
1677   {%

```

Is defined, so now check if empty.

```

1678     \ifdefempty\@glo@thisvalue
1679     {%

```

Is empty, so doesn't have field set.

```

1680       #4%
1681     }%
1682   {%

```

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

```

1683     \ifdefequal\@glo@thisvalue\@gls@default@value
1684     {%

```

Value is set to the default value.

```

1685       #4%
1686     }%
1687   {%

```

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

```

1688     \let\glscurrentfieldvalue\@glo@thisvalue
1689     #3%
1690   }%
1691 }%
1692 }%
1693 {%

```

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

```

1694   \@gls@fetchfield{\@gls@thisfield}{#1}%

```

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

```

1695   \ifdef\@gls@thisfield
1696   {%

```

Is defined, so now check if empty.

```
1697      \letcs{\@glo@thisvalue}{glo@glsetoklabel{#2}@@gls@thisfield}%
1698      \ifdefempty\@glo@thisvalue
1699      {%
```

Is empty so field hasn't been set.

```
1700      #4%
1701      }%
1702      {%
```

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

```
1703      \ifdefequal\@glo@thisvalue\@gls@default@value
1704      {%
```

Value is set to the default value.

```
1705      #4%
1706      }%
1707      {%
```

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

```
1708      \let\glscurrentfieldvalue\@glo@thisvalue
1709      #3%
1710      }%
1711      }%
1712      }%
1713      {%
```

Not defined.

```
1714      \GlossariesWarning{Unknown entry field '#1'}%
1715      #4%
1716      }%
1717      }%
1718      }%
1719 }
```

urrentfieldvalue

```
1720 \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

```
1721 \newcommand*{\@glo@types}{,}
```

`\provide@newglossary` If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```
1722 \newcommand*\@gls@provide@newglossary{%
1723   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%

```

Only need to do this once.

```
1724   \let\@gls@provide@newglossary\relax
1725 }
```

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

```
1726 \newcommand*\defglsentryfmt}[2][\glsdefaulttype]{%
1727   \csgdef{gls@#1@entryfmt}{#2}%
1728 }
```

`\gls@doentryfmt`

```
1729 \newcommand*\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}
```

`\ls@forbidtexext` 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.

```
1730 \newcommand*\@gls@forbidtexext}[1]{%
1731   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1732             or test {\ifdefstring{#1}{TEX}}}
1733   {%
1734     \def#1{nottex}%
1735     \PackageError{glossaries}%
1736       {Forbidden '.tex' extension replaced with '.nottex'}%
1737       {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1738        Don't use '.tex' as an extension for a temporary file.}%
1739   }%
1740   {%
1741   }%
1742 }
```

`\gls@gobbleopt` Discard optional argument.

```
1743 \newcommand*\gls@gobbleopt{}\new@ifnextchar[{\@gls@gobbleopt}{}
1744 \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`

```
1745 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}
```

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

```
1746 \newcommand*{\s@newglossary}[2]{%
1747 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1748 }
```

`\ns@newglossary` Define the unstarred version.

```
1749 \newcommand*{\ns@newglossary}[5][glg]{%
1750 \doifglossarynoexistsordo{#2}%
1751 {%
```

Check if default has been set

```
1752 \ifundef\glsdefaultttype
1753 {%
1754 \gdef\glsdefaultttype{#2}%
1755 }{}}%
```

Add this to the list of glossary types:

```
1756 \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.

```
1757 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1758 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1759 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1760 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1761 \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
1762 \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
1763 \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1764 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
1765 \@gls@provide@newglossary
1766 \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.

```
1767 \ifcsundef{gls@#2@entryfmt}%
1768 {%
1769 \defglsentryfmt[#2]{\glsentryfmt}%
1770 }%
1771 {}%
```

Define sort counter if required:

```
1772 \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.)

```
1773 \ifnextchar[{\@gls@setcounter{#2}}{%
1774   {\@gls@setcounter{#2}[\glscounter]}%
1775 }%
1776 {%
1777   \gls@gobbleopt
1778 }%
1779 }
```

`\altnewglossary`

```
1780 \newcommand*{\altnewglossary}[3]{%
1781   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1782 }
```

Only define new glossaries in the preamble:

```
1783 \@onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1784 \@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 \TeX , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```
1785 \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`

```
1786 \def\@gls@setcounter#1[#2]{%
1787   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%
1788   \ifglsxindy
1789     \GlsAddXdyCounters{#2}%
1790   \fi
1791 }
```

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

`@gls@getcounter`

```
1792 \newcommand*{\@gls@getcounter}[1]{%
1793   \csname @glotype@#1@counter\endcsname
1794 }
```


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

```
1795 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1796 \@gls@do@acronymsdef
```

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

```
1797 \@gls@do@symbolsdef
```

```
1798 \@gls@do@numbersdef
```

```
1799 \@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.

```
1800 \newcommand*{\newignoredglossary}[1]{%
1801   \ifdefempty\@ignored@glossaries
1802   {%
1803     \edef\@ignored@glossaries{#1}%
1804   }%
1805   {%
1806     \eappto\@ignored@glossaries{, #1}%
1807   }%
1808   \csgdef{glolist@#1}{,}%
1809   \ifcsundef{gls@#1@entryfmt}%
1810   {%
1811     \defglsentryfmt[#1]{\glsentryfmt}%
1812   }%
1813   {}%
1814   \ifdefempty\@gls@nohyperlist
1815   {%
1816     \renewcommand*{\@gls@nohyperlist}{#1}%
1817   }%
1818   {%
1819     \eappto\@gls@nohyperlist{, #1}%
1820   }%
1821 }
```

`\@ignored@glossaries` List of ignored glossaries.

```
1822 \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.

```
1823 \newcommand*{\ifignoredglossary}[3]{%
1824   \edef\@gls@igtype{#1}%
1825   \expandafter\DTLifinlist\expandafter
1826   {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1827 }
```

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.

```
1828 \define@key{glossentry}{name}{%  
1829 \def\@glo@name{#1}}%  
1830 }
```

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`.

```
1831 \define@key{glossentry}{description}{%  
1832 \def\@glo@desc{#1}}%  
1833 }
```

descriptionplural

```
1834 \define@key{glossentry}{descriptionplural}{%  
1835 \def\@glo@descplural{#1}}%  
1836 }
```

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`.

```
1837 \define@key{glossentry}{sort}{%  
1838 \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.

```
1839 \define@key{glossentry}{text}{%  
1840 \def\@glo@text{#1}}%  
1841 }
```

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.

```
1842 \define@key{glossentry}{plural}{%  
1843 \def\@glo@plural{#1}}%  
1844 }
```

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.

```
1845 \define@key{glossentry}{first}{%
1846 \def\@glo@first{#1}%
1847 }
```

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.

```
1848 \define@key{glossentry}{firstplural}{%
1849 \def\@glo@firstplural{#1}%
1850 }
```

s@default@value

```
1851 \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).

```
1852 \define@key{glossentry}{symbol}{%
1853 \def\@glo@symbol{#1}%
1854 }
```

symbolplural

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

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

```
1858 \define@key{glossentry}{type}{%
1859 \def\@glo@type{#1}}
```

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

```
1860 \define@key{glossentry}{counter}{%
1861 \ifcsundef{c@#1}%
1862 {%
1863 \PackageError{glossaries}%
1864 {There is no counter called ‘#1’}%
1865 {%
1866 The counter key should have the name of a valid counter
1867 as its value%
1868 }%
1869 }%
```

```

1870  {%
1871    \def\@glo@counter{#1}%
1872  }%
1873 }

```

see The see key specifies a list of cross-references

```

1874 \define@key{glossentry}{see}{%
1875   \gls@set@xr@key{see}{\@glo@see}{#1}%
1876 }

```

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

Assign a cross-reference key.

```

1877 \newcommand*{\gls@set@xr@key}[3]{%
1878   \renewcommand*{\gls@xr@key}{#1}%
1879   \gls@checkseeallowed
1880   \def#2{#3}%
1881   \@glo@seeautonumberlist
1882 }

```

`\gls@xr@key`

```

1883 \newcommand*{\gls@xr@key}{see}

```

`checkseeallowed`

```

1884 \newcommand*{\gls@checkseeallowed}{%
1885   \@gls@see@noindex
1886 }

```

`ed@preambleonly`

```

1887 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1888   \GlossariesWarning{glossaries}%
1889   {'\gls@xr@key' key doesn't have any effect when used in the document
1890    environment. Move the definition to the preamble
1891    after \string\makeglossaries\space
1892    or \string\makenoidxglossaries}%
1893 }

```

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

```

1894 \define@key{glossentry}{parent}{%
1895 \def\@glo@parent{#1}}

```

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

```

1896 \define@choicekey{glossentry}{nonumberlist}{%
1897   [\gls@nonumberlist@val\gls@nonumberlist@nr]{true,false}[true]%
1898 }%
1899 \ifcase\gls@nonumberlist@nr\relax
1900   \def\@glo@prefix{\glsnonextpages}%

```

```

1901 \gls@savenonumberlist{true}%
1902 \else
1903 \def\glo@prefix{\glsnextpages}%
1904 \gls@savenonumberlist{false}%
1905 \fi
1906 }

```

`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.

```

1907 \newcommand*{\@gls@savenonumberlist}[1]{%

```

`initnonumberlist`

```

1908 \newcommand*{\@gls@initnonumberlist}{}%

```

`initnonumberlist`

```

1909 \newcommand*{\@gls@storenonumberlist}[1]{%

```

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

```

1910 \newcommand*{\@gls@enablesavenonumberlist}{%
1911 \renewcommand*{\@gls@initnonumberlist}{%
1912 \undef\glo@nonumberlist
1913 }%
1914 \renewcommand*{\@gls@savenonumberlist}[1]{%
1915 \def\glo@nonumberlist{##1}%
1916 }%
1917 \renewcommand*{\@gls@storenonumberlist}[1]{%
1918 \ifdef\glo@nonumberlist
1919 {%
1920 \cslet{glo@glsdetoklabel{##1}@nonumberlist}{\glo@nonumberlist}%
1921 }%
1922 }%
1923 }%
1924 \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1925 }

```

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

`user1`

```

1926 \define@key{glossentry}{user1}{%
1927 \def\glo@useri{#1}%
1928 }

```

`user2`

```

1929 \define@key{glossentry}{user2}{%
1930 \def\glo@userii{#1}%
1931 }

```

user3

```
1932 \define@key{glossentry}{user3}{%
1933   \def\@glo@useriii{#1}%
1934 }
```

user4

```
1935 \define@key{glossentry}{user4}{%
1936   \def\@glo@useriv{#1}%
1937 }
```

user5

```
1938 \define@key{glossentry}{user5}{%
1939   \def\@glo@userv{#1}%
1940 }
```

user6

```
1941 \define@key{glossentry}{user6}{%
1942   \def\@glo@uservi{#1}%
1943 }
```

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.

```
1944 \define@key{glossentry}{short}{%
1945   \def\@glo@short{#1}%
1946 }
```

shortplural This key is provided for use by `\newacronym`.

```
1947 \define@key{glossentry}{shortplural}{%
1948   \def\@glo@shortpl{#1}%
1949 }
```

long This key is provided for use by `\newacronym`.

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

longplural This key is provided for use by `\newacronym`.

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

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

```
1956 \newcommand*{\@glsnname}{%
1957   \PackageError{glossaries}{name key required in
1958     \string\newglossaryentry\space for entry '@glo@label'}{You
1959     haven't specified the entry name}}
```

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

```

1960 \newcommand*\@glsnodelsc{%
1961   \PackageError{glossaries}
1962   {%
1963     description key required in \string\newglossaryentry\space
1964     for entry '\@glo@label'%
1965   }%
1966   {%
1967     You haven't specified the entry description%
1968   }%
1969 }%
```

`lsdefaultplural` Now obsolete. Don't use.

```

1970 \newcommand*\@glsdefaultplural{}
```

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

```

1971 \newcommand*\@gls@missingnumberlist}[1]{%
1972   ??%
1973   \ifglssavenumberlist
1974     \GlossariesWarning{Missing number list for entry '#1'.
1975       Maybe makeglossaries + rerun required}%
1976   \else
1977     \PackageError{glossaries}%
1978     {Package option 'savenumberlist=true' required}%
1979     {%
1980       You must use the 'savenumberlist' package option
1981       to reference location lists.%
1982     }%
1983   \fi
1984 }
```

`@glsdefaultsort` Define command to set default sort.

```

1985 \newcommand*\@glsdefaultsort{\@glo@name}
```

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

```

1986 \newcount\gls@level
```

`@noexpand@field`

```

1987 \newcommand{\@@gls@noexpand@field}[3]{%
1988   \expandafter\global\expandafter
1989   \let\csname glo@#1@#2\endcsname#3%
1990 }
```

`noexpand@fields`

```

1991 \newcommand{\@gls@noexpand@fields}[4]{%
1992   \ifcsdef{gls@assign@#3@field}
1993   {%
1994     \ifdefequal{#4}{\@gls@default@value}%

```

```

1995      {%
1996      \edef\@gls@value{\expandonce{#1}}%
1997      \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1998      }%
1999      {%
2000      \csuse{gls@assign@#3@field}{#2}{#4}%
2001      }%
2002  }%
2003  {%
2004      \ifdefequal{#4}{\@gls@default@value}%
2005      {%
2006      \edef\@gls@value{\expandonce{#1}}%
2007      \@gls@noexpand@field{#2}{#3}{\@gls@value}%
2008      }%
2009      {%
2010      \@gls@noexpand@field{#2}{#3}{#4}%
2011      }%
2012  }%
2013 }

```

ls@expand@field

```

2014 \newcommand{\@gls@expand@field}[3]{%
2015   \expandafter
2016   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
2017 }

```

s@expand@fields

```

2018 \newcommand{\@gls@expand@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       \expandafter\@gls@startswithexpandonce#4\relax\relax\@gls@endcheck
2028     }%
2029     \@gls@expand@field{#2}{#3}{#4}%
2030   }%
2031   {%
2032     \csuse{gls@assign@#3@field}{#2}{#4}%
2033   }%
2034 }%
2035 }%
2036 {%
2037   \ifdefequal{#4}{\@gls@default@value}%
2038   {%

```



```

2039      \@gls@expand@field{#2}{#3}{#1}%
2040    }%
2041    {%
2042      \@gls@expand@field{#2}{#3}{#4}%
2043    }%
2044  }%
2045 }

```

switexpandonce

```

2046 \def\@gls@expandonce{\expandonce}
2047 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
2048   \def\@gls@tmp{#1}%
2049   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
2050 }

```

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.

```
2051 \let\gls@assign@field\@gls@expand@fields
```

glsexpandfields

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

```

2052 \newcommand*{\glsexpandfields}{%
2053   \let\gls@assign@field\@gls@expand@fields
2054 }

```

snoexpandfields

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

```

2055 \newcommand*{\glsnoexpandfields}{%
2056   \let\gls@assign@field\@gls@noexpand@fields
2057 }

```

newglossaryentry

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

```
2058 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```

2059   \glsdoifnoexists{#1}%
2060   {%
2061     \gls@defglossaryentry{#1}{#2}%
2062   }%
2063 }

```

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.

```

2064 \newcommand*{\gls@defdocnewglossaryentry}{%
2065   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
2066   \let\newglossaryentry\new@glossaryentry
2067 }

```

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

```

2068 \newrobustcmd{\provideglossaryentry}[2]{%
2069   \ifglstryexists{#1}%
2070   {}%
2071   {%
2072     \gls@defglossaryentry{#1}{#2}%
2073   }%
2074 }
2075 \@onlypreamble{\provideglossaryentry}

```

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

```

2076 \newrobustcmd{\new@glossaryentry}[2]{%
2077   \ifundef\@gls@deffile
2078   {%
2079     \global\newwrite\@gls@deffile
2080     \immediate\openout\@gls@deffile=\jobname.glsdefs
2081   }%
2082   {}%
2083   \ifglstryexists{#1}{}%
2084   {%
2085     \gls@defglossaryentry{#1}{#2}%
2086   }%
2087   \@gls@writedef{#1}%
2088 }

```

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`.

```

2089 \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.

```

2090 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}

```

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

```

2091 \newcommand*{\gls@begindocdefs}{%
2092   \@gls@enablesavenonumberlist
2093   \edef\@gls@restoreat{\noexpand\catcode'\noexpand\@=\number\catcode'\@relax}%
2094   \makeatletter
2095   \InputIfFileExists{\jobname.glsdefs}{\@gls@restoreat}{\@gls@restoreat}
2096   \@gls@restoreat

```

```

2097 \undef\@gls@restoreat
2098 \gls@defdocnewglossaryentry
2099 }

```

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

```

2100 \newcommand*{\@gls@writedef}[1]{%
2101   \immediate\write\@gls@deffile
2102   {%
2103     \string\ifglsentryexists{#1}{}\glspercentchar^^J%
2104     \expandafter\@gobble\string\{\glspercentchar^^J%
2105     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
2106     \expandafter\@gobble\string\{\glspercentchar%
2107   }%

```

Write key value information:

```

2108   \@for\@gls@map:=\@gls@keymap\do
2109   {%
2110     \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
2111     \ifdef\glo@value
2112     {%
2113       \@onelevel@sanitize\glo@value
2114       \immediate\write\@gls@deffile
2115       {%
2116         \expandafter\@firstoftwo\@gls@map
2117         =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
2118         \glspercentchar
2119       }%
2120     }%
2121   }%
2122 }%

```

Provide hook:

```

2123 \gls@writedefhook
2124 \immediate\write\@gls@deffile
2125 {%
2126   \glspercentchar^^J%
2127   \expandafter\@gobble\string\}\glspercentchar^^J%
2128   \expandafter\@gobble\string\}\glspercentchar%
2129 }%
2130 }

```

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

```

2131 \newcommand*{\@gls@keymap}{%
2132   {name}{name},%
2133   {sort}{sortvalue},% unescaped sort value
2134   {type}{type},%
2135   {first}{first},%
2136   {firstplural}{firstpl},%
2137   {text}{text},%

```

```

2138 {plural}{plural},%
2139 {description}{desc},%
2140 {descriptionplural}{descplural},%
2141 {symbol}{symbol},%
2142 {symbolplural}{symbolplural},%
2143 {user1}{useri},%
2144 {user2}{userii},%
2145 {user3}{useriii},%
2146 {user4}{useriv},%
2147 {user5}{userv},%
2148 {user6}{uservi},%
2149 {long}{long},%
2150 {longplural}{longpl},%
2151 {short}{short},%
2152 {shortplural}{shortpl},%
2153 {counter}{counter},%
2154 {parent}{parent}%
2155 }

```

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

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

```
2156 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
2157 \edef\@gls@thisval{#2}%
```

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

```

2158 \@for\@gls@map:=\@gls@keymap\do{%
2159 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2160 \ifdefequal{\@this@key}{\@gls@thisval}%
2161 {%

```

Found it.

```
2162 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```

2163 \@endfortrue
2164 }%
2165 {}%
2166 }%
2167 }

```

`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.

```
2168 \newcommand*{\glsaddstoragekey}{\@ifstar\sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```

2169 \newcommand*{\@sglsaddstoragekey}[1]{%
2170   \key@ifundefined{glossentry}{#1}%
2171   {%
2172     \expandafter\newcommand\expandafter*\expandafter
2173       {\csname gls@assign@#1@field\endcsname}[2]{%
2174         \@gls@expand@field{##1}{#1}{##2}%
2175       }%
2176   }%
2177   }%
2178   \@glsaddstoragekey{#1}%
2179 }

```

Unstarred version doesn't override default expansion.

```

2180 \newcommand*{\@glsaddstoragekey}[3]{%

```

Check the specified key doesn't already exist.

```

2181   \key@ifundefined{glossentry}{#1}%
2182   {%

```

Set up the key.

```

2183     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2184     \appto\@gls@keymap{,{#1}{#1}}%

```

Set the default value.

```

2185     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

2186     \appto\@newglossaryentryposthook{%
2187       \letcs{\@glo@tmp}{@glo@#1}%
2188       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2189     }%

```

Define the no-link commands.

```

2190     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2191   }%
2192   {%
2193     \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2194   }%
2195 }

```

\glsaddkey	<code>\glsaddkey{<key>}{<default value>}{<no link cs>}{<no link ucfirst cs>}</code> <code>{<link cs>}{<link ucfirst cs>}{<link allcaps cs>}</code>
------------	---

Allow user to add their own custom keys.

```

2196 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}

```

Starred version switches on expansion for this key.

```

2197 \newcommand*{\@sglsaddkey}[1]{%
2198   \key@ifundefined{glossentry}{#1}%

```

```

2199  {%
2200    \expandafter\newcommand\expandafter*\expandafter
2201      {\csname gls@assign@#1@field\endcsname}[2]{%
2202        \@gls@expand@field{##1}{#1}{##2}%
2203      }%
2204  }%
2205  {}%
2206  \@gls@saddkey{#1}%
2207 }

```

Unstarred version doesn't override default expansion.

```

2208 \newcommand*{\@gls@saddkey}[7]{%

```

Check the specified key doesn't already exist.

```

2209   \key@ifundefined{glossentry}{#1}%
2210   {%

```

Set up the key.

```

2211     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2212     \appto\@gls@keymap{, {#1}{#1}}%

```

Set the default value.

```

2213     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

2214     \appto\@newglossaryentryposthook{%
2215       \letcs{\@glo@tmp}{@glo@#1}%
2216       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2217     }%

```

Define the no-link commands.

```

2218     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2219     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2220     \ifcsdef{@gls@user@#1@}%
2221     {%
2222       \PackageError{glossaries}%
2223       {Can't define '\string#5' as helper command
2224       '\expandafter\string\csname @gls@user@#1\endcsname' already exists}%
2225       {}%
2226     }%
2227     {%
2228       \expandafter\newcommand\expandafter*\expandafter
2229       {\csname @gls@user@#1\endcsname}[2][ ]{%
2230         \new@ifnextchar[%
2231           {\csuse{@gls@user@#1@}{##1}{##2}}%
2232           {\csuse{@gls@user@#1@}{##1}{##2} [ ]}%
2233       \csdef{@gls@user@#1@}##1##2[##3]{%
2234         \@gls@field@link{##1}{##2}{#3{##2}##3}%
2235       }%

```

```

2236     \newrobustcmd*{#5}{%
2237     \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2238 }%

```

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

```

2239     \ifcsdef{@Gls@user@#1@}%
2240     {%
2241     \PackageError{glossaries}%
2242     {Can't define '\string#6' as helper command
2243     '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
2244     }%
2245 }%
2246 {%
2247     \expandafter\newcommand\expandafter*\expandafter
2248     {\csname @Gls@user@#1\endcsname}[2][{%
2249     \new@ifnextchar[%
2250     {\csuse{@Gls@user@#1@}{##1}{##2}}%
2251     {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
2252     \csdef{@Gls@user@#1@}##1##2[##3]{%
2253     \@gls@field@link{##1}{##2}{#4{##2}##3}%
2254     }%
2255     \newrobustcmd*{#6}{%
2256     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2257 }%

```

Finally the all caps version:

```

2258     \ifcsdef{@GLS@user@#1@}%
2259     {%
2260     \PackageError{glossaries}%
2261     {Can't define '\string#7' as helper command
2262     '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2263     }%
2264 }%
2265 {%
2266     \expandafter\newcommand\expandafter*\expandafter
2267     {\csname @GLS@user@#1\endcsname}[2][{%
2268     \new@ifnextchar[%
2269     {\csuse{@GLS@user@#1@}{##1}{##2}}%
2270     {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
2271     \csdef{@GLS@user@#1@}##1##2[##3]{%
2272     \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2273     }%
2274     \newrobustcmd*{#7}{%
2275     \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2276 }%
2277 }%
2278 {%
2279     \PackageError{glossaries}{Key '#1' already exists}{}%

```

```

2280 }%
2281 }

```

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

```

2282 \newcommand{\glsfieldxdef}[3]{%
2283   \glsdoifexists{#1}%
2284   {%
2285     \edef\@glo@label{\glsdetoklabel{#1}}%
2286     \ifcsdef{glo@\@glo@label @#2}%
2287     {%
2288       \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2289     }%
2290     {%
2291       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2292     }%
2293   }%
2294 }

```

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

```

2295 \newcommand{\glsfielddedef}[3]{%
2296   \glsdoifexists{#1}%
2297   {%
2298     \edef\@glo@label{\glsdetoklabel{#1}}%
2299     \ifcsdef{glo@\@glo@label @#2}%
2300     {%
2301       \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2302     }%
2303     {%
2304       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2305     }%
2306   }%
2307 }

```

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

```

2308 \newcommand{\glsfieldgdef}[3]{%
2309   \glsdoifexists{#1}%
2310   {%
2311     \edef\@glo@label{\glsdetoklabel{#1}}%
2312     \ifcsdef{glo@\@glo@label @#2}%
2313     {%

```



```

2314     \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2315 }%
2316 {%
2317     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2318 }%
2319 }%
2320 }

```

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

```

2321 \newcommand{\glsfielddef}[3]{%
2322   \glsdoifexists{#1}%
2323   {%
2324     \edef\@glo@label{\glsdetoklabel{#1}}%
2325     \ifcsdef{glo@\@glo@label @#2}%
2326     {%
2327       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2328     }%
2329     {%
2330       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2331     }%
2332   }%
2333 }

```

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

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

```

2334 \newcommand{\glsfieldfetch}[3]{%
2335   \glsdoifexists{#1}%
2336   {%
2337     \edef\@glo@label{\glsdetoklabel{#1}}%
2338     \ifcsdef{glo@\@glo@label @#2}%
2339     {%
2340       \letcs#3{glo@\@glo@label @#2}%
2341     }%
2342     {%
2343       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2344     }%
2345   }%
2346 }

```

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

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

```

2347 \newcommand{\ifglsfieldeq}[5]{%
2348   \glsdoifexists{#1}%
2349   {%
2350     \edef\@glo@label{\glsdetoklabel{#1}}%
2351     \ifcsdef{glo@\@glo@label @#2}%
2352     {%
2353       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2354     }%
2355     {%
2356       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2357     }%
2358   }%
2359 }

```

`\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.

```

2360 \newcommand{\ifglsfieldddefeq}[5]{%
2361   \glsdoifexists{#1}%
2362   {%
2363     \edef\@glo@label{\glsdetoklabel{#1}}%
2364     \ifcsdef{glo@\@glo@label @#2}%
2365     {%
2366       \expandafter\ifdefstrequal
2367       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2368     }%
2369     {%
2370       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2371     }%
2372   }%
2373 }

```

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

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

```

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

```

```
2385 }%
2386 }
```

gls.writedefhook

```
2387 \newcommand*{\gls.writedefhook}{}%
```

gls.assign@desc

```
2388 \newcommand*{\gls.assign@desc}[1]{%
2389   \gls.assign@field{#1}{desc}{\@glo@desc}%
2390   \gls.assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2391 }
```

ewglossaryentry

```
2392 \newcommand{\longnewglossaryentry}[3]{%
2393   \glsdoifnoexists{#1}%
2394   {%
2395     \bgroup
2396     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2397     \long\def\@newglossaryentryprehook{%
2398       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2399       \@org@newglossaryentryprehook
2400     }%
2401     \renewcommand*{\gls.assign@desc}[1]{%
2402       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2403       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2404     }
2405     \gls@defglossaryentry{#1}{#2}%
2406   \egroup
2407 }%
2408 }
```

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

```
2409 \@onlypreamble{\longnewglossaryentry}
```

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

```
2410 \newcommand{\longprovideglossaryentry}[3]{%
2411   \ifglstryexists{#1}{}%
2412   {\longnewglossaryentry{#1}{#2}{#3}}%
2413 }
2414 \@onlypreamble{\longprovideglossaryentry}
```

defglossaryentry

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

Defines a new entry without checking if it already exists.

```
2415 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of \GlsSetQuote:

```
2416 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2417 \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2418 \let\glslabel\@glo@label
```

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

```
2419 \let\@glo@name\@gls@noname
```

```
2420 \let\@glo@desc\@gls@nodesc
```

```
2421 \let\@glo@descplural\@gls@default@value
```

```
2422 \let\@glo@type\@gls@default@value
```

```
2423 \let\@glo@symbol\@gls@default@value
```

```
2424 \let\@glo@symbolplural\@gls@default@value
```

```
2425 \let\@glo@text\@gls@default@value
```

```
2426 \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.)

```
2427 \let\@glo@first\@gls@default@value
```

```
2428 \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2429 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2430 \let\@glo@counter\@gls@default@value
```

```
2431 \def\@glo@see{}%
```

```
2432 \def\@glo@parent{}%
```

```
2433 \def\@glo@prefix{}%
```

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

```
2434 \@gls@initnonnumberlist
```

```
2435 \def\@glo@useri{}%
```

```
2436 \def\@glo@userii{}%
```

```
2437 \def\@glo@useriii{}%
```

```
2438 \def\@glo@useriv{}%
```

```
2439 \def\@glo@userv{}%
```

```
2440 \def\@glo@uservi{}%
```

```

2441 \def\@glo@short{}%
2442 \def\@glo@shortpl{}%
2443 \def\@glo@long{}%
2444 \def\@glo@longpl{}%

```

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

```
2445 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2446 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```

2447 \ifundef\glsdefaulttype
2448 {%
2449   \PackageError{glossaries}%
2450     {No default glossary type (have you used ‘nomain’ by mistake?)}%
2451     {If you use package option ‘nomain’ you must define
2452      a new glossary before you can define entries}%
2453 }%
2454 {}%

```

Assign type. This must be fully expandable

```

2455 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2456 \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.

```

2457 \ifcsundef{glolist@\@glo@type}%
2458 {%
2459   \PackageError{glossaries}%
2460     {Glossary type ‘\@glo@type’ has not been defined}%
2461     {You need to define a new glossary type, before making entries
2462      in it}%
2463 }%
2464 {}%

```

Check if it's an ignored glossary

```

2465 \ifignoredglossary\@glo@type
2466 {%

```

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

```

2467   \ifx\@glo@desc\@glsnodel
2468     \let\@glo@desc\@empty
2469   \fi
2470 }%
2471 {%
2472 }%
2473 \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
2474 \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
2475   \@glolist@{\@glo@label},}%
2476 }%

```

Initialise level to 0.

```

2477 \gls@level=0\relax

```

Has this entry been assigned a parent?

```

2478 \ifx\@glo@parent\@empty

```

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

```

2479 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2480 \else

```

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

```

2481 \ifdefequal\@glo@label\@glo@parent%
2482 {%
2483 \PackageError{glossaries}{Entry '@glo@label' can't be its own parent}{}%
2484 \def\@glo@parent{}%
2485 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2486 }%
2487 {%

```

Check the parent exists:

```

2488 \ifglentryexists{\@glo@parent}%
2489 {%

```

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

```

2490 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2491 \@glo@parent}%

```

Determine level.

```

2492 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2493 \advance\gls@level by 1\relax

```

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

```

2494 \ifx\@glo@name\@gls@name
2495 \expandafter\let\expandafter\@glo@name
2496 \csname glo@\@glo@parent @name\endcsname

```

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

```

2497 \ifx\@glo@plural\@gls@default@value
2498 \expandafter\let\expandafter\@glo@plural
2499 \csname glo@\@glo@parent @plural\endcsname
2500 \fi
2501 \fi
2502 }%
2503 {%

```

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

```

2504 \PackageError{glossaries}%
2505 {%
2506 Invalid parent '@glo@parent'
2507 for entry '@glo@label' - parent doesn't exist%
2508 }%
2509 {%
2510 Parent entries must be defined before their children%

```

```

2511     }%
2512     \def\@glo@parent{}%
2513     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2514 }%
2515 }%
2516 \fi

Set the level for this entry
2517 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

Define commands associated with this entry:
2518 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2519 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2520 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2521 \expandafter\gls@assign@field\expandafter
2522     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2523     {\@glo@label}{plural}{\@glo@plural}%
2524 \expandafter\gls@assign@field\expandafter
2525     {\csname glo@\@glo@label @text\endcsname}%
2526     {\@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.
2527 \ifx\@glo@first\@gls@default@value
2528     \expandafter\gls@assign@field\expandafter
2529         {\csname glo@\@glo@label @plural\endcsname}%
2530         {\@glo@label}{firstpl}{\@glo@firstplural}%
2531 \else
2532     \expandafter\gls@assign@field\expandafter
2533         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2534         {\@glo@label}{firstpl}{\@glo@firstplural}%
2535 \fi

2536 \ifcsundef{@glotype@\@glo@type @counter}%
2537 {%
2538     \def\@glo@defaultcounter{\glscounter}%
2539 }%
2540 {%
2541     \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
2542 }%
2543 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2544 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2545 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2546 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2547 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2548 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2549 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2550 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2551 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2552 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2553 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%

```

```

2554 \ifx\@glo@name\@glsnoname
2555   \@glsnoname
2556   \let\@glo@name\@gls@default@value
2557 \fi
2558 \gls@assign@field{}\@glo@label{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2559 \ifcsundef{glo@\@glo@label @numberlist}%
2560 {%
2561   \csxdef{glo@\@glo@label @numberlist}{%
2562     \noexpand\@gls@missingnumberlist{\@glo@label}}%
2563 }%
2564 {}%

```

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

```

2565 \@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.

```

2566 \def\@glo@@desc{\@glo@first}%
2567 \ifx\@glo@desc\@glo@@desc
2568   \let\@glo@desc\@glo@first
2569 \fi
2570 \ifx\@glo@desc\@gls@nodesc
2571   \@gls@nodesc
2572   \let\@glo@desc\@gls@default@value
2573 \fi
2574 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2575 \@gls@defsort{\@glo@type}{\@glo@label}%

2576 \def\@glo@@symbol{\@glo@text}%
2577 \ifx\@glo@symbol\@glo@@symbol
2578   \let\@glo@symbol\@glo@text
2579 \fi
2580 \gls@assign@field{relax}{\@glo@label}{symbol}{\@glo@symbol}%
2581 \expandafter
2582   \gls@assign@field\expandafter
2583   {\csname glo@\@glo@label @symbol\endcsname}
2584   {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2585 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2586   \noexpand\global
2587   \noexpand\let\expandafter\noexpand
2588   \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2589 }%
2590 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2591   \noexpand\global

```



```

2592      \noexpand\let\expandafter\noexpand
2593      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2594  }%
2595  \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

2596  \@glo@autosee

```

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

```

2597  \ifignoredglossary\@glo@type
2598  {%
2599    \csdef{glo@\@glo@label @index}{}%
2600  }
2601  {%
2602    \do@glo@storeentry{\@glo@label}%
2603  }%

```

Define entry counters if enabled:

```

2604  \@newglossaryentry@defcounters

```

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

```

2605  \@newglossaryentryposthook
2606 }

```

\@glo@autosee Automatically implement \glssee.

```

2607 \newcommand*{\@glo@autosee}{%
2608   \ifdefvoid\@glo@see{}%
2609   {%
2610     \protected@edef\@do@glssee{%
2611       \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2612       \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
2613     \@do@glssee
2614   }%
2615   \@glo@autoseehook
2616 }%

```

glo@autoseehook

```

2617 \newcommand*{\@glo@autoseehook}{}

```

aryentryprehook Allow extra information to be added to glossary entries:

```

2618 \newcommand*{\@newglossaryentryprehook}{}

```

ryentryposthook Allow extra information to be added to glossary entries:

```

2619 \newcommand*{\@newglossaryentryposthook}{}

```

try@defcounters

```

2620 \newcommand*{\@newglossaryentry@defcounters}{}

```

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

```

2621 \newcommand*{\glsmoveentry}[2]{%
2622   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2623   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2624   \def\glo@list{,%}
2625   \forglsentries[\glo@type]{\glo@label}%
2626   {%

2627     \ifdefequal\@glo@thislabel\glo@label
2628       {\eappto\glo@list{\glo@label,%}}%
2629     }%
2630   \cslet\glo@list@\glo@type{\glo@list}%
2631   \csdef{glo@\@glo@thislabel @type}{#2}%
2632 }
```

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

```

2633 \ifglxindy
2634   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2635 \else
2636   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2637 \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.)

```

2638 \ifglxindy
2639   \newcommand*{\@glossarysubentryfield}{%
2640     \string\subglossentry}
2641 \else
2642   \newcommand*{\@glossarysubentryfield}{%
2643     \string\subglossentry}
2644 \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.)

```

2645 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```

2646   \edef\@glo@esclabel{#1}%
2647   \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```

2648 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2649 \@gls@checkmkidxchars\@glo@sort
    Same again for the name string. Escape any special characters in the prefix
2650 \@gls@checkmkidxchars\@glo@prefix
    Get the parent, if one exists
2651 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
    Write the information to the glossary file.
2652 \ifglxsindy
    Store using xindy syntax.
2653 \ifx\@glo@parent\@empty
    Entry doesn't have a parent
2654 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2655 (\string"\@glo@sort\string" %
2656 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2657 }%
2658 \else
    Entry has a parent
2659 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2660 \csname glo@\@glo@parent @index\endcsname
2661 (\string"\@glo@sort\string" %
2662 \string"\@glo@prefix\@glossarysubentryfield
2663 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2664 }%
2665 \fi
2666 \else
    Store using makeindex syntax.
2667 \ifx\@glo@parent\@empty
    Sanitize \@glo@prefix
2668 \@onelevel@sanitize\@glo@prefix
    Entry doesn't have a parent
2669 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2670 \@glo@sort\@gls@actualchar\@glo@prefix
2671 \@glossaryentryfield{\@glo@esclabel}%
2672 }%
2673 \else
    Entry has a parent
2674 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2675 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2676 \@glo@sort\@gls@actualchar\@glo@prefix
2677 \@glossarysubentryfield
2678 {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2679 }%
2680 \fi
2681 \fi
2682 }

```

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`

```
2683 \AtBeginDocument{%
2684   \ifpackageloaded{amsmath}%
2685   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2686   }%
2687 }
2688 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2689   \ifmeasuring@
2690   \else
2691     #1%
2692   \fi
2693 }
2694 \newcommand*\gls@ifnotmeasuring[1]{#1}
```

`lspatchtabularx` 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.

```
2695 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2696   \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2697 }
2698 \newcommand*\glspatchtabularx{%
2699   \ifdef\TX@trial
2700   {%
2701     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2702     \let\glspatchtabularx\relax
2703   }%
2704   {%
2705 }
```

`\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.

```
2706 \newcommand*{\glsreset}[1]{%
2707   \gls@ifnotmeasuring
2708   {%
2709     \glsdoifexists{#1}%
2710     {%
2711       \@glsreset{#1}%
2712     }%
2713   }%
2714 }
```

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

```

2715 \newcommand*{\glslocalreset}[1]{%
2716   \gls@ifnotmeasuring
2717   {%
2718     \glsdoifexists{#1}%
2719     {%
2720       \@glslocalreset{#1}%
2721     }%
2722   }%
2723 }

```

`\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.

```

2724 \newcommand*{\glsunset}[1]{%
2725   \gls@ifnotmeasuring
2726   {%
2727     \glsdoifexists{#1}%
2728     {%
2729       \@glsunset{#1}%
2730     }%
2731   }%
2732 }

```

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

```

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

```

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

```

2742 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2743 \newcommand*{\@glslocalunset}[1]{%
2744   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2745 }

```

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

```

2746 \newcommand*{\@glsunset}{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2747 \newcommand*{\@@glsunset}[1]{%
2748   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2749 }

```

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

```
2750 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`\@@glslocalreset` Local reset without checks.

```
2751 \newcommand*{\@@glslocalreset}[1]{%
2752   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2753 }
```

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

```
2754 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2755 \newcommand*{\@@glsreset}[1]{%
2756   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2757 }
```

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

`\glsresetall`

```
2758 \newcommand*{\glsresetall}[1][\@glo@types]{%
2759   \forallglsentries[#1]{\@glsentry}%
2760   {%
2761     \glsreset{\@glsentry}%
2762   }%
2763 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2764 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2765   \forallglsentries[#1]{\@glsentry}%
2766   {%
2767     \glslocalreset{\@glsentry}%
2768   }%
2769 }
```

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

`\glsunsetall`

```
2770 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2771   \forallglsentries[#1]{\@glsentry}%
2772   {%
2773     \glsunset{\@glsentry}%
2774   }%
2775 }
```

As above, but with only a local effect:

lslocalunsetall

```

2776 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2777   \forallglsentries[#1]{\@glsentry}%
2778   {%
2779     \glslocalunset{\@glsentry}%
2780   }%
2781 }

```

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.

```

2782 \newcommand*{\@newglossaryentry@defcounters}{%
2783   \csdef{glo@\@glo@label @currcount}{0}%
2784   \csdef{glo@\@glo@label @prevcount}{0}%
2785 }

```

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

```

2786 \newcommand*{\glsenableentrycount}{%

```

Enable new entry fields.

```

2787 \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters

```

Disable `\newglossaryentry` in the document environment.

```

2788 \renewcommand*{\gls@defdocnewglossaryentry}{%
2789   \renewcommand*{\newglossaryentry}[2]{%
2790     \PackageError{glossaries}{\string\newglossaryentry\space
2791       may only be used in the preamble when entry counting has
2792       been activated}{If you use \string\glsenableentrycount\space
2793       you must place all entry definitions in the preamble not in
2794       the document environment}%
2795   }%
2796 }%

```

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

```

2797 \newcommand*{\glsentrycurrcount}[1]{%
2798   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2799     {0}{\@gls@entry@field{##1}{currcount}}%
2800   }%
2801 \newcommand*{\glsentryprevcount}[1]{%

```

```

2802 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2803 {0}{\@gls@entry@field{##1}{prevcount}}}%
2804 }%

```

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

```

2805 \renewcommand*{\@glsunset}[1]{%
2806   \@glsunset{##1}%
2807   \@gls@increment@currcount{##1}%
2808 }%
2809 \renewcommand*{\@glslocalunset}[1]{%
2810   \@glslocalunset{##1}%
2811   \@gls@local@increment@currcount{##1}%
2812 }%
2813 \renewcommand*{\@glsreset}[1]{%
2814   \@glsreset{##1}%
2815   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2816 }%
2817 \renewcommand*{\@glslocalreset}[1]{%
2818   \@glslocalreset{##1}%
2819   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2820 }%

```

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.)

```

2821 \def\@cgl's@##1##2[##3]{%
2822   \ifnum\glsentryprevcount{##2}=1\relax
2823     \cgl'sformat{##2}{##3}%
2824     \glsunset{##2}%
2825   \else
2826     \@gls@{##1}{##2}[##3]%
2827   \fi
2828 }%

```

Similarly for the analogous commands. No case change plural:

```

2829 \def\@cgl'spl@##1##2[##3]{%
2830   \ifnum\glsentryprevcount{##2}=1\relax
2831     \cgl'splformat{##2}{##3}%
2832     \glsunset{##2}%
2833   \else
2834     \@cgl'spl@{##1}{##2}[##3]%
2835   \fi
2836 }%

```

First letter uppercase singular:

```

2837 \def\@cGls@##1##2[##3]{%
2838   \ifnum\glsentryprevcount{##2}=1\relax
2839     \cGlsformat{##2}{##3}%
2840     \glsunset{##2}%
2841   \else
2842     \@Gls@{##1}{##2}[##3]%
2843   \fi

```


2844 }%

First letter uppercase plural:

```
2845 \def\@cGlspl@##1##2[##3]{%
2846   \ifnum\glsentryprevcount{##2}=1\relax
2847     \cGlsplformat{##2}{##3}%
2848     \glsunset{##2}%
2849   \else
2850     \@Glspl@{##1}{##2}[##3]%
2851   \fi
2852 }%
```

Write information to aux file at the end of the document

```
2853 \AtEndDocument{\@gls@write@entrycounts}%
```

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

```
2854 \renewcommand*{\@gls@entry@count}[2]{%
2855   \csxdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
2856 }%
```

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

```
2857 \let\glsenableentrycount\relax
2858 }
2859 \@onlypreamble\glsenableentrycount
```

ement@currcount

```
2860 \newcommand*{\@gls@increment@currcount}[1]{%
2861   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2862     \number\numexpr\glsentrycurrcount{##1}+1}%
2863 }
```

ement@currcount

```
2864 \newcommand*{\@gls@local@increment@currcount}[1]{%
2865   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2866     \number\numexpr\glsentrycurrcount{##1}+1}%
2867 }
```

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.)

```
2868 \newcommand*{\@gls@write@entrycounts}{%
2869   \immediate\write\@auxout
2870     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2871   \forallglsentries{\@glsentry}{%
2872     \ifglsused{\@glsentry}%
2873     {\immediate\write\@auxout
2874       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2875     }%
2876 }
```

```
2876 }%
2877 }
```

`\gls@entry@count` Default behaviour is to ignore arguments. Activated by `\glsenableentrycount`.

```
2878 \newcommand*{\@gls@entry@count}[2]{}
```

`\cgl` Define command that works like `\gls` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\gls` but issues a warning.)

```
2879 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}
```

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

```
2880 \newcommand*{\@cgl}[2][{}]{%
2881 \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[]}]%
2882 }
```

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

```
2883 \def\@cgl@#1#2[#3]{%
2884 \GlossariesWarning{\string\cgl\space is defaulting to
2885 \string\gls\space since you haven't enabled entry counting}%
2886 \@gls@{#1}{#2}[#3]%
2887 }
```

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

```
2888 \newcommand*{\cglformat}[2][{}]{%
2889 \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2890 }
```

`\cGl` Define command that works like `\Gls` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\Gls` but issues a warning.)

```
2891 \newrobustcmd*{\cGl}{\@gls@hyp@opt\@cGl}
```

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

```
2892 \newcommand*{\@cGl}[2][{}]{%
2893 \new@ifnextchar[{\@cGl@{#1}{#2}}{\@cGl@{#1}{#2}[]}]%
2894 }
```

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

```
2895 \def\@cGl@#1#2[#3]{%
2896 \GlossariesWarning{\string\cGl\space is defaulting to
2897 \string\Gls\space since you haven't enabled entry counting}%
2898 \@Gls@{#1}{#2}[#3]%
2899 }
```

`\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.

```
2900 \newcommand*{\cGlsformat}[2]{%
2901   \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2902 }
```

`\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.)

```
2903 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}
```

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

```
2904 \newcommand*{\@cglsp1}[2][{}]{%
2905   \new@ifnextchar[\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2}[{}]}%
2906 }
```

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

```
2907 \def\@cglsp1@#1#2[#3]{%
2908   \GlossariesWarning{\string\cglsp1\space is defaulting to
2909     \string\glsp1\space since you haven't enabled entry counting}%
2910   \@glsp1@{#1}{#2}[#3]%
2911 }
```

`\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.

```
2912 \newcommand*{\cglsp1format}[2]{%
2913   \ifglshaslong{#1}{\glsp1long{#1}}{\glsp1firstplural{#1}}#2%
2914 }
```

`\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.)

```
2915 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}
```

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

```
2916 \newcommand*{\@cGlspl}[2][{}]{%
2917   \new@ifnextchar[\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[{}]}%
2918 }
```

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

```
2919 \def\@cGlspl@#1#2[#3]{%
2920   \GlossariesWarning{\string\cGlspl\space is defaulting to
2921     \string\Glspl\space since you haven't enabled entry counting}%
2922   \@Glspl@{#1}{#2}[#3]%
2923 }
```

`\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.

```
2924 \newcommand*{\cGlsplformat}[2]{%
2925   \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2926 }
```

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`

```
2927 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2928   \let\@gls@default\glsdefaulttype
2929   \def\glsdefaulttype{#1}\input{#2}%
2930   \let\glsdefaulttype\@gls@default
2931 }
```

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

```
2932 \@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`

```
2933 \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

¹and 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

```
2934 \newcommand*{\glsentryfmt}{%
2935   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2936 }
```

Format that provides backwards compatibility:

```
2937 \newcommand*{\@gls@default@entryfmt}[2]{%
2938   \ifdefempty\glscustomtext
2939   {%
2940     \glsifplural
2941     {%
```

Plural form

```
2942     \glscapscase
2943     {%
```

Don't adjust case

```
2944     \ifglsused\glslabel
2945     {%
```

Subsequent use

```
2946         #2{\glsentryplural{\glslabel}}%
2947         {\glsentrydescplural{\glslabel}}%
2948         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2949     }%
2950     {%
```

First use

```
2951         #1{\glsentryfirstplural{\glslabel}}%
2952         {\glsentrydescplural{\glslabel}}%
2953         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2954     }%
2955     }%
2956     {%
```

Make first letter upper case

```
2957     \ifglsused\glslabel
2958     {%
```

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.)

```
2959     \ifbool{glscompatible-3.07}%
2960     {%
2961       \protected@edef\@glo@etext{%
2962         #2{\glsentryplural{\glslabel}}%
2963         {\glsentrydescplural{\glslabel}}%
2964         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2965       \xmakefirstuc\@glo@etext
2966     }%
```

```

2967      {%
2968      #2{\Glsentryplural{\glslabel}}%
2969      {\glsentrydescplural{\glslabel}}%
2970      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2971      }%
2972      }%
2973      {%

```

First use

```

2974      \ifbool{glscompatible-3.07}%
2975      {%
2976      \protected@edef\@glo@etext{%
2977      #1{\glsentryfirstplural{\glslabel}}%
2978      {\glsentrydescplural{\glslabel}}%
2979      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2980      \xmakefirstuc\@glo@etext
2981      }%
2982      {%
2983      #1{\Glsentryfirstplural{\glslabel}}%
2984      {\glsentrydescplural{\glslabel}}%
2985      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2986      }%
2987      }%
2988      }%
2989      {%

```

Make all upper case

```

2990      \ifglsused\glslabel
2991      {%

```

Subsequent use

```

2992      \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2993      {\glsentrydescplural{\glslabel}}%
2994      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2995      }%
2996      {%

```

First use

```

2997      \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2998      {\glsentrydescplural{\glslabel}}%
2999      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
3000      }%
3001      }%
3002      }%
3003      {%

```

Singular form

```

3004      \glscapscase
3005      {%

```

Don't adjust case

```

3006      \ifglsused\glslabel

```

```

3007      {%
Subsequent use
3008      #2{\glsentrytext{\glslabel}}%
3009      {\glsentrydesc{\glslabel}}%
3010      {\glsentrysymbol{\glslabel}}{\glsinsert}%
3011      }%
3012      {%

First use
3013      #1{\glsentryfirst{\glslabel}}%
3014      {\glsentrydesc{\glslabel}}%
3015      {\glsentrysymbol{\glslabel}}{\glsinsert}%
3016      }%
3017      }%
3018      {%

Make first letter upper case
3019      \ifglused\glslabel
3020      {%

Subsequent use
3021      \ifbool{glscompatible-3.07}%
3022      {%
3023      \protected@edef\@glo@etext{%
3024      #2{\glsentrytext{\glslabel}}%
3025      {\glsentrydesc{\glslabel}}%
3026      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3027      \xmakefirstuc\@glo@etext
3028      }%
3029      {%
3030      #2{\Glsentrytext{\glslabel}}%
3031      {\glsentrydesc{\glslabel}}%
3032      {\glsentrysymbol{\glslabel}}{\glsinsert}%
3033      }%
3034      }%
3035      {%

First use
3036      \ifbool{glscompatible-3.07}%
3037      {%
3038      \protected@edef\@glo@etext{%
3039      #1{\glsentryfirst{\glslabel}}%
3040      {\glsentrydesc{\glslabel}}%
3041      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
3042      \xmakefirstuc\@glo@etext
3043      }%
3044      {%
3045      #1{\Glsentryfirst{\glslabel}}%
3046      {\glsentrydesc{\glslabel}}%
3047      {\glsentrysymbol{\glslabel}}{\glsinsert}%

```

```

3048         }%
3049     }%
3050 }%
3051 {%

    Make all upper case
3052     \ifglsused\glslabel
3053     {%

        Subsequent use
3054         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
3055         {\glsentrydesc{\glslabel}}}%
3056         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
3057     }%
3058     {%

        First use
3059         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
3060         {\glsentrydesc{\glslabel}}}%
3061         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
3062     }%
3063 }%
3064 }%
3065 }%
3066 {%

    Custom text provided in \glsdisp
3067     \ifglsused{\glslabel}%
3068     {%

        Subsequent use
3069         #2{\glscustomtext}%
3070         {\glsentrydesc{\glslabel}}}%
3071         {\glsentrysymbol{\glslabel}}{}%
3072     }%
3073     {%

        First use
3074         #1{\glscustomtext}%
3075         {\glsentrydesc{\glslabel}}}%
3076         {\glsentrysymbol{\glslabel}}{}%
3077     }%
3078 }%
3079 }

```

`\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.

```

3080 \newcommand*{\glsgenentryfmt}{%
3081     \ifdefempty\glscustomtext
3082     {%
3083         \glsifplural
3084         {%

```


Plural form

```
3085      \glscapscase
3086      {%
```

Don't adjust case

```
3087      \ifglused\glslabel
3088      {%
```

Subsequent use

```
3089      \glentryplural{\glslabel}\glinsert
3090      }%
3091      {%
```

First use

```
3092      \glentryfirstplural{\glslabel}\glinsert
3093      }%
3094      }%
3095      {%
```

Make first letter upper case

```
3096      \ifglused\glslabel
3097      {%
```

Subsequent use.

```
3098      \Glsentryplural{\glslabel}\glinsert
3099      }%
3100      {%
```

First use

```
3101      \Glsentryfirstplural{\glslabel}\glinsert
3102      }%
3103      }%
3104      {%
```

Make all upper case

```
3105      \ifglused\glslabel
3106      {%
```

Subsequent use

```
3107      \mfirstucMakeUppercase
3108      {\glentryplural{\glslabel}\glinsert}%
3109      }%
3110      {%
```

First use

```
3111      \mfirstucMakeUppercase
3112      {\glentryfirstplural{\glslabel}\glinsert}%
3113      }%
3114      }%
3115      }%
3116      {%
```

Singular form

```
3117 \glscapscase
3118 {%
```

Don't adjust case

```
3119 \ifglused\glslabel
3120 {%
```

Subsequent use

```
3121 \glentrytext{\glslabel}\glsinsert
3122 }%
3123 {%
```

First use

```
3124 \glentryfirst{\glslabel}\glsinsert
3125 }%
3126 }%
3127 {%
```

Make first letter upper case

```
3128 \ifglused\glslabel
3129 {%
```

Subsequent use

```
3130 \Glentrytext{\glslabel}\glsinsert
3131 }%
3132 {%
```

First use

```
3133 \Glentryfirst{\glslabel}\glsinsert
3134 }%
3135 }%
3136 {%
```

Make all upper case

```
3137 \ifglused\glslabel
3138 {%
```

Subsequent use

```
3139 \mfirstucMakeUppercase{\glentrytext{\glslabel}\glsinsert}%
3140 }%
3141 {%
```

First use

```
3142 \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glsinsert}%
3143 }%
3144 }%
3145 }%
3146 }%
3147 {%
```

Custom text provided in \glldisp. (The insert is most likely to be empty at this point.)

```
3148 \glscustomtext\glsinsert
```

```

3149 }%
3150 }

```

`\glsgenacfmt` Define a generic acronym format that uses the long and short keys (or their plurals) and `\acrfullformat`, `\firstacronymfont` and `\acronymfont`.

```

3151 \newcommand*{\glsgenacfmt}{%
3152   \ifdefempty\glscustomtext
3153   {%
3154     \ifglused\glslabel
3155     {%

```

Subsequent use:

```

3156     \glsifplural
3157     {%

```

Subsequent plural form:

```

3158     \glscapscase
3159     {%

```

Subsequent plural form, don't adjust case:

```

3160     \acronymfont{\glentryshortpl{\glslabel}}\glsinsert
3161     }%
3162     {%

```

Subsequent plural form, make first letter upper case:

```

3163     \acronymfont{\Glentryshortpl{\glslabel}}\glsinsert
3164     }%
3165     {%

```

Subsequent plural form, all caps:

```

3166     \mfirstucMakeUppercase
3167     {\acronymfont{\glentryshortpl{\glslabel}}\glsinsert}%
3168     }%
3169     }%
3170     {%

```

Subsequent singular form

```

3171     \glscapscase
3172     {%

```

Subsequent singular form, don't adjust case:

```

3173     \acronymfont{\glentryshort{\glslabel}}\glsinsert
3174     }%
3175     {%

```

Subsequent singular form, make first letter upper case:

```

3176     \acronymfont{\Glentryshort{\glslabel}}\glsinsert
3177     }%
3178     {%

```

Subsequent singular form, all caps:

```

3179     \mfirstucMakeUppercase
3180     {\acronymfont{\glentryshort{\glslabel}}\glsinsert}%

```

```

3181      }%
3182      }%
3183      }%
3184      {%

```

First use:

```

3185      \glsifplural
3186      {%

```

First use plural form:

```

3187      \glscapscase
3188      {%

```

First use plural form, don't adjust case:

```

3189      \genplacrformat{\glslabel}{\glsinsert}%
3190      }%
3191      {%

```

First use plural form, make first letter upper case:

```

3192      \Genplacrformat{\glslabel}{\glsinsert}%
3193      }%
3194      {%

```

First use plural form, all caps:

```

3195      \mfirstucMakeUppercase
3196      {\genplacrformat{\glslabel}{\glsinsert}}%
3197      }%
3198      }%
3199      {%

```

First use singular form

```

3200      \glscapscase
3201      {%

```

First use singular form, don't adjust case:

```

3202      \genacrformat{\glslabel}{\glsinsert}%
3203      }%
3204      {%

```

First use singular form, make first letter upper case:

```

3205      \Genacrformat{\glslabel}{\glsinsert}%
3206      }%
3207      {%

```

First use singular form, all caps:

```

3208      \mfirstucMakeUppercase
3209      {\genacrformat{\glslabel}{\glsinsert}}%
3210      }%
3211      }%
3212      }%
3213      }%
3214      {%

```

User supplied text.

```
3215 \glscustomtext
3216 }%
3217 }
```

genacrfullformat `\genacrfullformat{<label>}{<insert>}`

The full format used by `\glsgenacfmt` (singular).

```
3218 \newcommand*{\genacrfullformat}[2]{%
3219 \glentrylong{#1}#2\space
3220 (\protect\firstacronymfont{\glentryshort{#1}})%
3221 }
```

Genacrfullformat `\Genacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```
3222 \newcommand*{\Genacrfullformat}[2]{%
3223 \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3224 \xmakefirstuc\gls@text
3225 }
```

nplacrfullformat `\genplacrfullformat{<label>}{<insert>}`

The full format used by `\glsgenacfmt` (plural).

```
3226 \newcommand*{\genplacrfullformat}[2]{%
3227 \glentrylongpl{#1}#2\space
3228 (\protect\firstacronymfont{\glentryshortpl{#1}})%
3229 }
```

nplacrfullformat `\Genplacrfullformat{<label>}{<insert>}`

As above but makes the first letter upper case.

```
3230 \newcommand*{\Genplacrfullformat}[2]{%
3231 \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3232 \xmakefirstuc\gls@text
3233 }
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
3234 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3235 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3236 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3237   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3238   Use \string\defglsentryfmt\space instead}%
3239   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3240   \edef\@gls@doentrydef{%
3241     \noexpand\defglsentryfmt[#1]{%
3242       \noexpand\ifcsdef{gls@#1@displayfirst}%
3243       {%
3244         \noexpand\@@gls@default@entryfmt
3245         {\noexpand\csuse{gls@#1@displayfirst}}}%
3246         {\noexpand\csuse{gls@#1@display}}}%
3247       }%
3248       {%
3249         \noexpand\@@gls@default@entryfmt
3250         {\noexpand\glsdisplayfirst}%
3251         {\noexpand\csuse{gls@#1@display}}}%
3252       }%
3253     }%
3254   }%
3255   \@gls@doentrydef
3256 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3257 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3258   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3259   Use \string\defglsentryfmt\space instead}%
3260   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3261   \edef\@gls@doentrydef{%
3262     \noexpand\defglsentryfmt[#1]{%
3263       \noexpand\ifcsdef{gls@#1@display}%
3264       {%
3265         \noexpand\@@gls@default@entryfmt
3266         {\noexpand\csuse{gls@#1@displayfirst}}}%
3267         {\noexpand\csuse{gls@#1@display}}}%
3268       }%
3269       {%
3270         \noexpand\@@gls@default@entryfmt
3271         {\noexpand\csuse{gls@#1@displayfirst}}}%
3272         {\noexpand\glsdisplay}%
3273       }%
3274     }%
3275   }%
3276   \@gls@doentrydef
3277 }
```

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.

```
3278 \define@key{glslink}{counter}{%
3279   \ifcsundef{c@#1}%
3280   {%
3281     \PackageError{glossaries}%
3282     {There is no counter called '#1'}%
3283     {%
3284       The counter key should have the name of a valid counter
3285       as its value%
3286     }%
3287   }%
3288   {%
3289     \def\@gls@counter{#1}%
3290   }%
3291 }
```

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.

```
3292 \define@key{glslink}{format}{%
3293   \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.

```
3294 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3295 \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.

```
3296 \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.

```
3297 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3298 \newcommand*{\glsifhyper}[2]{%
3299 \glslinkvar{#1}{#2}{#1}%
3300 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3301 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3302 }
```

`\@gls@hyp@opt` Used by the commands such as `\glslink` to determine whether to modify the hyper option.

```
3303 \newcommand*{\@gls@hyp@opt}[1]{%
3304 \let\glslinkvar\@firstofthree
3305 \let\@gls@hyp@opt@cs#1\relax
3306 \@ifstar{\s@gls@hyp@opt}%
3307 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3308 }
```

`\s@gls@hyp@opt` Starred version

```
3309 \newcommand*{\s@gls@hyp@opt}[1] []{%
3310 \let\glslinkvar\@secondofthree
3311 \@gls@hyp@opt@cs[hyper=false,#1]}
```

`\p@gls@hyp@opt` Plus version

```
3312 \newcommand*{\p@gls@hyp@opt}[1] []{%
3313 \let\glslinkvar\@thirdofthree
3314 \@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

```
3315 \newrobustcmd*{\glslink}{%  
3316 \@gls@hyp@opt\@gls@@link  
3317 }
```

\@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.

```
3318 \newcommand*{\@gls@@link}[3][\relax]{%  
3319 \glsdoifexistsordo{#2}%  
3320 {%  
3321 \let\do@gls@link@checkfirsthyper\relax  
3322 \@gls@link[#1]{#2}{#3}%  
3323 }%
```

Display the specified text. (The entry doesn't exist so there's nothing to link it to.)

```
3324 \glstextformat{#3}%  
3325 }%  
  
3326 \glspostlinkhook  
3327 }
```

glspostlinkhook

```
3328 \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.

```
3329 \newcommand*{\@gls@link@checkfirsthyper}{%  
3330 \ifglsused{\glslabel}%  
3331 {%  
3332 }%  
3333 {%  
3334 \gls@checkisacronymlist\glstype  
3335 \ifglshyperfirst  
3336 \if@glsisacronymlist  
3337 \ifglsacrfootnote  
3338 \KV@glslink@hyperfalse  
3339 \fi  
3340 \fi  
3341 \else  
3342 \KV@glslink@hyperfalse  
3343 \fi  
3344 }%
```

Allow user to hook into this

```
3345 \glslinkcheckfirsthyperhook  
3346 }
```

`checkfirsthyperhook` Allow used to hook into the `\@gls@link@checkfirsthyper` macro
3347 `\newcommand*{\glslinkcheckfirsthyperhook}{}`

`linkpostsetkeys`
3348 `\newcommand*{\glslinkpostsetkeys}{}`

`\glsifhyperon` Check the value of the hyper key:
3349 `\newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}`

`disablehyperinlist` Disable hyperlink if in the “nohyper” list.
3350 `\newcommand*{\do@glssdisablehyperinlist}{%`
3351 `\expandafter\DTLifinlist\expandafter{\gls@type}{\@gls@nohyperlist}%`
3352 `{\KV@glslink@hyperfalse}{}}%`
3353 `}`

`\let@glslink@opts` Hook to set default options for `\@glslink`.
3354 `\newcommand*{\@gls@setdefault@glslink@opts}{}`

`\@gls@link`
3355 `\def\@gls@link[#1]#2#3{%`
Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with tabularx).
3356 `\leavevmode`
3357 `\edef\glslabel{\glsdetoklabel{#2}}%`
Save options in `\@gls@link@opts` and label in `\@gls@link@label`
3358 `\def\@gls@link@opts{#1}%`
3359 `\let\@gls@link@label\glslabel`
3360 `\def\@glsnumberformat{glsnumberformat}%`
3361 `\edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%`
If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default
3362 `\edef\gls@type{\csname glo@\glslabel @type\endcsname}%`
Save original setting
3363 `\let\org@ifKV@glslink@hyper\ifKV@glslink@hyper`
Set defaults:
3364 `\@gls@setdefault@glslink@opts`
Switch off hyper setting if the glossary type has been identified in nohyperlist.
3365 `\do@glssdisablehyperinlist`
Macros must set this before calling `\@gls@link`. The commands that check the first use flag should set this to `\@gls@link@checkfirsthyper` otherwise it should be set to `\relax`.
3366 `\do@gls@link@checkfirsthyper`
3367 `\setkeys{glslink}{#1}%`
Add a hook for the user to customise things after the keys have been set.
3368 `\glslinkpostsetkeys`

```

    Store the entry's counter in \theglsentrycounter
3369   \@gls@saveentrycounter

    Define sort key if necessary:
3370   \@gls@setsort{\glslabel}%

    (De-tok'ing done by \@do@wrglossary)
3371   \@do@wrglossary{#2}%
3372   \ifKV@glslink@hyper
3373     \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3374   \else

3375     \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3376   \fi

    Restore original setting
3377   \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3378 }

\glolinkprefix
3379 \newcommand*{\glolinkprefix}{glo:}

glsentrycounter  Set default value of entry counter
3380 \def\glsentrycounter{\glscounter}%

saveentrycounter  Need to check if using equation counter in align environment:
3381 \newcommand*{\@gls@saveentrycounter}{%
3382   \def\@gls@Hcounter{}}%

    Are we using equation counter?
3383   \ifthenelse{\equal{\@gls@counter}{equation}}{%
3384     {

    If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as
    may be inside an inner environment.)
3385     \ifcsundef{xatlevel@}%
3386     {%
3387       \edef\theglsentrycounter{\expandafter\noexpand
3388         \csname the\@gls@counter\endcsname}%
3389     }%
3390     {%
3391       \ifx\xatlevel@\@empty
3392         \edef\theglsentrycounter{\expandafter\noexpand
3393           \csname the\@gls@counter\endcsname}%
3394       \else
3395         \savecounters@
3396         \advance\c@equation by 1\relax
3397         \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3398 \ifcsundef{theH\@gls@counter}%
3399 {%
3400 \def\@gls@Hcounter{\theglsentrycounter}%
3401 }%
3402 {%
3403 \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3404 }%
3405 \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3406 \restorecounters@
3407 \fi
3408 }%
3409 }%
3410 {%

```

Not using equation counter so no special measures:

```

3411 \edef\theglsentrycounter{\expandafter\noexpand
3412 \csname the\@gls@counter\endcsname}%
3413 }%

```

Check if hyperref version of this counter

```

3414 \ifx\@gls@Hcounter\@empty
3415 \ifcsundef{theH\@gls@counter}%
3416 {%
3417 \def\theHglentrycounter{\theglsentrycounter}%
3418 }%
3419 {%
3420 \protected@edef\theHglentrycounter{\expandafter\noexpand
3421 \csname theH\@gls@counter\endcsname}%
3422 }%
3423 \fi
3424 }

```

`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.

```

3425 \def\@set@glo@numformat#1#2#3#4{%
3426 \expandafter\@glo@check@mkidxrangechar#3\@nil
3427 \protected@edef#1{%
3428 \@glo@prefix setentrycounter[#4]{#2}%
3429 \expandafter\string\csname\@glo@suffix\endcsname
3430 }%
3431 \@gls@checkmkidxchars#1%
3432 }

```

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.

```

3433 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3434 \if#1(\relax
3435   \def\@glo@prefix{()%
3436   \if\relax#2\relax
3437     \def\@glo@suffix{glsnumberformat}%
3438   \else
3439     \def\@glo@suffix{#2}%
3440   \fi
3441 \else
3442   \if#1)\relax
3443     \def\@glo@prefix{}}}%
3444   \if\relax#2\relax
3445     \def\@glo@suffix{glsnumberformat}%
3446   \else
3447     \def\@glo@suffix{#2}%
3448   \fi
3449 \else
3450   \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3451 \fi
3452 \fi}

```

`\@gls@escbsdq` Escape backslashes and double quote marks. The argument must be a control sequence.

```

3453 \newcommand*{\@gls@escbsdq}[1]{%
3454   \def\@gls@checkedmkidx{%
3455     \let\gls@xdystring=#1\relax
3456     \@onelevel@sanitize\gls@xdystring
3457     \edef\do@gls@xdycheckbackslash{%
3458       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3459       \@backslashchar\@backslashchar\noexpand\null}%
3460     \do@gls@xdycheckbackslash
3461     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3462     \def\@gls@checkedmkidx{%
3463       \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3464       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

Unsanitize\gls@numberpage,\gls@alphpage,\gls@Alphpage and\gls@romanpage (thanks to David Carlisle for the suggestion.)

```

3465   \@for\@gls@tmp:=\gls@protected@pagefmts\do
3466   {%
3467     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3468     \@onelevel@sanitize\@gls@sanitized@tmp
3469     \edef\gls@dosubst{%
3470       \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3471       {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3472     }%
3473     \gls@dosubst
3474   }%

```

Assign to required control sequence

```

3475   \let#1=\gls@xdystring

```

3476 }

Catch special characters (argument must be a control sequence):

checkmkidxchars

```
3477 \newcommand{\@gls@checkmkidxchars}[1]{%
3478   \ifglxsindy
3479     \@gls@escbsdq{#1}%
3480   \else
3481     \def\@gls@checkedmkidx{%
3482       \expandafter\@gls@checkquote#1\@nil""\null
3483       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3484     \def\@gls@checkedmkidx{%
3485       \expandafter\@gls@checkescquote#1\@nil\""\null
3486       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3487     \def\@gls@checkedmkidx{%
3488       \expandafter\@gls@checkescactual#1\@nil"??\null
3489       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3490     \def\@gls@checkedmkidx{%
3491       \expandafter\@gls@checkactual#1\@nil??\null
3492       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3493     \def\@gls@checkedmkidx{%
3494       \expandafter\@gls@checkbar#1\@nil||\null
3495       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3496     \def\@gls@checkedmkidx{%
3497       \expandafter\@gls@checkescbar#1\@nil\\|\null
3498       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3499     \def\@gls@checkedmkidx{%
3500       \expandafter\@gls@checklevel#1\@nil!!\null
3501       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3502     \fi
3503 }
```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```
3504 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
3505 \newtoks\@gls@tmpb
```

@gls@checkquote Replace " with "" since " is a makeindex special character.

```
3506 \def\@gls@checkquote#1"#2"#3\@nil{%
3507   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3508   \toks@={#1}%
3509   \ifx\@nil#2\@nil
3510     \ifx\@nil#3\@nil
3511       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3512     \def\@gls@checkquote{\relax}%
3513   \else
```

```

3514 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3515 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3516 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
3517 \fi
3518 \else
3519 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3520 \@gls@quotechar\@gls@quotechar}%
3521 \ifx\null#3\null
3522 \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
3523 \else
3524 \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
3525 \fi
3526 \fi
3527 \@@gls@checkquote
3528 }

```

s@checkescquote Do the same for \":

```

3529 \def\@gls@checkescquote#1\"#2\"#3\null{%
3530 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3531 \toks@={#1}%
3532 \ifx\null#2\null
3533 \ifx\null#3\null
3534 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3535 \def\@@gls@checkescquote{\relax}%
3536 \else
3537 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3538 \@gls@quotechar\string\"@gls@quotechar
3539 \@gls@quotechar\string\"@gls@quotechar}%
3540 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3541 \fi
3542 \else
3543 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3544 \@gls@quotechar\string\"@gls@quotechar}%
3545 \ifx\null#3\null
3546 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3547 \else
3548 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3549 \fi
3550 \fi
3551 \@@gls@checkescquote
3552 }

```

@checkescactual Similarly for \? (which is replaces @ as makeindex's special character):

```

3553 \def\@gls@checkescactual#1\?#2\?#3\null{%
3554 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3555 \toks@={#1}%
3556 \ifx\null#2\null
3557 \ifx\null#3\null
3558 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```

```

3559 \def\@gls@checkescactual{\relax}%
3560 \else
3561 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3562 \@gls@quotechar\string"\@gls@actualchar
3563 \@gls@quotechar\string"\@gls@actualchar}%
3564 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3565 \fi
3566 \else
3567 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3568 \@gls@quotechar\string"\@gls@actualchar}%
3569 \ifx\null#3\null
3570 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3571 \else
3572 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3573 \fi
3574 \fi
3575 \@gls@checkescactual
3576 }

```

`gls@checkescbar` Similarly for `\|`:

```

3577 \def\@gls@checkescbar#1\|#2\|#3\null{%
3578 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3579 \toks@={#1}%
3580 \ifx\null#2\null
3581 \ifx\null#3\null
3582 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3583 \def\@gls@checkescbar{\relax}%
3584 \else
3585 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3586 \@gls@quotechar\string"\@gls@encapchar
3587 \@gls@quotechar\string"\@gls@encapchar}%
3588 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3589 \fi
3590 \else
3591 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3592 \@gls@quotechar\string"\@gls@encapchar}%
3593 \ifx\null#3\null
3594 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3595 \else
3596 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3597 \fi
3598 \fi
3599 \@gls@checkescbar
3600 }

```

`s@checkesclevel` Similarly for `\!`:

```

3601 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3602 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3603 \toks@={#1}%

```



```

3604 \ifx\null#2\null
3605 \ifx\null#3\null
3606 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3607 \def\@gls@checkesclevel{\relax}%
3608 \else
3609 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3610 \@gls@quotechar\string"\@gls@levelchar
3611 \@gls@quotechar\string"\@gls@levelchar}%
3612 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3613 \fi
3614 \else
3615 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3616 \@gls@quotechar\string"\@gls@levelchar}%
3617 \ifx\null#3\null
3618 \def\@gls@checkesclevel{\@gls@checkesclevel#2!!\null}%
3619 \else
3620 \def\@gls@checkesclevel{\@gls@checkesclevel#2!#3\null}%
3621 \fi
3622 \fi
3623 \@gls@checkesclevel
3624 }

```

\@gls@checkbar and for |:

```

3625 \def\@gls@checkbar#1|#2|#3\null{%
3626 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3627 \toks@={#1}%
3628 \ifx\null#2\null
3629 \ifx\null#3\null
3630 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3631 \def\@gls@checkbar{\relax}%
3632 \else
3633 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3634 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3635 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3636 \fi
3637 \else
3638 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3639 \@gls@quotechar\@gls@encapchar}%
3640 \ifx\null#3\null
3641 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3642 \else
3643 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3644 \fi
3645 \fi
3646 \@gls@checkbar
3647 }

```

@gls@checklevel and for !:

```

3648 \def\@gls@checklevel#1!#2!#3\null{%

```

```

3649 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3650 \toks@={#1}%
3651 \ifx\null#2\null
3652   \ifx\null#3\null
3653     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3654     \def\@gls@checklevel{\relax}%
3655   \else
3656     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3657       \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3658     \def\@gls@checklevel{\@gls@checklevel#3\null}%
3659   \fi
3660 \else
3661   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3662     \@gls@quotechar\@gls@levelchar}%
3663   \ifx\null#3\null
3664     \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3665   \else
3666     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3667   \fi
3668 \fi
3669 \@gls@checklevel
3670 }

```

gls@checkactual and for ?:

```

3671 \def\@gls@checkactual#1?#2?#3\null{%
3672   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3673   \toks@={#1}%
3674   \ifx\null#2\null
3675     \ifx\null#3\null
3676       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3677       \def\@gls@checkactual{\relax}%
3678     \else
3679       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3680         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3681       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3682     \fi
3683   \else
3684     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3685       \@gls@quotechar\@gls@actualchar}%
3686     \ifx\null#3\null
3687       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3688     \else
3689       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3690     \fi
3691   \fi
3692   \@gls@checkactual
3693 }

```

s@xdycheckquote As before but for use with xindy

```

3694 \def\@gls@xdycheckquote#1"#2"#3\null{%
3695   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3696   \toks@={#1}%
3697   \ifx\null#2\null
3698     \ifx\null#3\null
3699       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3700       \def\@gls@xdycheckquote{\relax}%
3701     \else
3702       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3703         \string"\string"}%
3704       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3705     \fi
3706   \else
3707     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3708       \string"}%
3709     \ifx\null#3\null
3710       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3711     \else
3712       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3713     \fi
3714   \fi
3715   \@gls@xdycheckquote
3716 }

```

ycheckbackslash Need to escape all backslashes for xindy. Define command that will define \@gls@xdycheckbackslash

```

3717 \edef\def\@gls@xdycheckbackslash{%
3718   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3719     ##2\@backslashchar##3\noexpand\null{%
3720     \noexpand\@gls@tmpb=\noexpand\expandafter
3721       {\noexpand\@gls@checkedmkidx}%
3722     \noexpand\toks@={##1}%
3723     \noexpand\ifx\noexpand\null##2\noexpand\null
3724       \noexpand\ifx\noexpand\null##3\noexpand\null
3725         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3726           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3727         \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3728       \noexpand\else
3729         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3730           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3731           \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3732       \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3733         \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3734       \noexpand\fi
3735     \noexpand\else
3736       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3737         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3738         \@backslashchar\@backslashchar}%
3739     \noexpand\ifx\noexpand\null##3\noexpand\null
3740       \noexpand\def\noexpand\@gls@xdycheckbackslash{%

```

```

3741 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3742 \@backslashchar\noexpand\null}%
3743 \noexpand\else
3744 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3745 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3746 ##3\noexpand\null}%
3747 \noexpand\fi
3748 \noexpand\fi
3749 \noexpand\@gls@xdycheckbackslash
3750 }%
3751 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3752 \def@gls@xdycheckbackslash

```

\glsdohypertarget

```

3753 \newlength@gls@tmplen
3754 \newcommand*{\glsdohypertarget}[2]{%
3755 \@glsshowtarget{#1}%
3756 \settoheight@gls@tmplen{#2}%
3757 \raisebox@gls@tmplen{\hypertarget{#1}-}{#2}%
3758 }

```

\glsdohyperlink

```

3759 \newcommand*{\glsdohyperlink}[2]{%
3760 \@glsshowtarget{#1}%
3761 \hyperlink{#1}{#2}%
3762 }

```

\glsdonohyperlink

```

3763 \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.

```

3764 \ifcsundef{hyperlink}%
3765 {%
3766 \let@glslink@glsdonohyperlink
3767 }%
3768 {%
3769 \let@glslink@glsdohyperlink
3770 }

```

\@glstarget If \hypertarget is not defined, \@glstarget ignores its first argument and just does the second argument, otherwise it is equivalent to \hypertarget.

```

3771 \ifcsundef{hypertarget}%
3772 {%
3773 \let@glstarget@secondoftwo
3774 }%

```

```

3775 {%
3776   \let\@glstarget\glstdohypertarget
3777 }

```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```

3778 \newcommand{\glsdisablehyper}{%
3779   \KV@glslink@hyperfalse
3780   \let\@glslink\glstdonohyperlink
3781   \let\@glstarget\@secondoftwo
3782 }

```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```

3783 \newcommand{\glsenablehyper}{%
3784   \KV@glslink@hypertrue
3785   \let\@glslink\glstdohyperlink
3786   \let\@glstarget\glstdohypertarget
3787 }

```

Provide some convenience commands if not already defined:

```

3788 \providecommand{\@firstofthree}[3]{#1}
3789 \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`

```
3790 \newrobustcmd*\gls{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```

3791 \newcommand*\@gls[2][ ]{%
3792   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}]%
3793 }

```

`\@gls@` Read in the final optional argument:

```
3794 \def\@gls@#1#2[#3]{%
3795   \glsdoifexists{#2}%
3796   {%
3797     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3798     \let\glsifplural\@secondoftwo
3799     \let\glscapscase\@firstofthree
3800     \let\glscustomtext\@empty
3801     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3802   \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.

```
3803   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3804   \ifKV@glslink@local
3805     \glsllocalunset{#2}%
3806   \else
3807     \glsunset{#2}%
3808   \fi
3809 }%
```

```
3810 \glspostlinkhook
3811 }
```

`\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`

```
3812 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3813 \newcommand*{\@Gls}[2] []{%
3814   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2} []}%
3815 }
```

`\@Gls@` Read in the final optional argument:

```
3816 \def\@Gls@#1#2[#3]{%
3817   \glsdoifexists{#2}%
3818   {%
3819     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3820 \let\glsifplural\@secondoftwo
3821 \let\glsifscapscase\@secondofthree
3822 \let\glsifcustomtext\@empty
3823 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```

3824 \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.

```

3825 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3826 \ifKV@glslink@local
3827 \glslocalunset{#2}%
3828 \else
3829 \glsunset{#2}%
3830 \fi
3831 }%

```

```

3832 \glspostlinkhook
3833 }

```

\GLS behaves like \gls, but the link text is converted to uppercase:

\GLS

```

3834 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

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

```

3835 \newcommand*{\@GLS}[2] [] {%
3836 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} []}%
3837 }

```

\@GLS@ Read in the final optional argument:

```

3838 \def\@GLS@#1#2[#3] {%
3839 \glsdoifexists{#2}%
3840 {%
3841 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3842 \let\glsifplural\@secondoftwo
3843 \let\glsifscapscase\@thirdofthree
3844 \let\glsifcustomtext\@empty
3845 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \@glo@text). Note that \@gls@link sets \glstype.

```

3846 \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.

```

3847 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```
3848 \ifKV@glslink@local
3849 \glsllocalunset{#2}%
3850 \else
3851 \glsunset{#2}%
3852 \fi
3853 }%

3854 \glspostlinkhook
3855 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3856 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3857 \newcommand*{\@glspl}[2] []{%
3858 \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2} []}%
3859 }
```

`\@glspl@` Read in the final optional argument:

```
3860 \def\@glspl@#1#2[#3]{%
3861 \glsdofexists{#2}%
3862 {%
3863 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3864 \let\glsifplural\@firstoftwo
3865 \let\glscapscase\@firstofthree
3866 \let\glscustomtext\@empty
3867 \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```
3868 \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.

```
3869 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3870 \ifKV@glslink@local
3871 \glsllocalunset{#2}%
3872 \else
3873 \glsunset{#2}%
3874 \fi
3875 }%

3876 \glspostlinkhook
3877 }
```


`\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`

```
3878 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3879 \newcommand*{\@Glspl}[2] [] {%
3880   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2} []}%
3881 }
```

`\@Glspl@` Read in the final optional argument:

```
3882 \def\@Glspl@#1#2[#3] {%
3883   \glsdoifexists{#2}%
3884   {%
3885     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3886     \let\glsifplural\@firstoftwo
3887     \let\glsapscase\@secondofthree
3888     \let\glscustomtext\@empty
3889     \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`.

```
3890   \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.

```
3891   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3892   \ifKV@glslink@local
3893     \glslocalunset{#2}%
3894   \else
3895     \glsunset{#2}%
3896   \fi
3897 }%
```

```
3898   \glspostlinkhook
3899 }
```

`\GLSp1` behaves like `\glsp1` except that all the link text is converted to uppercase.

`\GLSp1`

```
3900 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3901 \newcommand*{\@GLSp1}[2] [] {%
3902   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
3903 }
```

`\@GLSp1` Read in the final optional argument:

```

3904 \def\@GLSp1@#1#2[#3]{%
3905   \glsdoifexists{#2}%
3906   {%
3907     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3908     \let\glsifplural\@firstoftwo
3909     \let\glscapscase\@thirdofthree
3910     \let\glscustomtext\@empty
3911     \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstyp`.

```

3912   \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.

```

3913   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3914   \ifKV@glslink@local
3915     \glslocalunset{#2}%
3916   \else
3917     \glsunset{#2}%
3918   \fi
3919 }%

```

```

3920 \glspostlinkhook
3921 }

```

`\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:

```

3922 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3923 \newcommand*{\@glsdisp}[3][ ]{%
3924   \glsdoifexists{#2}{%

3925     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3926     \let\glsifplural\@secondoftwo
3927     \let\glscapscase\@firstofthree
3928     \def\glscustomtext{#3}%
3929     \def\glsinsert{}%

```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstyp`.

```
3930 \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.

```
3931 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3932 \ifKV@gls@link@local
3933 \glslocalunset{#2}%
3934 \else
3935 \glsunset{#2}%
3936 \fi
3937 }%
```

```
3938 \glspostlinkhook
3939 }
```

`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.

```
3940 \newcommand*\@gls@link@nocheckfirsthyper{}
```

`@gls@field@link`

```
3941 \newcommand{\@gls@field@link}[3]{%
3942 \glsdoifexists{#2}%
3943 {%
3944 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3945 \@gls@link[#1]{#2}{#3}%
3946 }%
3947 \glspostlinkhook
3948 }
```

`\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`

```
3949 \newrobustcmd*\glstext{\@gls@hyp@opt\@glstext}
```

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

```
3950 \newcommand*\@glstext[2][{}]{%
3951 \new@ifnextchar[\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[{}]}}
```

Read in the final optional argument:

```
3952 \def\@glstext@#1#2[#3]{%
3953 \@gls@field@link{#1}{#2}{\glsentrytext{#2}#3}%
3954 }
```

`\GLStext` behaves like `\glstext` except the text is converted to uppercase.

\GLStext

```
3955 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3956 \newcommand*{\@GLStext}[2] [] {%
```

```
3957   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3958 \def\@GLStext@#1#2[#3] {%
```

```
3959   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstrytext{#2}#3}}%
```

```
3960 }
```

\Glsfirst behaves like \glstry except that the first letter of the text is converted to uppercase.

\Glsfirst

```
3961 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3962 \newcommand*{\@Glsfirst}[2] [] {%
```

```
3963   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3964 \def\@Glsfirst@#1#2[#3] {%
```

```
3965   \@gls@field@link{#1}{#2}{\glstryfirst{#2}#3}}%
```

```
3966 }
```

\glstryfirst behaves like \glstry except it always uses the value given by the first key and it doesn't mark the entry as used.

\glstryfirst

```
3967 \newrobustcmd*{\glstryfirst}{\@gls@hyp@opt\@glstryfirst}
```

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

```
3968 \newcommand*{\@glstryfirst}[2] [] {%
```

```
3969   \new@ifnextchar[{\@glstryfirst@{#1}{#2}}{\@glstryfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3970 \def\@glstryfirst@#1#2[#3] {%
```

```
3971   \@gls@field@link{#1}{#2}{\glstryfirst{#2}#3}}%
```

```
3972 }
```

\Glsfirst behaves like \glstryfirst except it displays the first letter in uppercase.

\Glsfirst

```
3973 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3974 \newcommand*{\@Glsfirst}[2] [] {%
```

```
3975   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3976 \def\@Glsfirst@#1#2[#3]{%
3977   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3978 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
3979 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3980 \newcommand*{\@GLSfirst}[2] [] {%
3981   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3982 \def\@GLSfirst@#1#2[#3]{%
3983   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3984 }
```

\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

```
3985 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3986 \newcommand*{\@glsplural}[2] [] {%
3987   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3988 \def\@glsplural@#1#2[#3]{%
3989   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
3990 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
3991 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3992 \newcommand*{\@Glsplural}[2] [] {%
3993   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3994 \def\@Glsplural@#1#2[#3]{%
3995   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3996 }
```

\GLSplural behaves like \glsplural except that the text is converted to uppercase.

\GLSplural

```
3997 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
3998 \newcommand*{\@GLSplural}[2] [] {%  
3999   \new@ifnextchar [{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4000 \def\@GLSplural@#1#2[#3] {%  
4001   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}{#3}}}%  
4002 }
```

\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

```
4003 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
4004 \newcommand*{\@glsfirstplural}[2] [] {%  
4005   \new@ifnextchar [{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4006 \def\@glsfirstplural@#1#2[#3] {%  
4007   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}{#3}}%  
4008 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
4009 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
4010 \newcommand*{\@Glsfirstplural}[2] [] {%  
4011   \new@ifnextchar [{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4012 \def\@Glsfirstplural@#1#2[#3] {%  
4013   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}{#3}}%  
4014 }
```

\GLSfirstplural behaves like \glsfirstplural except that the link text is converted to uppercase.

\GLSfirstplural

```
4015 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
4016 \newcommand*{\@GLSfirstplural}[2] [] {%  
4017   \new@ifnextchar [{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4018 \def\@GLSfirstplural@#1#2[#3] {%  
4019   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}{#3}}}%  
4020 }
```

`\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`

```
4021 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
4022 \newcommand*{\@glsname}[2] [] {%
```

```
4023   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4024 \def\@glsname@#1#2[#3] {%
```

```
4025   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
```

```
4026 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
4027 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
4028 \newcommand*{\@Glsname}[2] [] {%
```

```
4029   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4030 \def\@Glsname@#1#2[#3] {%
```

```
4031   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
```

```
4032 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
4033 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4034 \newcommand*{\@GLSname}[2] [] {%
```

```
4035   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4036 \def\@GLSname@#1#2[#3] {%
```

```
4037   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
```

```
4038 }
```

`\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`

```
4039 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
4040 \newcommand*{\@glsdesc}[2] [] {%
```

```
4041   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4042 \def\@glsdesc#1#2[#3]{%
4043   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
4044 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
4045 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4046 \newcommand*{\@Glsdesc}[2][\%]
4047   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4048 \def\@Glsdesc#1#2[#3]{%
4049   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
4050 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
4051 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4052 \newcommand*{\@GLSdesc}[2][\%]
4053   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4054 \def\@GLSdesc#1#2[#3]{%
4055   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
4056 }
```

\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

```
4057 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4058 \newcommand*{\@glsdescplural}[2][\%]
4059   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4060 \def\@glsdescplural#1#2[#3]{%
4061   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
4062 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
4063 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```


Define the un-starred form. Need to determine if there is a final optional argument

```
4064 \newcommand*{\@Glsdescplural}[2] [] {%
4065   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4066 \def\@Glsdescplural@#1#2[#3] {%
4067   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
4068 }
```

`\Glsdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\Glsdescplural`

```
4069 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4070 \newcommand*{\@Glsdescplural}[2] [] {%
4071   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4072 \def\@Glsdescplural@#1#2[#3] {%
4073   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
4074 }
```

`\glsymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glsymbol`

```
4075 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

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

```
4076 \newcommand*{\@glsymbol}[2] [] {%
4077   \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4078 \def\@glsymbol@#1#2[#3] {%
4079   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
4080 }
```

`\Glsymbol` behaves like `\glsymbol` except that the first letter is converted to uppercase.

`\Glsymbol`

```
4081 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4082 \newcommand*{\@Glsymbol}[2] [] {%
4083   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4084 \def\@Glsymbol@#1#2[#3] {%
4085   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}%
4086 }
```

`\GLSsymbol` behaves like `\glsymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
4087 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4088 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
4089   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4090 \def\@GLSsymbol@#1#2[#3] {%
```

```
4091   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
```

```
4092 }
```

`\glsymbolplural` behaves like `\gls` except it always uses the value given by the symbol-plural key and it doesn't mark the entry as used.

`glsymbolplural`

```
4093 \newrobustcmd*{\glsymbolplural}{\@gls@hyp@opt\@glsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4094 \newcommand*{\@glsymbolplural}[2] [] {%
```

```
4095   \new@ifnextchar[{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4096 \def\@glsymbolplural@#1#2[#3] {%
```

```
4097   \@gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}}%
```

```
4098 }
```

`\Glsymbolplural` behaves like `\glsymbolplural` except that the first letter is converted to uppercase.

`Glsymbolplural`

```
4099 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4100 \newcommand*{\@Glsymbolplural}[2] [] {%
```

```
4101   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4102 \def\@Glsymbolplural@#1#2[#3] {%
```

```
4103   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}}%
```

```
4104 }
```

`\GLSsymbolplural` behaves like `\glsymbolplural` except that the link text is converted to uppercase.

`GLSsymbolplural`

```
4105 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4106 \newcommand*{\@GLSsymbolplural}[2] [] {%
```

```
4107   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4108 \def\@GLSsymbolplural@#1#2[#3]{%
4109   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
4110 }
```

`\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`

```
4111 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4112 \newcommand*{\@glsuseri}[2] [] {%
4113   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4114 \def\@glsuseri@#1#2[#3]{%
4115   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
4116 }
```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```
4117 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4118 \newcommand*{\@Glsuseri}[2] [] {%
4119   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4120 \def\@Glsuseri@#1#2[#3]{%
4121   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
4122 }
```

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
4123 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4124 \newcommand*{\@GLSuseri}[2] [] {%
4125   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4126 \def\@GLSuseri@#1#2[#3]{%
4127   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
4128 }
```

`\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`

```
4129 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
4130 \newcommand*{\@glsuserii}[2] [] {%  
4131   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4132 \def\@glsuserii@#1#2[#3] {%  
4133   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%  
4134 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
4135 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4136 \newcommand*{\@Glsuserii}[2] [] {%  
4137   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4138 \def\@Glsuserii@#1#2[#3] {%  
4139   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%  
4140 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
4141 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
4142 \newcommand*{\@GLSuserii}[2] [] {%  
4143   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4144 \def\@GLSuserii@#1#2[#3] {%  
4145   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%  
4146 }
```

\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

```
4147 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4148 \newcommand*{\@glsuseriii}[2] [] {%  
4149   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4150 \def\@glsuseriii@#1#2[#3] {%  
4151   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%  
4152 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
4153 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4154 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
4155   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4156 \def\@Glsuseriii@#1#2[#3] {%
```

```
4157   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
4158 }
```

\Glsuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
4159 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4160 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
4161   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4162 \def\@GLSuseriii@#1#2[#3] {%
```

```
4163   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
```

```
4164 }
```

\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

```
4165 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4166 \newcommand*{\@glsuseriv}[2] [] {%
```

```
4167   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4168 \def\@glsuseriv@#1#2[#3] {%
```

```
4169   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
4170 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\GLSuseriv

```
4171 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4172 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
4173   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4174 \def\@GLSuseriv@#1#2[#3] {%
```

```
4175   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
4176 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
4177 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4178 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
4179   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4180 \def\@GLSuseriv@#1#2[#3] {%
```

```
4181   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
```

```
4182 }
```

\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

```
4183 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4184 \newcommand*{\@glsuserv}[2] [] {%
```

```
4185   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4186 \def\@glsuserv@#1#2[#3] {%
```

```
4187   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}}%
```

```
4188 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
4189 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4190 \newcommand*{\@Glsuserv}[2] [] {%
```

```
4191   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4192 \def\@Glsuserv@#1#2[#3] {%
```

```
4193   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}}%
```

```
4194 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4195 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4196 \newcommand*{\@GLSuserv}[2] [] {%
```

```
4197   \new@ifnextchar[{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4198 \def\@GLSuserv@#1#2[#3]{%
4199   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4200 }
```

\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

```
4201 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4202 \newcommand*{\@glsuservi}[2][{}]{%
4203   \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4204 \def\@glsuservi@#1#2[#3]{%
4205   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
4206 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4207 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4208 \newcommand*{\@Glsuservi}[2][{}]{%
4209   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4210 \def\@Glsuservi@#1#2[#3]{%
4211   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
4212 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4213 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4214 \newcommand*{\@GLSuservi}[2][{}]{%
4215   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4216 \def\@GLSuservi@#1#2[#3]{%
4217   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
4218 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4219 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4220 \newcommand*{\ns@acrshort}[2] [] {%
4221   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} [] }%
4222 }
```

Read in the final optional argument:

```
4223 \def\@acrshort#1#2[#3] {%
4224   \glsdoifexists{#2}%
4225   {%
4226     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4227     \let\glsifplural\@secondoftwo
4228     \let\glsapscase\@firstofthree
4229     \let\glsinsert\@empty
4230     \def\glscustomtext{%
4231       \acronymfont{\glsentryshort{#2}}#3%
4232     }%
4233     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4234   }%
4235   \glspostlinkhook
4236 }
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4233   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4234   }%
4235   \glspostlinkhook
4236 }
```

\Acrshort

```
4237 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\ns@Acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4238 \newcommand*{\ns@Acrshort}[2] [] {%
4239   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} [] }%
4240 }
```

Read in the final optional argument:

```
4241 \def\@Acrshort#1#2[#3] {%
4242   \glsdoifexists{#2}%
4243   {%
4244     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4245     \def\glslabel{#2}%
4246     \let\glsifplural\@secondoftwo
4247     \let\glsapscase\@secondofthree
4248     \let\glsinsert\@empty
4249     \def\glscustomtext{%
4250       \acronymfont{\Glsentryshort{#2}}#3%
4251     }%
4252     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4253   }%
4254   \glspostlinkhook
4255 }
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4252   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4253   }%
4254   \glspostlinkhook
4255 }
```



```

4254 \glspostlinkhook
4255 }

```

\ACRshort

```

4256 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\@ns@ACRshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4257 \newcommand*{\ns@ACRshort}[2][\%
4258   \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
4259 }

```

Read in the final optional argument:

```

4260 \def\@ACRshort#1#2[#3]{%
4261   \glsdoifexists{#2}%
4262   {%
4263     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4264     \def\glslabel{#2}%
4265     \let\glsifplural\@secondoftwo
4266     \let\glsapscase\@thirdofthree
4267     \let\glsinsert\@empty
4268     \def\glscustomtext{%
4269       \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4270     }%

```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```

4271   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4272   }%
4273   \glspostlinkhook
4274 }

```

Short plural:

\acrshortpl

```

4275 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\@ns@acrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4276 \newcommand*{\ns@acrshortpl}[2][\%
4277   \new@ifnextchar[\@acrshortpl{#1}{#2}]{\@acrshortpl{#1}{#2}[]}%
4278 }

```

Read in the final optional argument:

```

4279 \def\@acrshortpl#1#2[#3]{%
4280   \glsdoifexists{#2}%
4281   {%
4282     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4283 \def\glslabel{#2}%
4284 \let\glsifplural\@firstoftwo
4285 \let\glsupcase\@firstofthree
4286 \let\glsinsert\@empty
4287 \def\glscustomtext{%
4288 \acronymfont{\glsentryshortpl{#2}}#3%
4289 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4290 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4291 }%

4292 \glspostlinkhook
4293 }

```

\Acrshortpl

```

4294 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4295 \newcommand*{\ns@Acrshortpl}[2] [] {%
4296 \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
4297 }

Read in the final optional argument:
4298 \def\@Acrshortpl#1#2[#3] {%
4299 \glsdoifexists{#2}%
4300 {%

4301 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4302 \def\glslabel{#2}%
4303 \let\glsifplural\@firstoftwo
4304 \let\glsupcase\@secondofthree
4305 \let\glsinsert\@empty
4306 \def\glscustomtext{%
4307 \acronymfont{\Glsentryshortpl{#2}}#3%
4308 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4309 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4310 }%

4311 \glspostlinkhook
4312 }

```

\ACRshortpl

```

4313 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4314 \newcommand*{\ns@ACRshortpl}[2] [] {%
4315 \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
4316 }

```

Read in the final optional argument:

```
4317 \def\@ACRshortpl#1#2[#3]{%
4318   \glsdoifexists{#2}%
4319   {%
4320     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4321     \def\glslabel{#2}%
4322     \let\glsifplural\@firstoftwo
4323     \let\glscapscase\@thirdofthree
4324     \let\glsinsert\@empty
4325     \def\glscustomtext{%
4326       \mfirstucMakeUppercase{\acronymfont{\glentryshortpl{#2}}#3}%
4327     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4328   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4329   }%
4330 \glspostlinkhook
4331 }
```

\acrlong

```
4332 \newrobustcmd*{\acrlong}{\@gl@hyp@opt\@ns@acrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4333 \newcommand*{\ns@acrlong}[2][ ]{%
4334   \new@ifnextchar{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[ ]}%
4335 }
```

Read in the final optional argument:

```
4336 \def\@acrlong#1#2[#3]{%
4337   \glsdoifexists{#2}%
4338   {%
4339     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4340     \def\glslabel{#2}%
4341     \let\glsifplural\@secondoftwo
4342     \let\glscapscase\@firstofthree
4343     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4344   \def\glscustomtext{%
4345     \glentrylong{#2}#3%
4346   }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4347   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4348   }%
```

```

4349 \glspostlinkhook
4350 }

```

\Acrlong

```

4351 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4352 \newcommand*{\ns@Acrlong}[2] [] {%
4353   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%
4354 }

```

Read in the final optional argument:

```

4355 \def\@Acrlong#1#2[#3] {%
4356   \glsdoifexists{#2}%
4357   {%
4358     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4359     \def\glslabel{#2}%
4360     \let\glsifplural\@secondoftwo
4361     \let\glsapscase\@secondofthree
4362     \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4363   \def\glscustomtext{%
4364     \Glsentrylong{#2}#3%
4365   }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4366   \@gls@link[#1]{#2}{\csname gls\@glstype @entryfmt\endcsname}%
4367   }%
4368   \glspostlinkhook
4369 }

```

\ACRlong

```

4370 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\@ns@ACRlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4371 \newcommand*{\ns@ACRlong}[2] [] {%
4372   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4373 }

```

Read in the final optional argument:

```

4374 \def\@ACRlong#1#2[#3] {%
4375   \glsdoifexists{#2}%
4376   {%
4377     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4378 \def\glslabel{#2}%
4379 \let\glsifplural\@secondoftwo
4380 \let\glsupcase\@thirdofthree
4381 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4382 \def\glscustomtext{%
4383 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4384 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4385 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4386 }%

```

```

4387 \glspostlinkhook
4388 }

```

Short plural:

\acrlongpl

```

4389 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4390 \newcommand*{\ns@acrlongpl}[2][\%
4391 \new@ifnextchar[\@acrlongpl{#1}{#2}]{\@acrlongpl{#1}{#2}[]}%
4392 }

```

Read in the final optional argument:

```

4393 \def\@acrlongpl#1#2[#3]{%
4394 \glsdoifexists{#2}%
4395 {%
4396 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4397 \def\glslabel{#2}%
4398 \let\glsifplural\@firstoftwo
4399 \let\glsupcase\@firstofthree
4400 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4401 \def\glscustomtext{%
4402 \glsentrylongpl{#2}#3%
4403 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4404 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4405 }%

```

```

4406 \glspostlinkhook
4407 }

```

\Acrlongpl

```
4408 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4409 \newcommand*{\ns@Acrlongpl}[2][\%  
4410 \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%  
4411 }
```

Read in the final optional argument:

```
4412 \def\@Acrlongpl#1#2[#3]{%  
4413 \glsdoifexists{#2}%  
4414 {%  
  
4415 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4416 \def\glslabel{#2}%  
4417 \let\glsifplural\@firstoftwo  
4418 \let\glsapscase\@secondofthree  
4419 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4420 \def\glscustomtext{%  
4421 \Glsentrylongpl{#2}#3%  
4422 }%
```

Call \@gls@link. Note that \@gls@link sets \glsstyle.

```
4423 \@gls@link[#1]{#2}{\csname gls@glstyle @entryfmt\endcsname}%  
4424 }%  
  
4425 \glspostlinkhook  
4426 }
```

\ACRlongpl

```
4427 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\ns@ACRlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4428 \newcommand*{\ns@ACRlongpl}[2][\%  
4429 \new@ifnextchar[\@ACRlongpl{#1}{#2}]{\@ACRlongpl{#1}{#2}[]}%  
4430 }
```

Read in the final optional argument:

```
4431 \def\@ACRlongpl#1#2[#3]{%  
4432 \glsdoifexists{#2}%  
4433 {%  
  
4434 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4435 \def\glslabel{#2}%  
4436 \let\glsifplural\@firstoftwo  
4437 \let\glsapscase\@thirdofthree  
4438 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4439 \def\glscustomtext{%
4440 \mfirstucMakeUppercase{\glsenrylongpl{#2}#3}%
4441 }%

Call \@gls@link. Note that \@gls@link sets \glstype.
4442 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4443 }%

4444 \glspostlinkhook
4445 }

```

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>}`

```

4446 \newcommand*{\@gls@entry@field}[2]{%
4447 \csname glo@\glsdetoklabel{#1}@\#2\endcsname
4448 }

```

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

```

4449 \newcommand*{\glsletentryfield}[3]{%
4450 \letcs{#1}{glo@\glsdetoklabel{#2}@\#3}%
4451 }

```

`\Gls@entry@field` Generic first letter uppercase version.

`\@Gls@entry@field{<label>}{<field>}`

```

4452 \newcommand*{\@Gls@entry@field}[2]{%
4453 \glsdoifexistsordo{#1}%
4454 {%
4455 \letcs{\@glo@text}{glo@\glsdetoklabel{#1}@\#2}%
4456 \ifdef\@glo@text
4457 {%
4458 \xmakefirstuc{\@glo@text}%
4459 }%
4460 }%
4461 ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4462 entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry

```

```

4463     label and the field name}%
4464 }%
4465 }%
4466 {%
4467   ??%
4468 }%
4469 }

```

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`

```

4470 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4471 \newrobustcmd*{\Glsentryname}[1]{%
4472   \@Gls@entryname{#1}%
4473 }

```

`\@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:

```

4474 \newcommand*{\@Gls@entryname}[1]{%
4475   \@Gls@entry@field{#1}{name}%
4476 }

```

`\ls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

4477 \newcommand*{\@Gls@acrentryname}[1]{%
4478   \ifglshaslong{#1}%
4479   {%
4480     \letcs\@glo@text{glo\@glsdetoklabel{#1}@name}%
4481     \expandafter\@gls@getbody\@glo@text{}\@nil
4482     \expandafter\ifx\@gls@body\glsentrylong\relax
4483     \expandafter\Glsentrylong\@gls@rest
4484   }else
4485     \expandafter\ifx\@gls@body\glsentryshort\relax
4486     \expandafter\Glsentryshort\@gls@rest
4487   }else
4488     \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.

```

4489     {%
4490       \let\glsentryshort\Glsentryshort
4491       \@glo@text
4492     }%
4493   }else

```



```

4494      \xmakefirstuc{\@glo@text}%
4495      \fi
4496      \fi
4497      \fi
4498  }%
4499  {%

```

Not an acronym

```

4500      \@Gls@entry@field{#1}{name}%
4501  }%
4502 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

`\glsentrydesc`

```

4503 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4504 \newrobustcmd*{\Glsentrydesc}[1]{%
4505   \@Gls@entry@field{#1}{desc}%
4506 }

```

Plural form:

`entrydescplural`

```

4507 \newcommand*{\glsentrydescplural}[1]{%
4508   \@gls@entry@field{#1}{descplural}%
4509 }

```

`entrydescplural`

```

4510 \newrobustcmd*{\Glsentrydescplural}[1]{%
4511   \@Gls@entry@field{#1}{descplural}%
4512 }

```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

`\glsentrytext`

```

4513 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}

```

`\Glsentrytext`

```

4514 \newrobustcmd*{\Glsentrytext}[1]{%
4515   \@Gls@entry@field{#1}{text}%
4516 }

```

Get the plural form:

`\glentryplural`

```
4517 \newcommand*{\glentryplural}[1]{%
4518   \@gls@entry@field{#1}{plural}%
4519 }
```

`\Glsentryplural`

```
4520 \newrobustcmd*{\Glsentryplural}[1]{%
4521   \@Gls@entry@field{#1}{plural}%
4522 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glentrysymbol`

```
4523 \newcommand*{\glentrysymbol}[1]{%
4524   \@gls@entry@field{#1}{symbol}%
4525 }
```

`\Glsentrysymbol`

```
4526 \newrobustcmd*{\Glsentrysymbol}[1]{%
4527   \@Gls@entry@field{#1}{symbol}%
4528 }
```

Plural form:

`trysymbolplural`

```
4529 \newcommand*{\glentrysymbolplural}[1]{%
4530   \@gls@entry@field{#1}{symbolplural}%
4531 }
```

`trysymbolplural`

```
4532 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4533   \@Gls@entry@field{#1}{symbolplural}%
4534 }
```

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`

```
4535 \newcommand*{\glentryfirst}[1]{%
4536   \@gls@entry@field{#1}{first}%
4537 }
```

`\Glsentryfirst`

```
4538 \newrobustcmd*{\Glsentryfirst}[1]{%
4539   \@Gls@entry@field{#1}{first}%
4540 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

entryfirstplural

```
4541 \newcommand*{\glsentryfirstplural}[1]{%
4542   \@gls@entry@field{#1}{firstpl}%
4543 }
```

entryfirstplural

```
4544 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4545   \@Gls@entry@field{#1}{firstpl}%
4546 }
```

entrytitlecase

```
4547 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4548   \glsfieldfetch{#1}{#2}{\@gls@value}%
4549   \xcapitalisewords{\@gls@value}%
4550 }
4551 \ifdef\texorpdfstring
4552 {
4553   \newcommand*{\glsentrytitlecase}[2]{%
4554     \texorpdfstring
4555       {\@glsentrytitlecase{#1}{#2}}%
4556       {\@gls@entry@field{#1}{#2}}%
4557   }
4558 }
4559 {
4560   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4561 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
4562 \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

```
4563 \newcommand*{\glsentrysort}[1]{%
4564   \@gls@entry@field{#1}{sort}%
4565 }
```

\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.

```
4566 \newcommand*{\glsentryuseri}[1]{%
4567   \@gls@entry@field{#1}{useri}%
4568 }
```

\Glsentryuseri

```
4569 \newrobustcmd*{\Glsentryuseri}[1]{%
```

```

4570 \@Gls@entry@field{#1}{useri}%
4571 }

```

`\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.

```

4572 \newcommand*{\glentryuserii}[1]{%
4573 \@Gls@entry@field{#1}{userii}%
4574 }

```

`\Glsentryuserii`

```

4575 \newrobustcmd*{\Glsentryuserii}[1]{%
4576 \@Gls@entry@field{#1}{userii}%
4577 }

```

`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.

```

4578 \newcommand*{\glentryuseriii}[1]{%
4579 \@Gls@entry@field{#1}{useriii}%
4580 }

```

`Glsentryuseriii`

```

4581 \newrobustcmd*{\Glsentryuseriii}[1]{%
4582 \@Gls@entry@field{#1}{useriii}%
4583 }

```

`\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.

```

4584 \newcommand*{\glentryuseriv}[1]{%
4585 \@Gls@entry@field{#1}{useriv}%
4586 }

```

`\Glsentryuseriv`

```

4587 \newrobustcmd*{\Glsentryuseriv}[1]{%
4588 \@Gls@entry@field{#1}{useriv}%
4589 }

```

`\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.

```

4590 \newcommand*{\glentryuserv}[1]{%
4591 \@Gls@entry@field{#1}{userv}%
4592 }

```

`\Glsentryuserv`

```

4593 \newrobustcmd*{\Glsentryuserv}[1]{%
4594 \@Gls@entry@field{#1}{userv}%
4595 }

```

`\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.

```
4596 \newcommand*{\glentryuservi}[1]{%
4597   \@gls@entry@field{#1}{uservi}%
4598 }
```

`\Glsentryuservi`

```
4599 \newrobustcmd*{\Glsentryuservi}[1]{%
4600   \@Gls@entry@field{#1}{uservi}%
4601 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4602 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4603 \newrobustcmd*{\Glsentryshort}[1]{%
4604   \@Gls@entry@field{#1}{short}%
4605 }
```

`glentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4606 \newcommand*{\glentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`Glsentryshortpl`

```
4607 \newrobustcmd*{\Glsentryshortpl}[1]{%
4608   \@Gls@entry@field{#1}{shortpl}%
4609 }
```

`\glentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4610 \newcommand*{\glentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4611 \newrobustcmd*{\Glsentrylong}[1]{%
4612   \@Gls@entry@field{#1}{long}%
4613 }
```

`\glentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4614 \newcommand*{\glentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4615 \newrobustcmd*{\Glsentrylongpl}[1]{%
4616   \@Gls@entry@field{#1}{longpl}%
4617 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4618 \newcommand*{\glsentryfull}[1]{%
4619   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4620 }
```

`\Glsentryfull`

```
4621 \newrobustcmd*{\Glsentryfull}[1]{%
4622   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4623 }
```

`\glsentryfullpl`

```
4624 \newcommand*{\glsentryfullpl}[1]{%
4625   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4626 }
```

`\Glsentryfullpl`

```
4627 \newrobustcmd*{\Glsentryfullpl}[1]{%
4628   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4629 }
```

`entrynumberlist` Displays the number list as is.

```
4630 \newcommand*{\glsentrynumberlist}[1]{%
4631   \glsdoifexists{#1}%
4632   {%
4633     \@gls@entry@field{#1}{numberlist}%
4634   }%
4635 }
```

`splaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4636 \@ifpackageloaded{hyperref} {%
4637   \newcommand*{\glsdisplaynumberlist}[1]{%
4638     \GlossariesWarning
4639     {%
4640       \string\glsdisplaynumberlist\space
4641       doesn't work with hyperref.^^JUsing
4642       \string\glsentrynumberlist\space instead%
4643     }%
4644     \glsentrynumberlist{#1}%
4645   }%
4646 }%
4647 {%
4648   \newcommand*{\glsdisplaynumberlist}[1]{%
4649     \glsdoifexists{#1}%
4650     {%
4651       \bgroup
```



```

4687 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4688 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

This key is only used by `\glsaddall`:

```

4689 \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`

```

4690 \newrobustcmd*{\glsadd}[2] [] {%

```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4691 \@gls@adjustmode
4692 \glsdoifexists{#2}%
4693 {%
4694 \def\@glsnumberformat{glsnumberformat}%
4695 \edef\@gls@counter{\csname glo@glstetoklabel{#2}@counter\endcsname}%
4696 \setkeys{glossadd}{#1}%

```

Store the entry's counter in `\theglsentrycounter`

```

4697 \@gls@saveentrycounter

```

Define sort key if necessary:

```

4698 \@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.

```

4699 \@@do@wrglossary{#2}%
4700 }%
4701 }

```

`@gls@adjustmode`

```

4702 \newcommand*{\@gls@adjustmode}{}
4703 \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`

```

4704 \newrobustcmd*{\glsaddall}[1] [] {%
4705 \edef\@glo@type{\@glo@types}%

```



```

4706 \setkeys{glossadd}{#1}%
4707 \forallglsentries[\@glo@type]{\@glo@entry}{%
4708   \glsadd[#1]{\@glo@entry}%
4709 }%
4710 }

```

```
\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.

```

4711 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4712   \forallglsentries[#1]{\@glo@entry}%
4713   {%
4714     \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4715   }%
4716 }

```

`\glsignore`

```
4717 \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.

```
4718 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4719 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4720 \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.

```
4721 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4722 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4723 \edef\glstildechar{\string~}
```

`@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4724 \ifglsxindy
4725   \newcommand*{\@glsfirstletter}{A}
4726 \fi
```

`letterAfterDigits` Sets the first letter to come after the digits 0,...,9. The starred version sanitizes.

```
4727 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}{%
4728   \@ifstar\s@GlsSetXdyFirstLetterAfterDigits\@GlsSetXdyFirstLetterAfterDigits}
4729 \ifglsxindy
4730   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4731     \renewcommand*{\@glsfirstletter}{#1}}
4732   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}[1]{%
4733     \renewcommand*{\@glsfirstletter}{#1}%
4734     \@onelevel@sanitize\@glsfirstletter
4735   }
4736 \else
4737   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4738     \glsnoindywarning\GlsSetXdyFirstLetterAfterDigits}
4739   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}{%
4740     \@GlsSetXdyFirstLetterAfterDigits
4741   }
4742 \fi
```

`numbergrouporder` Specifies the order of the number group.

```
4743 \ifglsxindy
4744   \newcommand*{\@xdynumbergrouporder}{:before \string"\@glsfirstletter\string"}
4745 \fi
```

`numberGroupOrder` Sets the relative location of the number group. The starred version sanitizes.

```
4746 \newcommand*{\GlsSetXdyNumberGroupOrder}[1]{%
4747   \@ifstar\s@GlsSetXdyNumberGroupOrder\@GlsSetXdyNumberGroupOrder
4748 }
4749 \ifglsxindy
4750   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4751     \renewcommand*{\@xdynumbergrouporder}{#1}%
4752   }
4753 \fi
```

```

4752 }
4753 \newcommand*{\s@GlsSetXdyNumberGroupOrder}[1]{%
4754   \renewcommand*{\@xdynumbergrouporder}{#1}%
4755   \@onelevel@sanitize\@xdynumbergrouporder
4756 }
4757 \else
4758 \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4759   \glsnoindywarning\GlsSetXdyNumberGroupOrder}
4760 \newcommand*{\s@GlsSetXdyNumberGroupOrder}{%
4761   \@GlsSetXdyNumberGroupOrder}
4762 \fi

```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```

4763 \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.

```

4764 \ifglsxindy
4765   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4766     \renewcommand*{\@glsminrange}{#1}}
4767 \else
4768   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4769     \glsnoindywarning\GlsSetXdyMinRangeLength}
4770 \fi

```

`\writeist`

```

4771 \ifglsxindy

```

Code to use if xindy is required.

```

4772 \def\writeist{%

```

Define write register if not already defined

```

4773   \ifundef{\glswrite}{\newwrite\glswrite}{}%

```

Update attributes list

```

4774   \@gls@addpredefinedattributes

```

Open the file.

```

4775   \openout\glswrite=\istfilename

```

Write header comment at the start of the file

```

4776   \write\glswrite{;; xindy style file created by the glossaries
4777     package}%
4778   \write\glswrite{;; for document '\jobname' on
4779     \the\year-\the\month-\the\day}%

```

Specify the required styles

```

4780   \write\glswrite{^^J; required styles^^J}
4781   \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4782     \ifx\@xdystyle\@empty
4783     \else

```

```

4784         \protected@write\glswrite{}\{(require
4785         \string"\@xdystyle.xdy\string")}\}%
4786     \fi
4787 }%

```

List the allowed attributes (possible values used by the format key)

```

4788     \write\glswrite{^^J%
4789     ; list of allowed attributes (number formats)^^J}%
4790     \write\glswrite{(define-attributes ((\@xdyattributes)))}%

```

Define any additional alphabets

```

4791     \write\glswrite{^^J; user defined alphabets^^J}%
4792     \write\glswrite{\@xdyuseralphabets}%

```

Define location classes.

```

4793     \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.

```

4794     \@for\@gls@classI:=\@gls@xdy@locationlist\do{%

```

Case where $\langle Hprefix \rangle$ is empty:

```

4795         \protected@write\glswrite{}\{(define-location-class
4796         \string"\@gls@classI\string"^^J\space\space\space
4797         (
4798             :sep "{{"
4799             \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4800             :sep "}"
4801         )
4802         ^^J\space\space\space
4803         :min-range-length \@glsminrange^^J%
4804     )
4805 }%

```

Nested iteration over all classes:

```

4806     {%
4807         \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4808             \protected@write\glswrite{}\{(define-location-class
4809             \string"\@gls@classII-\@gls@classI\string"
4810             ^^J\space\space\space
4811             (
4812                 :sep "{{"
4813                 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4814                 :sep "{{"
4815                 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4816                 :sep "}"
4817             )
4818             ^^J\space\space\space
4819             :min-range-length \@glsminrange^^J%
4820         )
4821     }%
4822 }%

```

```

4823     }%
4824 }%

```

User defined location classes (needs checking for new location format).

```

4825 \write\glswrite{^^J; user defined location classes}%
4826 \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.)

```

4827 \write\glswrite{^^J; define cross-reference class^^J}%
4828 \write\glswrite{(define-crossref-class \string"see\string"
4829 :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.)

```

4830 \write\glswrite{(markup-crossref-list
4831 :class \string"see\string"^^J\space\space\space
4832 :open \string"\string\glsseeformat\string"
4833 :close \string"{}\string")}%

```

Provide hook to write extra material here (used by glossaries-extra to define a seealso class).

```

4834 \@xdycrossrefhook

```

List the order to sort the classes.

```

4835 \write\glswrite{^^J; define the order of the location classes}%
4836 \write\glswrite{(define-location-class-order
4837 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4838 \write\glswrite{^^J; define the glossary markup^^J}%
4839 \write\glswrite{(markup-index^^J\space\space\space
4840 :open \string"\string
4841 \glossarysection[\string\glossarytoctitle]{\string
4842 \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)

```

4843 \@for\@this@ctr:=\@xdycounters\do{%
4844 {%
4845 \@for\@this@attr:=\@xdyattributelist\do{%
4846 \protected@write\glswrite{}\string\providecommand*%
4847 \expandafter\string
4848 \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4849 {%
4850 \string\setentrycounter
4851 [\expandafter\@gobble\string\#1]{\@this@ctr}%
4852 \expandafter\string

```

```

4853             \csname\@this@attr\endcsname
4854             {\expandafter\@gobble\string\#2}%
4855         }%
4856     }%
4857 }%
4858 }%
4859 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4860 \write\glswrite{%
4861     \string\begin
4862     {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4863     \space\space:close \string"\glpercentchar\glstildechar n\string
4864     \end{theglossary}\string\glossarypostamble
4865     \glstildechar n\string" ^^J\space\space\space
4866     :tree)}}%

```

Specify what to put between letter groups

```

4867 \write\glswrite{(markup-letter-group-list
4868     :sep \string"\string\glsgroupskip\glstildechar n\string"))}%

```

Specify what to put between entries

```

4869 \write\glswrite{(markup-indexentry
4870     :open \string"\string\relax \string\glresetentrylist
4871     \glstildechar n\string"))}%

```

Specify how to format entries

```

4872 \write\glswrite{(markup-locclass-list :open
4873     \string"\glsoopenbrace\string\glossaryentrynumbers
4874     \glsoopenbrace\string\relax\space \string"^^J\space\space\space
4875     :sep \string", \string"
4876     :close \string"\glsclosebrace\glsclosebrace\string"))}%

```

Specify how to separate location numbers

```

4877 \write\glswrite{(markup-locref-list
4878     :sep \string"\string\delimN\space\string"))}%

```

Specify how to indicate location ranges

```

4879 \write\glswrite{(markup-range
4880     :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.

```

4881 \@onelevel@sanitize\gls@suffiF
4882 \@onelevel@sanitize\gls@suffiFF
4883 \ifx\gls@suffiF\@empty
4884 \else
4885     \write\glswrite{(markup-range
4886         :close "\gls@suffiF" :length 1 :ignore-end)}}%
4887 \fi
4888 \ifx\gls@suffiFF\@empty

```

```

4889 \else
4890 \write\glswrite{(markup-range
4891 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4892 \fi

Specify how to format locations.
4893 \write\glswrite{^^J; define format to use for locations^^J}%
4894 \write\glswrite{@xdylocref}%

Specify how to separate letter groups.
4895 \write\glswrite{^^J; define letter group list format^^J}%
4896 \write\glswrite{(markup-letter-group-list
4897 :sep \string"\string\glsgroupskip\glstildechar n\string")}}%

Define letter group headings.
4898 \write\glswrite{^^J; letter group headings^^J}%
4899 \write\glswrite{(markup-letter-group
4900 :open-head \string"\string\glsgroupheading
4901 \glsopenbrace\string"^^J\space\space\space
4902 :close-head \string"\glsclosebrace\string")}}%

Define additional letter groups.
4903 \write\glswrite{^^J; additional letter groups^^J}%
4904 \write\glswrite{@xdylettergroups}%

Define additional sort rules
4905 \write\glswrite{^^J; additional sort rules^^J}%
4906 \write\glswrite{@xdysortrules}%

Hook for any additional information:
4907 \@gls@writeisthook

Close the style file
4908 \closeout\glswrite

Suppress any further calls.
4909 \let\writeist\relax
4910 }
4911 \else

Code to use if makeindex is required.
4912 \edef\@gls@actualchar{\string?}
4913 \edef\@gls@encapchar{\string|}
4914 \edef\@gls@levelchar{\string!}
4915 \edef\@gls@quotechar{\string"%}
4916 \let\GlsSetQuote\gls@nosetquote
4917 \def\writeist{\relax
4918 \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4919 \openout\glswrite=\istfilename
4920 \write\glswrite{\glspercentchar\space makeindex style file
4921 created by the glossaries package}
4922 \write\glswrite{\glspercentchar\space for document
4923 '\jobname' on \the\year-\the\month-\the\day}

```

```

4924 \write\glswrite{actual '@gls@actualchar'}
4925 \write\glswrite{encap '@gls@encapchar'}
4926 \write\glswrite{level '@gls@levelchar'}
4927 \write\glswrite{quote '@gls@quotechar'}
4928 \write\glswrite{keyword \string\string\glossaryentry\string}
4929 \write\glswrite{preamble \string\string\glossarysection[\string
4930 \glossarytoctitle]{\string\glossarytitle}\string
4931 \glossarypreamble\string\n\string\begin{theglossary}\string
4932 \glossaryheader\string\n\string}
4933 \write\glswrite{postamble \string\string%\string\n\string
4934 \end{theglossary}\string\glossarypostamble\string\n
4935 \string}
4936 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
4937 \string}
4938 \write\glswrite{item_0 \string\string%\string\n\string}
4939 \write\glswrite{item_1 \string\string%\string\n\string}
4940 \write\glswrite{item_2 \string\string%\string\n\string}
4941 \write\glswrite{item_01 \string\string%\string\n\string}
4942 \write\glswrite{item_x1
4943 \string\string\relax \string\glsresetentrylist\string\n
4944 \string}
4945 \write\glswrite{item_12 \string\string%\string\n\string}
4946 \write\glswrite{item_x2
4947 \string\string\relax \string\glsresetentrylist\string\n
4948 \string}

4949 \write\glswrite{delim_0 \string\string\{\string
4950 \glossaryentrynumbers\string\{\string\relax \string}
4951 \write\glswrite{delim_1 \string\string\{\string
4952 \glossaryentrynumbers\string\{\string\relax \string}
4953 \write\glswrite{delim_2 \string\string\{\string
4954 \glossaryentrynumbers\string\{\string\relax \string}
4955 \write\glswrite{delim_t \string\string\}\string\}\string}
4956 \write\glswrite{delim_n \string\string\delimN \string}
4957 \write\glswrite{delim_r \string\string\delimR \string}
4958 \write\glswrite{headings_flag 1}
4959 \write\glswrite{heading_prefix
4960 \string\string\glsgroupheading\string\{\string}
4961 \write\glswrite{heading_suffix
4962 \string\string\}\string\relax
4963 \string\glsresetentrylist \string}
4964 \write\glswrite{symhead_positive \string\glsymbols\string}
4965 \write\glswrite{numhead_positive \string\glsnumbers\string}
4966 \write\glswrite{page_compositor \string\glscompositor\string}
4967 \@gls@escbsdq\gls@suffixF
4968 \@gls@escbsdq\gls@suffixFF
4969 \ifx\gls@suffixF\@empty
4970 \else
4971 \write\glswrite{suffix_2p \string\gls@suffixF\string}
4972 \fi

```



```

4973 \ifx\gls@suffixFF\@empty
4974 \else
4975 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4976 \fi

```

Hook for any additional information:

```

4977 \@gls@writeisthook

```

Close the file and disable \writeist.

```

4978 \closeout\glswrite
4979 \let\writeist\relax
4980 }
4981 \fi

```

SetWriteIstHook Allow user to append information to the style file.

```

4982 \newcommand*{\GlsSetWriteIstHook}[1]{\renewcommand*{\@gls@writeisthook}{#1}}
4983 \@onlypremakeg\GlsSetWriteIstHook

```

ls@writeisthook

```

4984 \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.

```

4985 \ifglxsindy
4986 \newcommand*{\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4987 \newcommand*{\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4988 \else
4989 \newcommand*{\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}%

```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```

4990 \@ifpackageloaded{tracklang}%
4991 {%
4992 \IfTrackedLanguage{german}%
4993 {%
4994 \def\@gls@extramakeindexopts{-g}%
4995 }%
4996 }%
4997 }%
4998 {}%

```

Need to redefine \@gls@checkquote

```

4999 \edef\@gls@docheckquotedef{%
5000 \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
5001 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5002 \noexpand\toks@={####1}%
5003 \noexpand\ifx\noexpand\null####2\noexpand\null
5004 \noexpand\ifx\noexpand\null####3\noexpand\null
5005 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5006 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5007 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%

```

```

5008 \noexpand\else
5009 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5010 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5011 \noexpand\@gls@quotearch\noexpand\@gls@quotearch
5012 \noexpand\@gls@quotearch\noexpand\@gls@quotearch}%
5013 \noexpand\def\noexpand\@gls@checkquote{%
5014 \noexpand\@gls@checkquote####3\noexpand\null}%
5015 \noexpand\fi
5016 \noexpand\else
5017 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5018 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5019 \noexpand\@gls@quotearch\noexpand\@gls@quotearch}%
5020 \noexpand\ifx\noexpand\null####3\noexpand\null
5021 \noexpand\def\noexpand\@gls@checkquote{%
5022 \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
5023 \noexpand\else
5024 \noexpand\def\noexpand\@gls@checkquote{%
5025 \noexpand\@gls@checkquote####2#1####3\noexpand\null}%
5026 \noexpand\fi
5027 \noexpand\fi
5028 \noexpand\@gls@checkquote
5029 }%
5030 }%
5031 \@gls@docheckquotedef
5032 \edef\@gls@docheckquotedef{%
5033 \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
5034 \noexpand\def\noexpand\@gls@checkedmkidx{%
5035 \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
5036 #1#1\noexpand\null
5037 \noexpand\expandafter\noexpand\@gls@updatechecked
5038 \noexpand\@gls@checkedmkidx{####1}%
5039 \noexpand\def\noexpand\@gls@checkedmkidx{%
5040 \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
5041 \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
5042 \noexpand\null
5043 \noexpand\expandafter\noexpand\@gls@updatechecked
5044 \noexpand\@gls@checkedmkidx{####1}%
5045 \noexpand\def\noexpand\@gls@checkedmkidx{%
5046 \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
5047 \noexpand\?\noexpand\?\noexpand\null
5048 \noexpand\expandafter\noexpand\@gls@updatechecked
5049 \noexpand\@gls@checkedmkidx{####1}%
5050 \noexpand\def\noexpand\@gls@checkedmkidx{%
5051 \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
5052 \noexpand?\noexpand?\noexpand\null
5053 \noexpand\expandafter\noexpand\@gls@updatechecked
5054 \noexpand\@gls@checkedmkidx{####1}%
5055 \noexpand\def\noexpand\@gls@checkedmkidx{%
5056 \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil

```

```

5057 \noexpand|\noexpand|\noexpand\null
5058 \noexpand\expandafter\noexpand\@gls@updatechecked
5059 \noexpand\@gls@checkedmkidx{####1}%
5060 \noexpand\def\noexpand\@gls@checkedmkidx{%
5061 \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
5062 \noexpand\\ \noexpand\\ \noexpand\null
5063 \noexpand\expandafter\noexpand\@gls@updatechecked
5064 \noexpand\@gls@checkedmkidx{####1}%
5065 \noexpand\def\noexpand\@gls@checkedmkidx{%
5066 \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
5067 \noexpand!\noexpand!\noexpand\null
5068 \noexpand\expandafter\noexpand\@gls@updatechecked
5069 \noexpand\@gls@checkedmkidx{####1}%
5070 }%
5071 }%
5072 \@gls@docheckquotedef
5073 \edef\@gls@docheckquotedef{%
5074 \noexpand\def\noexpand\@gls@checkescquote####1%
5075 \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
5076 ####3\noexpand\null{%
5077 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5078 \noexpand\toks@={####1}%
5079 \noexpand\ifx\noexpand\null####2\noexpand\null
5080 \noexpand\ifx\noexpand\null####3\noexpand\null
5081 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5082 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5083 \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
5084 \noexpand\else
5085 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5086 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5087 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5088 \csname#1\endcsname}\noexpand\@gls@quotechar
5089 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5090 \csname#1\endcsname}\noexpand\@gls@quotechar}%
5091 \noexpand\def\noexpand\@gls@checkescquote{%
5092 \noexpand\@gls@checkescquote####3\noexpand\null}%
5093 \noexpand\fi
5094 \noexpand\else
5095 \noexpand\edef\noexpand\@gls@checkedmkidx{%
5096 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5097 \noexpand\@gls@quotechar\noexpand\string
5098 \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
5099 \noexpand\ifx\noexpand\null####3\noexpand\null
5100 \noexpand\def\noexpand\@gls@checkescquote{%
5101 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5102 \expandonce{\csname#1\endcsname}\noexpand\null}%
5103 \noexpand\else
5104 \noexpand\def\noexpand\@gls@checkescquote{%
5105 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%

```

```

5106      #####3\noexpand\null}%
5107      \noexpand\fi
5108      \noexpand\fi
5109      \noexpand\@@gls@checkescquote
5110    }%
5111  }%
5112  \@gls@docheckquotedef
5113 }
5114 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
5115   {\string\GlsSetQuote\space not permitted here}%
5116   {Move \string\GlsSetQuote\space earlier in the preamble, as
5117    soon as possible after glossaries.sty has been loaded}}
5118 \fi

```

ramakeindexopts

```

5119 \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`

```

5120 \newcommand{\noist}{%
  Update attributes list
5121  \@gls@addpredefinedattributes
5122  \let\writeist\relax
5123 }

```

`\@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`

```

5124 \newcommand*{\@makeglossary}[1]{%
5125   \ifglossaryexists{#1}%
5126   {%

```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```

5127   \ifglssavewrites
5128     \expandafter\newtoks\csname glo@#1@filetok\endcsname
5129   \else
5130     \expandafter\newwrite\csname glo@#1@file\endcsname

```

```

5131      \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
5132      \fi
5133      \@gls@renewglossary
5134      \writeist
5135  }%
5136  {%
5137      \PackageError{glossaries}%
5138      {Glossary type ‘#1’ not defined}%
5139      {New glossaries must be defined before using \string\makeglossaries}%
5140  }%
5141 }

```

`\@glsopenfile` Open write file associated with the given glossary.

```

5142 \newcommand*{\@glsopenfile}[2]{%
5143   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
5144   \PackageInfo{glossaries}{Writing glossary file
5145     \jobname.\csname @glotype@#2@out\endcsname}%
5146 }

```

`\@closegls`

```

5147 \newcommand*{\@closegls}[1]{%
5148   \closeout\csname glo@#1@file\endcsname
5149 }

```

`\@gls@automake`

```

5150 \ifglxindy
5151 \newcommand*{\@gls@automake}[1]{%
5152   \ifglossaryexists{#1}
5153   {%
5154     \@closegls{#1}%
5155     \ifdefstring{\glsorder}{letter}%
5156     {\def\@gls@order{-M ord/letorder }}%
5157     {\let\@gls@order\@empty}%
5158     \ifcsundef{@xdy@#1@language}%
5159     {\let\@gls@langmod\@xdy@main@language}%
5160     {\letcs\@gls@langmod{@xdy@#1@language}}%
5161     \edef\@gls@dothiswrite{\noexpand\write18{xindy
5162       -I xindy
5163       \@gls@order
5164       -L \@gls@langmod\space
5165       -M \@gls@istfilebase\space
5166       -C \@gls@codepage\space
5167       -t \jobname.\csuse{@glotype@#1@log}
5168       -o \jobname.\csuse{@glotype@#1@in}
5169       \jobname.\csuse{@glotype@#1@out}}%
5170     }%
5171     \@gls@dothiswrite
5172   }%
5173   {%

```

```

5174     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5175 }%
5176 }
5177 \else
5178 \newcommand*{\@gls@automake}[1]{%
5179   \ifglossaryexists{#1}
5180   {%
5181     \@closegls{#1}%
5182     \ifdefstring{\glsorder}{letter}%
5183     {\def\@gls@order{-l }}%
5184     {\let\@gls@order\@empty}%
5185     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5186       -s \istfilename\space
5187       -t \jobname.\csuse{\glotype@#1@log}
5188       -o \jobname.\csuse{\glotype@#1@in}
5189       \jobname.\csuse{\glotype@#1@out}}}%
5190   }%
5191   \@gls@dothiswrite
5192 }%
5193 {%
5194   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5195 }%
5196 }
5197 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

5198 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

5199 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}

```

`\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`

```

5200 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

5201 \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.

```

5202 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{} }
5203 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{} }

```

If `\@gls@extramakeindexopts` has been defined, write it:

```

5204 \ifundef\@gls@extramakeindexopts
5205 {}%
5206 {%
5207   \protected@write\@auxout{}{\string\providecommand

```

```

5208     \string\@gls@extramakeindexopts[1]{}}
5209     \protected@write\@auxout{}{\string\@gls@extramakeindexopts
5210     {\@gls@extramakeindexopts}}%
5211 }%

```

Write the name of the style file to the aux file (needed by makeglossaries)

```

5212 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
5213 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```

5214 \@for\@glo@type:=\@glo@types\do{%
5215     \ifthenelse{equal{\@glo@type}{}}{\}%
5216     \@makeglossary{\@glo@type}}%
5217 }%

```

New glossaries must be created before \makeglossaries so disable \newglossary.

```

5218 \renewcommand*\newglossary[4][]{%
5219 \PackageError{glossaries}{New glossaries
5220 must be created before \string\makeglossaries}{You need
5221 to move \string\makeglossaries\space after all your
5222 \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect. The deprecated \makeglossary is not redefined here as it either implements \makeglossaries or has been restored to its original definition (in which case it shouldn't be changed).

```

5223 \let\@makeglossary\relax
5224 \let\makeglossaries\relax

```

Disable all commands that have no effect after \makeglossaries

```

5225 \@disable@onlypremakeg

```

Allow see key:

```

5226 \let\gls@checkseeallowed\relax

```

Suppress warning about no \makeglossaries

```

5227 \let\warn@nomakeglossaries\relax

```

Activate warning about missing \printglossary

```

5228 \def\warn@noprintglossary{%
5229     \ifdefstring{\@glo@types}{,}%
5230     {%
5231         \GlossariesWarningNoLine{No glossaries have been defined}%
5232     }%
5233     {%
5234         \GlossariesWarningNoLine{No \string\printglossary\space
5235         or \string\printglossaries\space
5236         found. ^^J(Remove \string\makeglossaries\space if you
5237         don't want any glossaries.) ^^JThis document will not
5238         have a glossary}%
5239     }%
5240 }%

```

Declare list parser for `\glsdisplaynumberlist`

```
5241 \ifglssavenumberlist
5242 \edef\@gls@ddeflistparser{\noexpand\DeclareListParser
5243   {\noexpand\glsnumlistparser}{\delimN}}%
5244 \@gls@ddeflistparser
5245 \fi
```

Prevent user from also using `\makenoidxglossaries`

```
5246 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in `\printgloss` family:

```
5247 \renewcommand*{\@printgloss@setsort}{%
5248   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5249 }%
```

Check the automake setting:

```
5250 \ifglssautomake
5251 \renewcommand*{\@gls@doautomake}{%
5252   \@for\@gls@type:=\@glo@types\do{%
5253     \ifdefempty{\@gls@type}{}%
5254     {\@gls@automake{\@gls@type}}%
5255   }%
5256 }%
5257 \fi
```

Check the sort setting:

```
5258 \@glo@check@sortallowed\makeglossaries
5259 }
```

Must occur in the preamble:

```
5260 \@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.

```
5261 \AtEndDocument{%
5262   \warn@nomakeglossaries
5263   \warn@noprintglossary
5264 }
```

`noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
5265 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5266 \renewcommand{\@gls@noref@warn}[1]{%
5267   \GlossariesWarning{Empty glossary for
5268   \string\printnoidxglossary[type={##1}].
5269   Rerun may be required (or you may have forgotten to use
5270   commands like \string\gls)}%
5271 }
```


Don't escape makeindex/xindy characters:

```
5272 \let\@gls@checkmkidxchars\@gobble
```

Don't escape locations:

```
5273 \glsesclocationsfalse
```

Write glossary information to aux instead of glossary files

```
5274 \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5275 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5276 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5277 \renewcommand{\@do@seeglossary}[2]{%
5278   \edef\@gls@label{\glsdetoklabel{##1}}%
5279   \protected@write\@auxout{}{%
5280     \string\@gls@reference
5281     {\csname glo@\@gls@label @type\endcsname}%
5282     {\@gls@label}%
5283     {%
5284       \string\glsseeformat##2}%
5285     }%
5286   }%
5287 }%
```

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.

```
5288 \AtBeginDocument
5289 {%
5290   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5291 }%
```

Change warning about no glossaries

```
5292 \def\warn@noprintglossary{%
5293   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5294     or \string\printnoidxglossaries ^^J
5295     found. (Remove \string\makenoidxglossaries\space if you
5296     don't want any glossaries.)^^JThis document will not have a glossary}%
5297 }%
```

Suppress warning about no \makeglossaries

```
5298 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5299 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5300 \renewcommand*{\@printgloss@setsort}{%
5301   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
5302 \def\@glo@sorttype{\@glo@default@sorttype}%
5303 }%
```

All entries must be defined in the preamble:

```
5304 \renewcommand*\new@glossaryentry[2]{%
5305 \PackageError{glossaries}{Glossary entries must be
5306 defined in the preamble^^Jwhen you use
5307 \string\makenoidxglossaries}%
5308 {Either move your definitions to the preamble or use
5309 \string\makeglossaries}%
5310 }%
```

Redefine \glsentrynumberlist

```
5311 \renewcommand*\glsentrynumberlist[1]{%
5312 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5313 \ifdef\@gls@loclist
5314 {%
5315 \glsnoidxloclist{\@gls@loclist}%
5316 }%
5317 {%
5318 ??\glsdoifexists{##1}%
5319 {%
5320 \GlossariesWarning{Missing location list for ‘##1’. Either
5321 a rerun is required or you haven’t referenced the entry}%
5322 }%
5323 }%
5324 }%
```

Redefine \glsdisplaynumberlist

```
5325 \renewcommand*\glsdisplaynumberlist[1]{%
5326 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5327 \ifdef\@gls@loclist
5328 {%
5329 \def\@gls@noidxloclist@sep{%
5330 \def\@gls@noidxloclist@sep{%
5331 \def\@gls@noidxloclist@sep{%
5332 \glsnumlistsep
5333 }%
5334 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5335 }%
5336 }%
5337 \def\@gls@noidxloclist@finalsep{}}%
5338 \def\@gls@noidxloclist@prev{}}%
5339 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5340 \@gls@noidxloclist@finalsep
5341 \@gls@noidxloclist@prev
5342 }%
5343 {%
5344 ??\glsdoifexists{##1}%
```

```

5345     {%
5346         \GlossariesWarning{Missing location list for ‘##1’. Either
5347             a rerun is required or you haven’t referenced the entry}%
5348     }%
5349 }%
5350 }%

```

Provide a generic way of iterating through the number list:

```

5351 \renewcommand*{\glsnumberlistloop}[3]{%
5352     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5353     \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5354     \let\@gls@org@glsseeformat\glsseeformat
5355     \let\glsnoidxdisplayloc##2\relax
5356     \let\glsseeformat##3\relax
5357     \ifdef\@gls@loclist
5358     {%
5359         \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5360     }%
5361     {%
5362         ??\glsdoifexists{##1}%
5363         {%
5364             \GlossariesWarning{Missing location list for ‘##1’. Either
5365                 a rerun is required or you haven’t referenced the entry}%
5366         }%
5367     }%
5368     \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5369     \let\glsseeformat\@gls@org@glsseeformat
5370 }%

```

Modify sanitize sort function

```

5371 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5372 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5373 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5374 \@glo@check@sortallowed\makenoidxglossaries
5375 }

```

Preamble-only command:

```

5376 \@onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

5377 \newcommand*{\glsnumberlistloop}[2]{%
5378     \PackageError{glossaries}{\string\glsnumberlistloop\space
5379         only works with \string\makenoidxglossaries}{}%
5380 }

```

`\listloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc {<prefix>}{<counter>}{<format>}{<n>}`)

```

5381 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5382   #1%
5383 }

```

@makeglossaries Can't use both `\makeglossaries` and `\makenoidxglossaries`

```

5384 \newcommand*{\@no@makeglossaries}{%
5385   \PackageError{glossaries}{You can't use both
5386   \string\makeglossaries\space and \string\makenoidxglossaries}%
5387   {Either use one or other (or none) of those commands but not both
5388   together.}%
5389 }

```

@gls@noref@warn Warning when no instances of `\@gls@reference` found.

```

5390 \newcommand{\@gls@noref@warn}[1]{%
5391   \GlossariesWarning{\string\makenoidxglossaries\space
5392   is required to make \string\printnoidxglossary[type={#1}] work}%
5393 }

```

1.14 Writing information to associated files

s@noidxglossary Write the glossary information to the aux file (for the ‘noidx’ method):

```

5394 \newcommand*{\gls@noidxglossary}{%
5395   \protected@write\@auxout{}{%
5396     \string\@gls@reference
5397     {\csname glo@\@gls@label @type\endcsname}%
5398     {\@gls@label}%
5399     {\string\glsnoidxdisplayloc
5400      {\@glo@counterprefix}%
5401      {\@gls@counter}%
5402      {\@glsnumberformat}%
5403      {\@glslocref}%
5404     }%
5405   }%
5406 }

```

\istfile Deprecated.

```

5407 \providecommand\istfile{\glswrite}

```

At the end of the document, the files should be created if `savewrites=true`.

```

5408 \AtEndDocument{%
5409   \glswritefiles
5410 }

```

\@glswritefiles Only write the files if `savewrites=true`.

```

5411 \newcommand*{\@glswritefiles}{%
  Iterate through all the glossaries.
5412   \forallglossaries{\@glo@type}{%

```

Check for empty glossaries (patch provided by Patrick Häcker)

```

5413 \ifcsundef{glo@\@glo@type @filetok}%
5414 {%
5415   \def\gls@tmp{}%
5416 }%
5417 {%
5418   \edef\gls@tmp{\expandafter\the
5419     \csname glo@\@glo@type @filetok\endcsname}%
5420 }%
5421 \ifx\gls@tmp\@empty
5422   \ifx\@glo@type\glsdefaulttype
5423     \GlossariesWarningNoLine{Glossary '\@glo@type' has no
5424       entries.^^JRemember to use package option 'nomain' if
5425 you
5426       don't want to^^Juse the main glossary}%
5427   \else
5428     \GlossariesWarningNoLine{Glossary '\@glo@type' has no
5429       entries}%
5430   \fi
5431 \else
5432   \@glsopenfile{\glswrite}{\@glo@type}%
5433   \immediate\write\glswrite{%
5434     \expandafter\the
5435       \csname glo@\@glo@type @filetok\endcsname}%
5436   \immediate\closeout\glswrite
5437 \fi
5438 }%
5439 }

```

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`

```

5440 \newcommand*{\gls@glossary}[1]{%
5441   \@gls@glossary{#1}%
5442 }

```

`\@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 `\glspersentchar` would be needed with the sort sanitization switched off.) This means that it's safe to simply ignore the second argument.

```
5443 \newcommand*{\@gls@glossary}[2]{%
5444   \if@gls@debug
5445     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5446   \fi
5447 }
```

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`

```
5448 \newcommand{\@gls@renewglossary}{%
5449   \gdef\@gls@glossary##1{\@bsphack\beginingroup\gls@wrglossary{##1}}%
5450   \let\@gls@renewglossary\@empty
5451 }
```

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`

```
5452 \newcommand*{\gls@wrglossary}[2]{%
5453   \ifglssavewrites
5454     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5455     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5456       \expandafter{\@gls@tmp^^J}%
5457   \else
5458     \ifcsdef{glo@#1@file}%
5459       {%
5460         \expandafter\protected@write\csname glo@#1@file\endcsname{%
5461           \gls@disablepagerefexpansion}{#2}%
5462       }%
5463     {%
5464       \ifignoredglossary{#1}{}%
5465     }
```

```

5466      \GlossariesWarning{No file defined for glossary ‘#1’}%
5467    }%
5468  }%
5469  \fi
5470  \endgroup\@esphack
5471 }

```

\do@wrglossary

```

5472 \newcommand*{\do@wrglossary}[1]{%
5473   \glswriteentry{#1}{\do@wrglossary{#1}}%
5474 }

```

\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.

```

5475 \newcommand*{\glswriteentry}[2]{%
5476   \ifglsexindexonlyfirst
5477     \ifglssused{#1}{#2}%
5478   \else
5479     #2%
5480   \fi
5481 }

```

protected@pagefmts List of page formats to be protected against expansion.

```

5482 \newcommand{\gls@protected@pagefmts}{\gls@numberpage,\gls@alphpage,%
5483   \gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage}

```

pagerefexpansion

```

5484 \newcommand*{\gls@disablepagerefexpansion}{%
5485   \@for\@gls@this:=\gls@protected@pagefmts\do
5486   {%
5487     \expandafter\let\@gls@this\relax
5488   }%
5489 }

```

\gls@alphpage

```

5490 \newcommand*{\gls@alphpage}{\@alph\c@page}

```

\gls@Alphpage

```

5491 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

```

\gls@numberpage

```

5492 \newcommand*{\gls@numberpage}{\number\c@page}

```

\gls@arabicpage

```

5493 \newcommand*{\gls@arabicpage}{\@arabic\c@page}

```

`\gls@romanpage`

```
5494 \newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

`\gls@Romanpage`

```
5495 \newcommand*{\gls@Romanpage}{\@Roman\c@page}
```

`protectedpagefmt`

```
\glsaddprotectedpagefmt{<cs name>}
```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument (`\<csname>\c@page` must be valid).

```
5496 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5497   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
5498   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5499   \eappto\@wrglossarynumberhook{%
5500     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5501     \expandonce{\csname#1\endcsname}%
5502     \noexpand\def\expandonce{\csname#1\endcsname}{%
5503       \noexpand\@wrglossary@pageformat
5504       \expandonce{\csname gls#1page\endcsname}%
5505       \expandonce{\csname org@gls#1\endcsname}%
5506     }%
5507   }%
5508 }
```

`ssarynumberhook` Hook used by `\@do@wrglossary`

```
5509 \newcommand*{\@wrglossarynumberhook{}
```

`sary@pageformat`

```
5510 \newcommand{\@wrglossary@pageformat}[3]{%
5511   \ifx#3\c@page #1\else #2#3\fi
5512 }
```

`@@do@wrglossary` Write the glossary entry in the appropriate format.

```
5513 \newcommand*{\@@do@wrglossary}[1]{%
5514   \ifglsecllocations
5515     \@@do@esc@wrglossary{#1}%
5516   \else
5517     \@@do@noesc@wrglossary{#1}%
5518   \fi
5519 }
```

`oesc@wrglossary` Write the glossary entry in the appropriate format. The locations don't need to be pre-processed before writing the information to the glossary file, but the prefix still needs to be found.

```
5520 \newcommand*{\@@do@noesc@wrglossary}[1]{%
```


Don't fully expand yet.

```
5521 \expandafter\def\expandafter\@glslocref\expandafter{\theglsentrycounter}%
5522 \expandafter\def\expandafter\@glsHlocref\expandafter{\theHglentrycounter}%
```

Find the prefix if `\@glsHlocref` and `\@glslocref` aren't the same.

```
5523 \ifx\@glsHlocref\@glslocref
5524   \def\@glo@counterprefix{}%
5525 \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 `\theglsentrycounter` and `\theHglentrycounter` at this point, even if it hasn't been updated yet. The page number is not expected to occur in the prefix.)

```
5526   \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5527     {\@glslocref}{\@glsHlocref}}%
5528   }%
5529   \@do@gls@getcounterprefix
5530 \fi
```

De-tok label if required.

```
5531 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5532 \@@do@@wrglossary
5533 }
```

`\owprimitivemods` Conditional to determine whether or not `\@@do@esc@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
5534 \newif\ifglswrallowprimitivemods
5535 \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).

```
5536 \newcommand*{\@@do@esc@wrglossary}[1]{% please read documented code!
5537 \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions (scoped):

```
5538 \let\gls@orgthe\the
5539 \let\gls@orgnumber\number
5540 \let\gls@orgarabic\@arabic
5541 \let\gls@orgromannumeral\romannumeral
5542 \let\gls@orgalph\@alph
5543 \let\gls@orgAlph\@Alph
5544 \let\gls@orgRoman\@Roman
```

Redefine:

```
5545 \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.

```

5546 \def\gls@the##1{%
5547 \ifx##1\c@page \gls@numberpage\else\gls@orgthe##1\fi}%
5548 \def\the{\expandafter\gls@the}%
5549 \def\gls@number##1{%
5550 \ifx##1\c@page \gls@numberpage\else\gls@orgnumber##1\fi}%
5551 \def\number{\expandafter\gls@number}%
5552 \fi
5553 \def\@arabic##1{%
5554 \ifx##1\c@page \gls@arabicpage\else\gls@orgarabic##1\fi}%
5555 \def\romannumeral##1{%
5556 \ifx##1\c@page \gls@romanpage\else\gls@orgromannumeral##1\fi}%
5557 \def\@Roman##1{%
5558 \ifx##1\c@page \gls@Romanpage\else\gls@orgRoman##1\fi}%
5559 \def\@alph##1{%
5560 \ifx##1\c@page \gls@alphpage\else\gls@orgalph##1\fi}%
5561 \def\@Alph##1{%
5562 \ifx##1\c@page \gls@Alphpage\else\gls@orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```

5563 \@wrglossarynumberhook

```

Prevent expansion:

```

5564 \gls@disablepagerefexpansion

```

Now store location in `\@glslocref`:

```

5565 \protected@xdef\@glslocref{\theHglentrycounter}%
5566 \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.

```

5567 \@gls@checkmkidxchars\@glslocref

```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

5568 \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5569 \def\@glo@counterprefix{}%
5570 \else
5571 \protected@edef\@glsHlocref{\theHglentrycounter}%
5572 \@gls@checkmkidxchars\@glsHlocref
5573 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5574 {\@glslocref}{\@glsHlocref}%
5575 }%
5576 \@do@gls@getcounterprefix
5577 \fi

```

De-tok label if required

```

5578 \edef\@gls@label{\glsdetoklabel{#1}}%

```

Write the information to file:

```

5579 \@@do@@wrglossary
5580 }

```

@@do@@wrglossary

```

5581 \newcommand*{\@@do@@wrglossary}{%

```

Determine whether to use xindy or makeindex syntax

```

5582 \ifglsxindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5583 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5584 \def\@glo@range{}%
5585 \expandafter\if\@glo@prefix(\relax
5586 \def\@glo@range{:open-range}%
5587 \else
5588 \expandafter\if\@glo@prefix)\relax
5589 \def\@glo@range{:close-range}%
5590 \fi
5591 \fi

```

Write to the glossary file using xindy syntax.

```

5592 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5593 (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)

5594 :locref \string"\@glo@counterprefix}\@gls@locref}\string" %
5595 :attr \string"\@gls@counter\@glo@suffix\string"
5596 \@glo@range
5597 )
5598 }%
5599 \else

```

Convert the format information into the format required for makeindex

```

5600 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5601 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5602 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5603 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5604 \@gls@encapchar\@glo@numfmt}{\@gls@locref}}%
5605 \fi
5606 }

```

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.)

```

5607 \newcommand*\@gls@getcounterprefix[2]{%
5608 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5609 \ifx\@gls@thisloc\@gls@thisHloc
5610 \def\@glo@counterprefix{}%

```

```

5611 \else
5612   \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5613     \def\@glo@tmp{##2}%
5614     \ifx\@glo@tmp\@empty
5615       \def\@glo@counterprefix{%
5616     \else
5617       \def\@glo@counterprefix{##1}%
5618     \fi
5619   }%
5620   \@gls@get@counterprefix#2.#1\end@getprefix

Warn if no prefix can be formed.

5621   \ifx\@glo@counterprefix\@empty
5622     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5623       prefixing^^Jlocation ‘#1’. You need to modify the
5624       definition of \string\theH\@gls@counter^^Jotherwise you
5625       will get the warning: “name{\@gls@counter.#1}’ has been^^J
5626       referenced but does not exist"}%
5627   \fi
5628 \fi
5629 }

```

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.

```

5630 \newcommand{\@do@seeglossary}[2]{%
5631 \def\@gls@xref{#2}%
5632 \@onelevel@sanitize\@gls@xref
5633 \@gls@checkmkidxchars\@gls@xref
5634 \ifglxindy
5635   \gls@glossary{\csname glo@#1@type\endcsname}{%
5636     (indexentry
5637       :tkey (\csname glo@#1@index\endcsname)
5638       :xref (\string"\@gls@xref\string")
5639       :attr \string"see\string"
5640     )
5641   }%
5642 \else
5643   \gls@glossary{\csname glo@#1@type\endcsname}{%
5644     \string\glossaryentry{\csname glo@#1@index\endcsname
5645       \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5646 \fi
5647 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5648 \def\@gls@fixbraces#1#2#3\@nil{%

```

```

5649 \ifx#2[\relax
5650 \@@gls@fixbraces#1#2#3\@end@fixbraces
5651 \else
5652 \def#1{{#2#3}}%
5653 \fi
5654 }

```

`\@@gls@fixbraces`

```

5655 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5656 \def#1{[#2]{#3}}%
5657 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

5658 \DeclareRobustCommand*\glssee[3][\seename]{%
5659 \do@seeglossary{#2}{#1}{#3}}
5660 \newcommand*\@glssee[3][\seename]{%
5661 \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.

```

5662 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5663 \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

5664 \DeclareRobustCommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

5665 \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

5666 \let\@gls@donext\relax

```

Iterate through the labels

```

5667 \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5668 \ifx\@xfor@nextelement\@nnil
5669 \@gls@dolast
5670 \else
5671 \@gls@donext
5672 \fi

```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```

5673 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

```

Update separators

```

5674 \let\@gls@dolast\glsseelastsep
5675 \let\@gls@donext\glsseesep
5676 }%
5677 }

```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5678 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
5679 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5680 \DeclareRobustCommand*{\glsseeitem}[1]{\glshyperlink[\glsseeitemformat{#1}]{#1}}
```

`lsseeitemformat` 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.)

```
5681 \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.

```
5682 \newcommand*{\gls@save@numberlist}[1]{%
5683   \ifglssavenumberlist
5684     \toks@{#1}%
5685     \edef\@do@writeaux@info{%
5686       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5687     }%
5688     \@onelevel@sanitize\@do@writeaux@info
5689     \protected@write\@auxout{}\@do@writeaux@info}%
5690   \fi
5691 }
```

`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.)

```
5692 \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.

```
5693 \ifcsundef{printglossary}{}%
5694 {%
  If \printglossary is already defined, issue a warning and undefine it.
5695   \@gls@warnonglossdefined
5696   \undef\printglossary
5697 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5698 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5699   \@printglossary{#1}{\@print@glossary}%
5700 }
```

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`

```
5701 \newcommand*{\printglossaries}{%
5702   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5703 }
```

`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.

```
5704 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5705   \@printglossary{#1}{\@print@noidx@glossary}%
5706 }
```

`printnoidxglossaries` Analogous to `\printglossaries`

```
5707 \newcommand*{\printnoidxglossaries}{%
5708   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5709 }
```

`printgloss@setsort` Initialise to do nothing.

```
5710 \newcommand*{\@printgloss@setsort}{}%
```

`preglossaryhook`

```
5711 \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.

```
5712 \newcommand{\@printglossary}[2]{%
5713   Set up defaults.
```

```
5713   \def\@glo@type{\glsdefaulttype}%
5714   \def\glossarytitle{\csname @glo@type @title\endcsname}%
5715   \def\glossarytoctitle{\glossarytitle}%
5716   \let\org@glossarytitle\glossarytitle
```

```
5713   \def\@glo@type{\glsdefaulttype}%
5714   \def\glossarytitle{\csname @glo@type @title\endcsname}%
5715   \def\glossarytoctitle{\glossarytitle}%
5716   \let\org@glossarytitle\glossarytitle
```

```
5713   \def\@glo@type{\glsdefaulttype}%
5714   \def\glossarytitle{\csname @glo@type @title\endcsname}%
5715   \def\glossarytoctitle{\glossarytitle}%
5716   \let\org@glossarytitle\glossarytitle
```

```
5715   \def\glossarytoctitle{\glossarytitle}%
5716   \let\org@glossarytitle\glossarytitle
```

```
5716   \let\org@glossarytitle\glossarytitle
```



```

5717 \def\@glossarystyle{%
5718   \ifx\@glossary@default@style\relax
5719     \GlossariesWarning{No default glossary style provided \MessageBreak
5720       for the glossary '@glo@type'. \MessageBreak
5721       Using deprecated fallback. \MessageBreak
5722       To fix this set the style with \MessageBreak
5723       \string\setglossarystyle\space or use the \MessageBreak
5724       style key=value option}%
5725   \fi
5726 }%
5727 \def\gls@dotoc{title{\glssettoc{title{\@glo@type}}}%

Store current value of \glossaryentrynumbers. (This may be changed via the optional argument)
5728 \let\org@glossaryentrynumbers\glossaryentrynumbers

Localise the effects of the optional argument
5729 \bgroup

Activate or deactivate sort key:
5730 \@printgloss@setsort

Determine settings specified in the optional argument.
5731 \setkeys{printgloss}{#1}%

Does the glossary exist?
5732 \ifglossaryexists{\@glo@type}%
5733 {%

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)
5734 \ifx\glossarytitle\org@glossarytitle
5735 \else
5736   \expandafter\let\csname @glo@type @title\endcsname
5737     \glossarytitle
5738 \fi

Allow a high-level user command to indicate the current glossary
5739 \let\currentglossary\@glo@type

Enable individual number lists to be suppressed.
5740 \let\org@glossaryentrynumbers\glossaryentrynumbers
5741 \let\glsnonextpages\@glsnonextpages

Enable individual number list to be activated:
5742 \let\glsnextpages\@glsnextpages

Enable suppression of description terminators.
5743 \let\nopostdesc\@nopostdesc

Set up the entry for the TOC
5744 \gls@dotoc{title

```

Set the glossary style

```
5745 \org@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):

```
5746 \let\gls@org@glossaryentryfield\glossentry
5747 \let\gls@org@glossarysubentryfield\subglossentry
5748 \renewcommand{\glossentry}[1]{%
5749   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5750   \gls@org@glossaryentryfield{##1}%
5751 }%
5752 \renewcommand{\subglossentry}[2]{%
5753   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5754   \gls@org@glossarysubentryfield{##1}{##2}%
5755 }%
```

```
5756 \gls@preglossaryhook
```

Now do the handler macro that deals with the actual glossary:

```
5757 #2%
5758 }%
5759 {\GlossariesWarning{Glossary ‘\org@type’ doesn’t exist}}%
```

End the current scope

```
5760 \egroup
```

Reset `\glossaryentrynumbers`

```
5761 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5762 \global\let\warn@noprntglossary\relax
5763 }
```

`@print@glossary` Internal workings of `\printglossary` dealing with reading the external file.

```
5764 \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.)

```
5765 \makeatletter
```

Input the glossary file, if it exists.

```
5766 \@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.

```
5767 \IfFileExists{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
5768 {}%
5769 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5770 \ifglxindy
5771 \ifcsundef{@xdy@\@glo@type @language}%
5772 {%
5773 \edef\@do@auxoutstuff{%
5774 \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.

```
5775 \noexpand\immediate\noexpand\write\@auxout{%
5776 \string\providecommand\string\@xdylanguage[2]{}}%
5777 \noexpand\immediate\noexpand\write\@auxout{%
5778 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5779 }%
5780 }%
5781 }%
5782 {%
5783 \edef\@do@auxoutstuff{%
5784 \noexpand\AtEndDocument{%
5785 \noexpand\immediate\noexpand\write\@auxout{%
5786 \string\providecommand\string\@xdylanguage[2]{}}%
5787 \noexpand\immediate\noexpand\write\@auxout{%
5788 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5789 @language\endcsname}}}%
5790 }%
5791 }%
5792 }%
5793 \@do@auxoutstuff
5794 \edef\@do@auxoutstuff{%
5795 \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.

```
5796 \noexpand\immediate\noexpand\write\@auxout{%
5797 \string\providecommand\string\@gls@codepage[2]{}}%
5798 \noexpand\immediate\noexpand\write\@auxout{%
5799 \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
5800 }%
5801 }%
5802 \@do@auxoutstuff
5803 \fi
```

Activate warning if \makeglossaries hasn't been used.

```
5804 \renewcommand*{\@warn@nomakeglossaries}{%
5805 \GlossariesWarningNoLine{\string\makeglossaries\space
5806 hasn't been used,^^Jthe glossaries will not be updated}%
5807 }%
5808 }
```

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 $\backslash@glsref@<type>$). The actual sorting is done by $\backslash@glo@sortentries\{<handler>\}\{<type>\}$.

$\backslash@glo@sortentries$

```

5809 \newcommand*\@glo@sortentries[2]{%
5810   \glosortentrieswarning
5811   \def\@glo@sortinglist{}%
5812   \def\@glo@sortinghandler{#1}%
5813   \edef\@glo@type{#2}%
5814   \forlistcsloop{\@glo@do@sortentries}\@glsref@#2}%
5815   \csdef{\@glsref@#2}{}%
5816   \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5817   \xifinlistcs{\@this@label}\@glsref@#2}%
5818   {}%
5819   {%
5820     \listcsxadd{\@glsref@#2}\@this@label}%
5821   }%
5822   \ifcsdef{\@glo@sortingchildren@\@this@label}%
5823   {%
5824     \@glo@addchildren{#2}\@this@label}%
5825   }%
5826   {}%
5827 }%
5828 }

```

$\backslash@glo@addchildren$

$\backslash@glo@addchildren\{<type>\}\{<parent>\}$

```

5829 \newcommand*\@glo@addchildren[2]{%

```

Scope to allow nesting.

```

5830   \bgroup
5831     \letcs{\@glo@childlist}\@glo@sortingchildren@#2}%
5832     \@for\@this@childlabel:=\@glo@childlist\do
5833     {%

```

Check this label hasn't already been added.

```

5834     \xifinlistcs{\@this@childlabel}\@glsref@#1}%
5835     {}%
5836     {%
5837       \listcsxadd{\@glsref@#1}\@this@childlabel}%
5838     }%

```

Does this child have children?

```

5839      \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5840      {%
5841          \@glo@addchildren{#1}{\@this@childlabel}%
5842      }%
5843      {%
5844      }%
5845      }%
5846  \egroup
5847 }
```

@do@sortentries

```

5848 \newcommand*{\@glo@do@sortentries}[1]{%
5849   \ifglshasparent{#1}%
5850   {%
```

This entry has a parent, so add it to the child list

```

5851     \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
5852     \ifcsundef{@glo@sortingchildren@\@glo@parent}%
5853     {%
5854         \csdef{@glo@sortingchildren@\@glo@parent}{}%
5855     }%
5856     {}%
5857     \expandafter\@glo@sortedinsert
5858     \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```

5859     \xifinlistcs{\@glo@parent}{@glstref@\@glo@type}%
5860     {%
```

Yes, it has so do nothing.

```

5861     }%
5862     {%
```

No, it hasn't so add it now.

```

5863         \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5864     }%
5865     }%
5866     {%
5867         \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5868     }%
5869 }
```

glo@sortedinsert

$\backslash@glo@sortedinsert\{\langle list \rangle\}\{\langle entry\ label \rangle\}$

Insert into list.

```

5870 \newcommand*{\@glo@sortedinsert}[2]{%
5871   \dtlinsertinto{#2}{#1}{\@glo@sortinghandler}%
5872 }%
```

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 $\#2$).

`sorthandler@word`

```
5873 \newcommand*{\@glo@sorthandler@word}[2]{%
5874   \letcs\@gls@sort@A{glo\glsdetoklabel{\#1}@sort}%
5875   \letcs\@gls@sort@B{glo\glsdetoklabel{\#2}@sort}%
5876   \edef\glo@do@compare{%
5877     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5878     {\expandonce\@gls@sort@B}%
5879     {\expandonce\@gls@sort@A}%
5880   }%
5881   \glo@do@compare
5882 }
```

`thandler@letter`

```
5883 \newcommand*{\@glo@sorthandler@letter}[2]{%
5884   \letcs\@gls@sort@A{glo\glsdetoklabel{\#1}@sort}%
5885   \letcs\@gls@sort@B{glo\glsdetoklabel{\#2}@sort}%
5886   \edef\glo@do@compare{%
5887     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5888     {\expandonce\@gls@sort@B}%
5889     {\expandonce\@gls@sort@A}%
5890   }%
5891   \glo@do@compare
5892 }
```

`sorthandler@case` Case-sensitive sort.

```
5893 \newcommand*{\@glo@sorthandler@case}[2]{%
5894   \letcs\@gls@sort@A{glo\glsdetoklabel{\#1}@sort}%
5895   \letcs\@gls@sort@B{glo\glsdetoklabel{\#2}@sort}%
5896   \edef\glo@do@compare{%
5897     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5898     {\expandonce\@gls@sort@B}%
5899     {\expandonce\@gls@sort@A}%
5900   }%
5901   \glo@do@compare
5902 }
```

`thandler@nocase` Case-insensitive sort.

```
5903 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5904   \letcs\@gls@sort@A{glo\glsdetoklabel{\#1}@sort}%
5905   \letcs\@gls@sort@B{glo\glsdetoklabel{\#2}@sort}%
5906   \edef\glo@do@compare{%
5907     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5908     {\expandonce\@gls@sort@B}%
5909     {\expandonce\@gls@sort@A}%
5910   }%
```

```

5911 \glo@do@compare
5912 }

```

@sortmacro@word Sort macro for ‘word’

```

5913 \newcommand*{\@glo@sortmacro@word}[1]{%
5914 \ifdefstring{\@glo@default@sorttype}{standard}%
5915 {%
5916 \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5917 }%
5918 {%
5919 \PackageError{glossaries}{Conflicting sort options:^^J
5920 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5921 \string\printnoidxglossary[sort=word]}{}%
5922 }%
5923 }

```

ortmacro@letter Sort macro for ‘letter’

```

5924 \newcommand*{\@glo@sortmacro@letter}[1]{%
5925 \ifdefstring{\@glo@default@sorttype}{standard}%
5926 {%
5927 \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5928 }%
5929 {%
5930 \PackageError{glossaries}{Conflicting sort options:^^J
5931 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5932 \string\printnoidxglossary[sort=letter]}{}%
5933 }%
5934 }

```

tmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5935 \newcommand*{\@glo@sortmacro@standard}[1]{%
5936 \ifdefstring{\@glo@default@sorttype}{standard}%
5937 {%
5938 \ifcsdef{\@glo@sorthandler@\glsorder}%
5939 {%
5940 \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
5941 }%
5942 {%
5943 \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5944 }%
5945 }%
5946 {%
5947 \PackageError{glossaries}{Conflicting sort options:^^J
5948 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5949 \string\printnoidxglossary[sort=standard]}{}%
5950 }%
5951 }

```

@sortmacro@case Sort macro for ‘case’

```

5952 \newcommand*{\@glo@sortmacro@case}[1]{%
5953   \ifdefstring{\@glo@default@sorttype}{standard}%
5954   {%
5955     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5956   }%
5957   {%
5958     \PackageError{glossaries}{Conflicting sort options:^^J
5959       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5960       \string\printnoidxglossary[sort=case]}{}%
5961   }%
5962 }

```

ortmacro@nocase Sort macro for ‘nocase’

```

5963 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5964   \ifdefstring{\@glo@default@sorttype}{standard}%
5965   {%
5966     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5967   }%
5968   {%
5969     \PackageError{glossaries}{Conflicting sort options:^^J
5970       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5971       \string\printnoidxglossary[sort=nocase]}{}%
5972   }%
5973 }

```

o@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glolist@*type*.

```

5974 \newcommand*{\@glo@sortmacro@def}[1]{%
5975   \def\@glo@sortinglist{%
5976     \forlgsentries[#1]{\@gls@thislabel}%
5977     {%
5978       \xifinlistcs{\@gls@thislabel}{\@glsref@#1}%
5979       {%
5980         \listeadadd{\@glo@sortinglist}{\@gls@thislabel}%
5981       }%
5982     }%

```

Hasn't been referenced.

```

5983   }%
5984 }%
5985 \cslet{\@glsref@#1}{\@glo@sortinglist}%
5986 }

```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```

5987 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5988   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5989   {}%
5990   {%
5991     \listcsadd{\@glsref@\@glo@type}{#1}%
5992   }%

```



```

5993 \ifcsdef{@glo@sortingchildren@#1}%
5994 {%
5995   \@glo@addchildren{\@glo@type}{#1}%
5996 }%
5997 {}%
5998 }

```

`\sortmacro@use` Sort macro for ‘use’. (No sorting is required, as the entries are already in order of use, so do nothing.)

```

5999 \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.

```

6000 \newcommand*{\@print@noidx@glossary}{%
6001   \ifcsdef{@gls@ref@{\@glo@type}}%
6002   {%
        Sort the entries:
6003     \ifcsdef{@glo@sortmacro@{\@glo@sorttype}}%
6004     {%
6005       \csuse{@glo@sortmacro@{\@glo@sorttype}}{\@glo@type}%
6006     }%
6007     {%
6008       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
6009     }%

```

Do the glossary heading and preamble

```

6010   \glossarysection[\glossarytoctitle]{\glossarytitle}%
6011   \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.

```

6012   \def\@gls@currentlettergroup{%
6013     \begin{theglossary}%
6014     \glossaryheader
6015     \glsresetentrylist

```

Iterate through the entries.

```

6016   \forlistcsloop{\@gls@noidx@do}{\@gls@ref@{\@glo@type}}%

```

Finally end the glossary and do the postamble:

```

6017   \end{theglossary}%
6018   \glossarypostamble
6019 }%
6020 {}%

```

```

6021 \gls@noref@warn{\glo@type}%
6022 }%
6023 }

```

\glo@grabfirst

```

6024 \def\glo@grabfirst#1#2\@nil{%
6025 \def\gls@firsttok{#1}%
6026 \ifdefempty\gls@firsttok
6027 {%
6028 \def\glo@thislettergrp{0}%
6029 }%
6030 {%

```

Sanitize it:

```

6031 \onelevel@sanitize\gls@firsttok

```

Fetch the first letter:

```

6032 \expandafter\glo@grabfirst\gls@firsttok{}\@nil
6033 }%
6034 }

```

\@glo@grabfirst

```

6035 \def\@glo@grabfirst#1#2\@nil{%
6036 \ifdefempty\glo@thislettergrp
6037 {%
6038 \def\glo@thislettergrp{glssymbols}%
6039 }%
6040 {%
6041 \count@=\uccode'#1\relax
6042 \ifnum\count@=0\relax
6043 \def\glo@thislettergrp{glssymbols}%
6044 \else
6045 \ifdefstring\glo@sorttype{case}%
6046 {%
6047 \count@='#1\relax
6048 }%
6049 {%
6050 }%
6051 \edef\glo@thislettergrp{\the\count@}%
6052 \fi
6053 }%
6054 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label.
This only allows one sublevel.

```

6055 \newcommand{\@gls@noidx@do}[1]{%

```

Get this entry's location list

```

6056 \global\letcs{\gls@loclist}{glo\glsdetoklabel{#1}@loclist}%

```

Does this entry have a parent?

```
6057 \ifglshasparent{#1}%  
6058 {%
```

Has a parent.

```
6059 \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax  
6060 \ifdefvoid{\@gls@loclist}  
6061 {%  
6062 \subglossentry{\gls@level}{#1}{}%  
6063 }%  
6064 {%  
6065 \subglossentry{\gls@level}{#1}%  
6066 {%  
6067 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%  
6068 }%  
6069 }%  
6070 }%  
6071 {%
```

Doesn't have a parent Get this entry's sort key

```
6072 \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
```

Fetch the first letter:

```
6073 \expandafter\glo@grabfirst\@gls@sort{ }\@nil  
6074 \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%  
6075 {}%  
6076 {%
```

Do the group header:

```
6077 \ifdefempty{\@gls@currentlettergroup}{}%  
6078 {%
```

The group skip may start a new scope, so make a global assignment.

```
6079 \global\let\@glo@thislettergrp\@glo@thislettergrp  
6080 \glsgroupskip  
6081 }%  
6082 \glsgroupheading{\@glo@thislettergrp}%  
6083 }%  
  
6084 \global\let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
6085 \ifdefvoid{\@gls@loclist}  
6086 {%  
6087 \glossentry{#1}{}%  
6088 }%  
6089 {%  
6090 \glossentry{#1}%  
6091 {%  
6092 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%  
6093 }%  
6094 }%
```

```
6095 }%
6096 }
```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```
6097 \newcommand*{\glsnoidxloclist}[1]{%
6098   \def\@gls@noidxloclist@sep{}%
6099   \def\@gls@noidxloclist@prev{}%
6100   \forlistloop{\glsnoidxloclisthandler}{#1}%
6101 }
```

`xloclisthandler` Handler for location list iterator.

```
6102 \newcommand*{\glsnoidxloclisthandler}[1]{%
6103   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6104   {%
6105     }%
6106     {%
6107       \@gls@noidxloclist@sep
6108       #1%
6109       \def\@gls@noidxloclist@sep{\delimN}%
6110       \def\@gls@noidxloclist@prev{#1}%
6111     }%
6112 }
```

`yloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```
6113 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
6114   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6115   {%
6116     }%
6117     {%
6118       \@gls@noidxloclist@sep
6119       \@gls@noidxloclist@prev
6120       \def\@gls@noidxloclist@prev{#1}%
6121     }%
6122 }
```

`snoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
6123 \newcommand*\glsnoidxdisplayloc[4]{%
6124   \setentrycounter[#1]{#2}%
6125   \csuse{#3}{#4}%
6126 }
```

`\@gls@reference` `\@gls@reference{\langle type \rangle}{\langle label \rangle}{\langle loc \rangle}`

Identifies that a reference has been used (for use in the aux file). All entries must be defined in the preamble.

```
6127 \newcommand*{\@gls@reference}[3]{%
```

Add to label list

```
6128 \glsdoifexistsorwarn{#2}%
6129 {%
6130 \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
6131 \ifinlistcs{#2}{@glsref@#1}%
6132 {}%
6133 {\listcsgadd{@glsref@#1}{#2}}%
```

Add to location list

```
6134 \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
6135 {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}%
6136 {}%
6137 \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
6138 }%
6139 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```
6140 \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`.

```
6141 \define@key{printgloss}{title}{%
6142 \def@glossarytitle{#1}%
6143 \let@gls@dotoc@title\relax
6144 }
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
6145 \define@key{printgloss}{toctitle}{%
6146 \def@glossarytoctitle{#1}%
6147 \let@gls@dotoc@title\relax
6148 }
```

The style key sets the glossary style (but only for the given glossary).

```
6149 \define@key{printgloss}{style}{%
6150 \ifcsundef{@glsstyle@#1}%
6151 {%
6152 \PackageError{glossaries}%
6153 {Glossary style ‘#1’ undefined}{}%
6154 }%
6155 {%
6156 \def@glossarystyle{\setglossentrycompatibility
6157 \csname @glsstyle@#1\endcsname}%
6158 }%
6159 }
```

The `numberedsection` key determines if this glossary should be in a numbered section.

```

6160 \define@choicekey{printgloss}{numberedsection}%
6161 [\gls@numberedsection@val\gls@numberedsection@nr]%
6162 {false,nolabel,autolabel,nameref}[nolabel]%
6163 {%
6164   \ifcase\gls@numberedsection@nr\relax
6165     \renewcommand*{\@@glossarysecstar}{*}%
6166     \renewcommand*{\@@glossaryseclabel}{}%
6167   \or
6168     \renewcommand*{\@@glossarysecstar}{}%
6169     \renewcommand*{\@@glossaryseclabel}{}%
6170   \or
6171     \renewcommand*{\@@glossarysecstar}{}%
6172     \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
6173   \or
6174     \renewcommand*{\@@glossarysecstar}{*}%
6175     \renewcommand*{\@@glossaryseclabel}{%
6176       \protected@edef\@currentlabelname{\glossarytoctitle}%
6177       \label{\glsautoprefix\@glo@type}}%
6178   \fi
6179 }

```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

6180 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
6181   \csuse{glsnogroupskip#1}%
6182 }

```

The `nopostdot` key has the same effect as the package option of the same name.

```

6183 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
6184   \csuse{glsnopostdot#1}%
6185 }

```

CounterLabelPrefix Make it easier to redefine the label prefix.

```

6186 \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.

```

6187 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
6188   \csuse{glsentrycounter#1}%
6189   \@gls@define@glossaryentrycounter
6190 }

```

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.

```

6191 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6192   \csuse{glssubentrycounter#1}%
6193   \@gls@define@glossarysubentrycounter
6194 }

```

The `nonumberlist` key determines if this glossary should have a number list.

```

6195 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
6196 \ifglsnonumberlist
6197   \def\glossaryentrynumbers##1{%
6198 \else
6199   \def\glossaryentrynumbers##1{##1}%
6200 \fi}

```

The `sort` key sets the glossary sort handler (`\printnoidxglossary` only).

```

6201 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

`@assign@sortkey` Issue error if used with `\printglossary`

```

6202 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6203   \PackageError{glossaries}{'sort' key not permitted with
6204   \string\printglossary}%
6205   {The 'sort' key may only be used with \string\printnoidxglossary}%
6206 }

```

`@assign@sortkey` For use with `\printnoidxglossary`

```

6207 \newcommand*{\@glo@assign@sortkey}[1]{%
6208   \def\@glo@sorttype{#1}%
6209 }

```

`@glsnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonextpages` is placed in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6210 \newcommand*{\@glsnonextpages}{%
6211   \gdef\glossaryentrynumbers##1{%
6212     \glsresetentrylist
6213   }%
6214 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is placed in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6215 \newcommand*{\@glsnextpages}{%
6216   \gdef\glossaryentrynumbers##1{%
6217     ##1\glsresetentrylist}}

```

`sresetentrylist` Resets `\glossaryentrynumbers`

```

6218 \newcommand*{\glsresetentrylist}{%
6219   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```

6220 \newcommand*{\glsnonextpages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

6221 \newcommand*{\glsnextpages}{}

```

Process entrycounter and then subentrycounter options (this ensures the sub-counter can pick up the main counter as the master if required):

```

6222 \@gls@define@glossaryentrycounter
6223 \@gls@define@glossarysubentrycounter

```

`subentrycounter` Resets the `glossarysubentry` counter.

```

6224 \newcommand*{\glsresetsubentrycounter}{%
6225   \ifglssubentrycounter
6226     \setcounter{glossarysubentry}{0}%
6227   \fi
6228 }

```

`subentrycounter` Resets the `glossaryentry` counter.

```

6229 \newcommand*{\glsresetentrycounter}{%
6230   \ifglssentrycounter
6231     \setcounter{glossaryentry}{0}%
6232   \fi
6233 }

```

`\glsstepentry` Advance the `glossaryentry` counter if in use. The argument is the label associated with the entry.

```

6234 \newcommand*{\glsstepentry}[1]{%
6235   \ifglssentrycounter
6236     \refstepcounter{glossaryentry}%
6237     \label{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6238   \fi
6239 }

```

`glsstepsubentry` Advance the `glossarysubentry` counter if in use. The argument is the label associated with the subentry.

```

6240 \newcommand*{\glsstepsubentry}[1]{%
6241   \ifglssubentrycounter
6242     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6243     \refstepcounter{glossarysubentry}%
6244     \label{\GlsEntryCounterLabelPrefix\currentglssubentry}%
6245   \fi
6246 }

```


`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```
6247 \newcommand*{\glsrefentry}[1]{%
6248   \ifglentrycounter
6249     \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6250   \else
6251     \ifglssubentrycounter
6252       \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6253     \else
6254       \gls{#1}%
6255     \fi
6256   \fi
6257 }
```

`trycounterlabel` Defines how to display the glossaryentry counter.

```
6258 \newcommand*{\glentrycounterlabel}{%
6259   \ifglentrycounter
6260     \theglossaryentry.\space
6261   \fi
6262 }
```

`trycounterlabel` Defines how to display the glossarysubentry counter.

```
6263 \newcommand*{\glssubentrycounterlabel}{%
6264   \ifglssubentrycounter
6265     \theglossarysubentry)\space
6266   \fi
6267 }
```

`\glentryitem` Step and display glossaryentry counter, if appropriate.

```
6268 \newcommand*{\glentryitem}[1]{%
6269   \ifglentrycounter
6270     \glsstepentry{#1}\glentrycounterlabel
6271   \else
6272     \glsresetsubentrycounter
6273   \fi
6274 }
```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```
6275 \newcommand*{\glssubentryitem}[1]{%
6276   \ifglssubentrycounter
6277     \glsstepsubentry{#1}\glssubentrycounterlabel
6278   \fi
6279 }
```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
6280 \ifcsundef{theglossary}%
6281 {%
6282   \newenvironment{theglossary}{}{}}%
```

```

6283 }%
6284 {%
6285   \@gls@warnontheGLOSSdefined
6286   \renewenvironment{theGLOSS}{-}{-}%
6287 }

```

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 `theGLOSS` 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`

```

6288 \newcommand*{\glossaryheader}{}

```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\glstarget` to make it easier to modify the glossary style in the document.

```

6289 \newcommand*{\glstarget}[2]{\@glstarget{\glo@linkprefix#1}{#2}}

```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

`\compatibleglossentry`

```

\glossentry{<label>}{<page-list>}

```

```

6290 \providecommand*{\compatibleglossentry}[2]{%
6291   \toks@{#2}%
6292   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6293     {\noexpand\glsnamefont
6294       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6295     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6296     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6297     {\the\toks@}}%
6298   }%
6299   \@do@glossentry
6300 }

```

`\glossentryname`

```

6301 \newcommand*{\glossentryname}[1]{%
6302   \glsdoifexistsorwarn{#1}%
6303   {%
6304     \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
6305     \expandafter\glsnamefont\expandafter{\glo@name}%
6306   }%
6307 }

```

\Glossentryname

```
6308 \newcommand*{\Glossentryname}[1]{%
6309   \glsdoifexistsorwarn{#1}%
6310   {%
6311     \glsnamefont{\Glsentryname{#1}}%
6312   }%
6313 }
```

\glossentrydesc

```
6314 \newcommand*{\glossentrydesc}[1]{%
6315   \glsdoifexistsorwarn{#1}%
6316   {%
6317     \glsentrydesc{#1}%
6318   }%
6319 }
```

\Glossentrydesc

```
6320 \newcommand*{\Glossentrydesc}[1]{%
6321   \glsdoifexistsorwarn{#1}%
6322   {%
6323     \Glsentrydesc{#1}%
6324   }%
6325 }
```

\glossentrysymbol

```
6326 \newcommand*{\glossentrysymbol}[1]{%
6327   \glsdoifexistsorwarn{#1}%
6328   {%
6329     \glsentrysymbol{#1}%
6330   }%
6331 }
```

\Glossentrysymbol

```
6332 \newcommand*{\Glossentrysymbol}[1]{%
6333   \glsdoifexistsorwarn{#1}%
6334   {%
6335     \Glsentrysymbol{#1}%
6336   }%
6337 }
```

\compatiblesubglossentry

`\subglossentry{<level>}{<label>}{<page-list>}`

```
6338 \providecommand*{\compatiblesubglossentry}[3]{%
6339   \toks@{#3}%
6340   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6341     {#2}%
6342     {\noexpand\glsnamefont
```

```

6343      {\expandafter\expandonce\csname glo@#2@name\endcsname}}%
6344      {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6345      {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6346      {\the\toks@}%
6347  }%
6348  \@do@subglossentry
6349 }

```

rycompatibility

```

6350 \newcommand*{\setglossentrycompatibility}{%
6351   \let\glossentry\compatibleglossentry
6352   \let\subglossentry\compatiblesubglossentry
6353 }
6354 \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.

```

6355 \newcommand{\glossaryentryfield}[5]{%
6356   \GlossariesWarning
6357   {Deprecated use of \string\glossaryentryfield.^^J
6358     I recommend you change to \string\glossentry.^^J
6359     If you've just upgraded, try removing your gls auxiliary
6360     files^^J and recompile}%
6361   \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.)

```

6362 \newcommand*{\glossarysubentryfield}[6]{%
6363   \GlossariesWarning
6364   {Deprecated use of \string\glossarysubentryfield.^^J
6365     I recommend you change to \string\subglossentry.^^J
6366     If you've just upgraded, try removing your gls auxiliary
6367     files^^J and recompile}%
6368   \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`

```
6369 \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`

```
6370 \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`

```
6371 \newcommand*{\glsgetgrouptitle}[1]{%
6372   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6373   \@gls@grptitle
6374 }
```

`\@gls@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
6375 \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.

```

6376 \dtl@ifsingle{#1}%
6377 {%
6378   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6379 }%
6380 {%
6381   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6382               or test{\ifstrequal{#1}{glsnumbers}}}%
6383   {%
6384     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6385   }%
6386   {%
6387     \def#2{#1}%
6388   }%
6389 }%
6390 }

```

`x@getgrouptitle` Version for the no-indexing app option:

```

6391 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6392   \DTLifint{#1}%
6393   {\edef#2{\char#1\relax}}%
6394   {%
6395     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}}%
6396   }%
6397 }

```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`lsgetgrouplabel`

```

6398 \newcommand*{\glsgetgrouplabel}[1]{%
6399 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6400 \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`

```

6401 \newcommand*{\setentrycounter}[2][ ]{%
6402   \def\@glo@counterprefix{#1}%
6403   \ifx\@glo@counterprefix\empty
6404     \def\@glo@counterprefix{.}%
6405   \else

```

```

6406 \def\@glo@counterprefix{.#1.}%
6407 \fi
6408 \def\glsentrycounter{#2}%
6409 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

6410 \newcommand*{\setglossarystyle}[1]{%
6411 \ifcsundef{@glsstyle@#1}%
6412 {%
6413 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6414 }%
6415 {%
6416 \csname @glsstyle@#1\endcsname
6417 }%

```

Set the default style if it's not already set.

```

6418 \ifx\@glossary@default@style\relax
6419 \protected@edef\@glossary@default@style{#1}%
6420 \fi
6421 }

```

`\glossarystyle`

```

6422 \newcommand*{\glossarystyle}[1]{%
6423 \ifcsundef{@glsstyle@#1}%
6424 {%
6425 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6426 }%
6427 {%
6428 \GlossariesWarning
6429 {Deprecated command \string\glossarystyle.^~J
6430 I recommend you switch to \string\setglossarystyle\space unless
6431 you want to maintain backward compatibility}%
6432 \setglossentrycompatibility
6433 \csname @glsstyle@#1\endcsname

6434 \ifcsdef{@glscompstyle@#1}%
6435 {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6436 {}%
6437 }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6438 \ifx\@glossary@default@style\relax
6439 \protected@edef\@glossary@default@style{#1}%
6440 \fi
6441 }

```

`\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.

```
6442 \newcommand{\newglossarystyle}[2]{%
6443   \ifcsundef{@glstyle@#1}%
6444   {%
6445     \expandafter\def\csname @glstyle@#1\endcsname{#2}%
6446   }%
6447   {%
6448     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6449   }%
6450 }
```

`\newglossarystyle` Code for this macro supplied by Marco Daniel.

```
6451 \newcommand{\renewglossarystyle}[2]{%
6452   \ifcsundef{@glstyle@#1}%
6453   {%
6454     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6455   }%
6456   {%
6457     \csdef{@glstyle@#1}{#2}%
6458   }%
6459 }
```

Glossary entries are encoded so that the second argument to `\glossaryentryfield` is always specified as `\glsglsnamefont{<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.

`\glsglsnamefont`

```
6460 \newcommand*{\glsglsnamefont}[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`

```
6461 \ifcsundef{hyperlink}%
6462 {%
6463   \def\glshypernumber#1{#1}%
6464 }%
6465 {%
6466   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6467 }
```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6468 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6469   \ifx\#1\%
6470   \else
6471     \@delimR#1\delimR\delimR\%
6472   \fi
6473   \ifx\#2\%
6474   \else
6475     #2%
6476   \fi
6477   \ifx\#3\%
6478   \else
6479     \@glshypernumber#3\@nil
6480   \fi
6481 }
```

`\@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`

```
6482 \def\@delimR#1\delimR #2\delimR #3\%
6483 \ifx\#2\%
6484   \@delimN{#1}%
6485 \else
6486   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6487 \fi}
```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```
6488 \def\@delimN#1{\@delimN#1\delimN \delimN\%
6489 \def\@delimN#1\delimN #2\delimN#3\%
6490 \ifx\#3\%
6491   \@gls@numberlink{#1}%
6492 \else
6493   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6494 \fi
6495 }
```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

6496 \def\@gls@numberlink#1{%
6497 \begingroup
6498 \toks@={}%
6499 \@gls@removespaces#1 \@nil
6500 \endgroup}

6501 \def\@gls@removespaces#1 #2\@nil{%
6502 \toks@=\expandafter{\the\toks@#1}%
6503 \ifx\#2\%
6504 \edef\x{\the\toks@}%
6505 \ifx\x\empty
6506 \else

6507 \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
6508 {\the\toks@}%
6509 \fi
6510 \else
6511 \@gls@ReturnAfterFi{%
6512 \@gls@removespaces#2\@nil
6513 }%
6514 \fi
6515 }
6516 \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

```
6517 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}
```

\hypersf

```
6518 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}
```

\hypertt

```
6519 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}
```

\hyperbf

```
6520 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}
```

\hypermd

```
6521 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}
```

\hyperit

```
6522 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}
```

\hypersl

```
6523 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6524 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6525 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6526 \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.

```
6527 \newcommand{\oldacronym}[4][\gls@label]{%
6528   \def\gls@label{#2}%
6529   \newacronym[#4]{#1}{#2}{#3}%
6530   \ifcsundef{xspace}%
6531     {%
6532       \expandafter\edef\csname#1\endcsname{%
6533         \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6534       }%
6535     }%
6536     {%
6537       \expandafter\edef\csname#1\endcsname{%
6538         \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6539           \noexpand\gls{#1}\noexpand\xspace}%
6540       }%
6541     }%
6542 }
```

`\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrev⟩}{⟨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`

```
6543 \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.

```
6544 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6545 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6546 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
6547 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6548 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
6549 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6550 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}
```

```
6551 \newcommand*{\ns@acrfull}[2] [] {%
```

```
6552 \new@ifnextchar[\@acrfull{#1}{#2}}%
```

```
6553 \@acrfull{#1}{#2} []}%
```

```
6554 }
```

`\@acrfull` Low-level macro:

```

6555 \def\@acrfull#1#2[#3]{%
    Make it easier for acronym styles to change this:
6556   \acrfullfmt{#1}{#2}{#3}%
6557 }

```

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.

```

6558 \newcommand*\acrfullfmt[3]{%
6559   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6560 }

```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```

6561 \newcommand*\acrlinkfullformat[5]{%
6562   \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4}[]}%
6563 }

```

`\acrfullformat` Default full form is *<long>* (*<short>*).

```

6564 \newcommand*\acrfullformat[2]{#1\glsspace(#2)}

```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```

6565 \newrobustcmd{\glsspace}{\space}

```

Default format for full acronym

`\Acrfull`

```

6566 \newrobustcmd*\Acrfull{\@gls@hyp@opt\ns@Acrfull}

6567 \newcommand*\ns@Acrfull[2][]{%
6568   \new@ifnextchar[\@Acrfull{#1}{#2}}%
6569   {\@Acrfull{#1}{#2}[]}%
6570 }

```

Low-level macro:

```

6571 \def\@Acrfull#1#2[#3]{%
    Make it easier for acronym styles to change this:
6572   \Acrfullfmt{#1}{#2}{#3}%
6573 }

```

`\Acrfullfmt` First letter upper case full format.

```

6574 \newcommand*\Acrfullfmt[3]{%
6575   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6576 }

```

`\ACRfull`

```
6577 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}
```

```
6578 \newcommand*\ns@ACRfull[2] [] {%
6579   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6580     {\@ACRfull{#1}{#2} []}%
6581 }
```

Low-level macro:

```
6582 \def\@ACRfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6583   \ACRfullfmt{#1}{#2}{#3}%
6584 }
```

`\ACRfullfmt` All upper case full format.

```
6585 \newcommand*{\ACRfullfmt}[3] {%
6586   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
6587 }
```

Plural:

`\acrfullpl`

```
6588 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}
```

```
6589 \newcommand*\ns@acrfullpl[2] [] {%
6590   \new@ifnextchar[{\@acrfullpl{#1}{#2}}%
6591     {\@acrfullpl{#1}{#2} []}%
6592 }
```

Low-level macro:

```
6593 \def\@acrfullpl#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6594   \acrfullplfmt{#1}{#2}{#3}%
6595 }
```

`\acrfullplfmt` No case change plural full format.

```
6596 \newcommand*{\acrfullplfmt}[3] {%
6597   \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%
6598 }
```

`\Acrfullpl`

```
6599 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}
```

```
6600 \newcommand*\ns@Acrfullpl[2] [] {%
6601   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%
6602     {\@Acrfullpl{#1}{#2} []}%
6603 }
```

Low-level macro:

```
6604 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6605 \Acrfullplfmt{#1}{#2}{#3}%  
6606 }
```

`\Acrfullplfmt` First letter upper case plural full format.

```
6607 \newcommand*\@Acrfullplfmt[3]{%  
6608 \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6609 }
```

`\ACRfullpl`

```
6610 \newrobustcmd*\@ACRfullpl{\@gls@hyp@opt\@ns@ACRfullpl}  
  
6611 \newcommand*\@ns@ACRfullpl[2][ ]{%  
6612 \new@ifnextchar[{\@ACRfullpl{#1}{#2}}{%  
6613 {\@ACRfullpl{#1}{#2}[ ]}%  
6614 }
```

Low-level macro:

```
6615 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6616 \ACRfullplfmt{#1}{#2}{#3}%  
6617 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6618 \newcommand*\@ACRfullplfmt[3]{%  
6619 \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6620 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6621 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6622 \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.

```
6623 \newcommand*\acrnameformat[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6624 \newtoks\glskeylisttok
```

```

\glslabeltok
6625 \newtoks\glslabeltok

\glsshorttok
6626 \newtoks\glsshorttok

\glslongtok
6627 \newtoks\glslongtok

\newacronymhook  Provide a hook for \newacronym:
6628 \newcommand*{\newacronymhook}{}

\genericNewAcronym  New improved version of setting the acronym style.
6629 \newcommand*{\SetGenericNewAcronym}{%
    Change the behaviour of \Glsentryname to workaround expansion issues that cause a prob-
    lem for \makefirstuc
6630 \let\@Gls@entryname\@Gls@acentryname
    Change the way acronyms are defined:
6631 \renewcommand{\newacronym}[4][\{%
6632 \ifdefempty{\@glsacronymlists}%
6633 {%
6634 \def\@glo@type{\acronymtype}%
6635 \setkeys{glossentry}{##1}%
6636 \DeclareAcronymList{\@glo@type}%
6637 }%
6638 }%
6639 \glskeylisttok{##1}%
6640 \glslabeltok{##2}%
6641 \glsshorttok{##3}%
6642 \glslongtok{##4}%
6643 \newacronymhook
6644 \protected@edef\@do@newglossaryentry{%
6645 \noexpand\newglossaryentry{\the\glslabeltok}%
6646 {%
6647 type=\acronymtype,%
6648 name={\expandonce{\acronymentry{##2}}},%
6649 sort={\acronymssort{\the\glsshorttok}{\the\glslongtok}},%
6650 text={\the\glsshorttok},%
6651 short={\the\glsshorttok},%
6652 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6653 long={\the\glslongtok},%
6654 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6655 \GenericAcronymFields,%
6656 \the\glskeylisttok
6657 }%
6658 }%
6659 \@do@newglossaryentry
6660 }%

```


Make sure that `\acrfull` etc reflects the new style:

```

6661 \renewcommand*{\acrfullfmt}[3]{%
6662   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6663 \renewcommand*{\Acrfullfmt}[3]{%
6664   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6665 \renewcommand*{\ACRfullfmt}[3]{%
6666   \glslink[##1]{##2}{%
6667     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}}%
6668 \renewcommand*{\acrfullplfmt}[3]{%
6669   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6670 \renewcommand*{\Acrfullplfmt}[3]{%
6671   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6672 \renewcommand*{\ACRfullplfmt}[3]{%
6673   \glslink[##1]{##2}{%
6674     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}}%

```

Make sure that `\glentryfull` etc reflects the new style:

```

6675 \renewcommand*{\glentryfull}[1]{\genacrfullformat{##1}{}}}%
6676 \renewcommand*{\Glentryfull}[1]{\Genacrfullformat{##1}{}}}%
6677 \renewcommand*{\glentryfullpl}[1]{\genplacrfullformat{##1}{}}}%
6678 \renewcommand*{\Glentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
6679 }

```

`\GenericAcronymFields` Fields used by `\SetGenericNewAcronym` that can be changed by the acronym style.

```

6680 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6681 \newcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6682 \newcommand*{\acronymsort}[2]{#1}

```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```

6683 \newcommand*{\setacronymstyle}[1]{%
6684   \ifcsundef{@glsacr@dispstyle@#1}
6685   {%
6686     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6687   }%
6688   {%

```

```

6689 \ifdefempty{\@glsacronymlists}%
6690 {%
6691   \DeclareAcronymList{\acronymtype}%
6692   }%
6693   {}%
6694   \SetGenericNewAcronym
6695   \GlsUseAcrStyleDefs{#1}%
6696   \@for\@gls@type:=\@glsacronymlists\do{%
6697     \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6698   }%
6699   }%
6700 }

```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```

6701 \newcommand*{\newacronymstyle}[3]{%
6702   \ifcsdef{@glsacr@dispstyle@#1}%
6703   {%
6704     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6705   }%
6706   {%
6707     \csdef{@glsacr@dispstyle@#1}{#2}%
6708     \csdef{@glsacr@styledefs@#1}{#3}%
6709   }%
6710 }

```

`\renewacronymstyle` Redefines the given acronym style.

```

6711 \newcommand*{\renewacronymstyle}[3]{%
6712   \ifcsdef{@glsacr@dispstyle@#1}%
6713   {%
6714     \csdef{@glsacr@dispstyle@#1}{#2}%
6715     \csdef{@glsacr@styledefs@#1}{#3}%
6716   }%
6717   {%
6718     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6719   }%
6720 }

```

`\rEntryDispStyle`

```

6721 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

`\UseAcrStyleDefs`

```

6722 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *<long>* (*<short>*) acronym style.

```
6723 \newacronymstyle{long-short}%
6724 {%
    Check for long form in case this is a mixed glossary.
6725   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6726 }%
6727 {%
6728   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6729   \renewcommand*{\genacrfullformat}[2]{%
6730     \glentrylong{##1}##2\space
6731     (\protect\firstacronymfont{\glentryshort{##1}})%
6732   }%
6733   \renewcommand*{\Genacrfullformat}[2]{%
6734     \Glentrylong{##1}##2\space
6735     (\protect\firstacronymfont{\glentryshort{##1}})%
6736   }%
6737   \renewcommand*{\genplacrfullformat}[2]{%
6738     \glentrylongpl{##1}##2\space
6739     (\protect\firstacronymfont{\glentryshortpl{##1}})%
6740   }%
6741   \renewcommand*{\Genplacrfullformat}[2]{%
6742     \Glentrylongpl{##1}##2\space
6743     (\protect\firstacronymfont{\glentryshortpl{##1}})%
6744   }%
6745   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6746   \renewcommand*{\acronymsort}[2]{##1}%
6747   \renewcommand*{\acronymfont}[1]{##1}%
6748   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6749   \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6750 }
```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```
6751 \newacronymstyle{long-sp-short}%
6752 {%
    Check for long form in case this is a mixed glossary.
6753   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6754 }%
6755 {%
6756   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6757   \renewcommand*{\genacrfullformat}[2]{%
6758     \glentrylong{##1}##2\glssacspace{##1}%
6759     (\protect\firstacronymfont{\glentryshort{##1}})%
6760   }%
6761   \renewcommand*{\Genacrfullformat}[2]{%
6762     \Glentrylong{##1}##2\glssacspace{##1}%
6763     (\protect\firstacronymfont{\glentryshort{##1}})%
6764   }%
}
```

```

6765 \renewcommand*{\genplacrfullformat}[2]{%
6766 \glentrylongpl{##1}##2\glsacspace{##1}%
6767 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6768 }%
6769 \renewcommand*{\Genplacrfullformat}[2]{%
6770 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6771 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6772 }%
6773 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6774 \renewcommand*{\acronymsort}[2]{##1}%
6775 \renewcommand*{\acronymfont}[1]{##1}%
6776 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6777 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6778 }

```

`\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.

```

6779 \newcommand*{\glsacspace}[1]{%
6780 \settowidth{\dimen@}{(\firstacronymfont{\glentryshort{##1}})}%
6781 \ifdim\dimen@<3em~\else\space\fi
6782 }

```

`short-long` (*short*) (*long*) acronym style.

```

6783 \newacronymstyle{short-long}%
6784 {%

```

Check for long form in case this is a mixed glossary.

```

6785 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6786 }%
6787 {%
6788 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6789 \renewcommand*{\genacrfullformat}[2]{%
6790 \protect\firstacronymfont{\glentryshort{##1}}##2\space
6791 (\glentrylong{##1})%
6792 }%
6793 \renewcommand*{\Genacrfullformat}[2]{%
6794 \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6795 (\glentrylong{##1})%
6796 }%
6797 \renewcommand*{\genplacrfullformat}[2]{%
6798 \protect\firstacronymfont{\glentryshortpl{##1}}##2\space
6799 (\glentrylongpl{##1})%
6800 }%
6801 \renewcommand*{\Genplacrfullformat}[2]{%
6802 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6803 (\glentrylongpl{##1})%
6804 }%

6805 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6806 \renewcommand*{\acronymsort}[2]{##1}%

```

```

6807 \renewcommand*{\acronymfont}[1]{##1}%
6808 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6809 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6810 }

```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```

6811 \newacronymstyle{long-sc-short}%
6812 {%
6813   \GlsUseAcrEntryDisplayStyle{long-short}%
6814 }%
6815 {%
6816   \GlsUseAcrStyleDefs{long-short}%
6817   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6818   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6819 }

```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6820 \newacronymstyle{long-sm-short}%
6821 {%
6822   \GlsUseAcrEntryDisplayStyle{long-short}%
6823 }%
6824 {%
6825   \GlsUseAcrStyleDefs{long-short}%
6826   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6827   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6828 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6829 \newacronymstyle{sc-short-long}%
6830 {%
6831   \GlsUseAcrEntryDisplayStyle{short-long}%
6832 }%
6833 {%
6834   \GlsUseAcrStyleDefs{short-long}%
6835   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6836   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6837 }

```

sm-short-long *<short>* (\textsmaller{<long>}) acronym style.

```

6838 \newacronymstyle{sm-short-long}%
6839 {%
6840   \GlsUseAcrEntryDisplayStyle{short-long}%
6841 }%
6842 {%
6843   \GlsUseAcrStyleDefs{short-long}%
6844   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6845   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6846 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6847 \newacronymstyle{long-short-desc}%
6848 {%
6849   \GlsUseAcrEntryDisplayStyle{long-short}%
6850 }%
6851 {%
6852   \GlsUseAcrStyleDefs{long-short}%
6853   \renewcommand*{\GenericAcronymFields}{}%
6854   \renewcommand*{\acronymsort}[2]{##2}%
6855   \renewcommand*{\acronymentry}[1]{%
6856     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6857 }

```

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`.

```

6858 \newacronymstyle{long-sp-short-desc}%
6859 {%
6860   \GlsUseAcrEntryDisplayStyle{long-sp-short}%
6861 }%
6862 {%
6863   \GlsUseAcrStyleDefs{long-sp-short}%
6864   \renewcommand*{\GenericAcronymFields}{}%
6865   \renewcommand*{\acronymsort}[2]{##2}%
6866   \renewcommand*{\acronymentry}[1]{%
6867     \glstrylong{##1}\glsacspace{##1}(\acronymfont{\glstryshort{##1}})}%
6868 }

```

g-sc-short-desc *<long>* (`\textsc{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6869 \newacronymstyle{long-sc-short-desc}%
6870 {%
6871   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
6872 }%
6873 {%
6874   \GlsUseAcrStyleDefs{long-sc-short}%
6875   \renewcommand*{\GenericAcronymFields}{}%
6876   \renewcommand*{\acronymsort}[2]{##2}%
6877   \renewcommand*{\acronymentry}[1]{%
6878     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6879 }

```

g-sm-short-desc *<long>* (`\textsmaller{<short>}`) acronym style that has an accompanying description (which the user needs to supply).

```

6880 \newacronymstyle{long-sm-short-desc}%
6881 {%
6882   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
6883 }%

```

```

6884 {%
6885   \GlsUseAcrStyleDefs{long-sm-short}%
6886   \renewcommand*{\GenericAcronymFields}{}%
6887   \renewcommand*{\acronymsort}[2]{##2}%
6888   \renewcommand*{\acronymentry}[1]{%
6889     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6890 }

```

short-long-desc *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

6891 \newacronymstyle{short-long-desc}%
6892 {%
6893   \GlsUseAcrEntryDisplayStyle{short-long}%
6894 }%
6895 {%
6896   \GlsUseAcrStyleDefs{short-long}%
6897   \renewcommand*{\GenericAcronymFields}{}%
6898   \renewcommand*{\acronymsort}[2]{##2}%
6899   \renewcommand*{\acronymentry}[1]{%
6900     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6901 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6902 \newacronymstyle{sc-short-long-desc}%
6903 {%
6904   \GlsUseAcrEntryDisplayStyle{sc-short-long}%
6905 }%
6906 {%
6907   \GlsUseAcrStyleDefs{sc-short-long}%
6908   \renewcommand*{\GenericAcronymFields}{}%
6909   \renewcommand*{\acronymsort}[2]{##2}%
6910   \renewcommand*{\acronymentry}[1]{%
6911     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6912 }

```

short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6913 \newacronymstyle{sm-short-long-desc}%
6914 {%
6915   \GlsUseAcrEntryDisplayStyle{sm-short-long}%
6916 }%
6917 {%
6918   \GlsUseAcrStyleDefs{sm-short-long}%
6919   \renewcommand*{\GenericAcronymFields}{}%
6920   \renewcommand*{\acronymsort}[2]{##2}%
6921   \renewcommand*{\acronymentry}[1]{%
6922     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6923 }

```

dua *<long>* only acronym style.

```
6924 \newacronymstyle{dua}%  
6925 {%
```

Check for long form in case this is a mixed glossary.

```
6926 \ifdefempty\glscustomtext  
6927 {%  
6928 \ifglshaslong{\glslabel}%  
6929 {%  
6930 \glsifplural  
6931 {%
```

Plural form:

```
6932 \glscapscase  
6933 {%
```

Plural form, don't adjust case:

```
6934 \glstrylongpl{\glslabel}\glsinsert  
6935 }%  
6936 {%
```

Plural form, make first letter upper case:

```
6937 \Glstrylongpl{\glslabel}\glsinsert  
6938 }%  
6939 {%
```

Plural form, all caps:

```
6940 \mfirstucMakeUppercase  
6941 {\glstrylongpl{\glslabel}\glsinsert}%  
6942 }%  
6943 }%  
6944 {%
```

Singular form

```
6945 \glscapscase  
6946 {%
```

Singular form, don't adjust case:

```
6947 \glstrylong{\glslabel}\glsinsert  
6948 }%  
6949 {%
```

Subsequent singular form, make first letter upper case:

```
6950 \Glstrylong{\glslabel}\glsinsert  
6951 }%  
6952 {%
```

Subsequent singular form, all caps:

```
6953 \mfirstucMakeUppercase  
6954 {\glstrylong{\glslabel}\glsinsert}%  
6955 }%  
6956 }%  
6957 }%  
6958 {%
```


Not an acronym:

```
6959     \glsgenentryfmt
6960     }%
6961     }%
6962     {\glscustomtext\glsinsert}%
6963 }%
6964 {%
6965   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6966   \renewcommand*{\acrfullfmt}[3]{%
6967     \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6968       (\acronymfont{\glsentryshort{##2}})}}%
6969   \renewcommand*{\Acrfullfmt}[3]{%
6970     \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6971       (\acronymfont{\glsentryshort{##2}})}}%
6972   \renewcommand*{\ACRfullfmt}[3]{%
6973     \glslink[##1]{##2}{%
6974       \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6975         (\acronymfont{\glsentryshort{##2}})}}}%

6976   \renewcommand*{\acrfullplfmt}[3]{%
6977     \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6978       (\acronymfont{\glsentryshortpl{##2}})}}%

6979   \renewcommand*{\Acrfullplfmt}[3]{%
6980     \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6981       (\acronymfont{\glsentryshortpl{##2}})}}%
6982   \renewcommand*{\ACRfullplfmt}[3]{%
6983     \glslink[##1]{##2}{%
6984       \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6985         (\acronymfont{\glsentryshortpl{##2}})}}}%
6986   \renewcommand*{\glsentryfull}[1]{%
6987     \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6988   }%
6989   \renewcommand*{\Glsentryfull}[1]{%
6990     \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6991   }%
6992   \renewcommand*{\glsentryfullpl}[1]{%
6993     \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6994   }%
6995   \renewcommand*{\Glsentryfullpl}[1]{%
6996     \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6997   }%
6998   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6999   \renewcommand*{\acronymsort}[2]{##1}%
7000   \renewcommand*{\acronymfont}[1]{##1}%
7001   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7002 }
```

dua-desc *<long>* only acronym style with user-supplied description.

```
7003 \newacronymstyle{dua-desc}%
7004 {%
7005   \GlsUseAcrEntryDispStyle{dua}%
7006 }%
7007 {%
7008   \GlsUseAcrStyleDefs{dua}%
7009   \renewcommand*{\GenericAcronymFields}{}%

7010   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
7011   \renewcommand*{\acronymsort}[2]{##2}%
7012 }%
```

footnote *<short>*\footnote{*<long>*} acronym style.

```
7013 \newacronymstyle{footnote}%
7014 {%

  Check for long form in case this is a mixed glossary.

7015   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
7016 }%
7017 {%
7018   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

Need to ensure hyperlinks are switched off on first use:

```
7019 \glshyperfirstfalse
7020 \renewcommand*{\genacrfullformat}[2]{%
7021   \protect\firstacronymfont{\glentryshort{##1}}##2%
7022   \protect\footnote{\glentrylong{##1}}%
7023 }%
7024 \renewcommand*{\Genacrfullformat}[2]{%
7025   \firstacronymfont{\Glentryshort{##1}}##2%
7026   \protect\footnote{\glentrylong{##1}}%
7027 }%
7028 \renewcommand*{\genplacrfullformat}[2]{%
7029   \protect\firstacronymfont{\glentryshortpl{##1}}##2%
7030   \protect\footnote{\glentrylongpl{##1}}%
7031 }%
7032 \renewcommand*{\Genplacrfullformat}[2]{%
7033   \protect\firstacronymfont{\Glentryshortpl{##1}}##2%
7034   \protect\footnote{\glentrylongpl{##1}}%
7035 }%
7036 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
7037 \renewcommand*{\acronymsort}[2]{##1}%
7038 \renewcommand*{\acronymfont}[1]{##1}%
7039 \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%

Don't use footnotes for \acrfull:
```

```
7040 \renewcommand*{\acrfullfmt}[3]{%
7041   \glslink[##1]{##2}{\acronymfont{\glentryshort{##2}}##3\space
7042     (\glentrylong{##2})}%}
```

```

7043 \renewcommand*{\Acrfullfmt}[3]{%
7044   \glslink{##1}{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
7045   (\glsentrylong{##2})}%
7046 \renewcommand*{\ACRfullfmt}[3]{%
7047   \glslink{##1}{##2}{%
7048     \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
7049     (\glsentrylong{##2})}}}%
7050 \renewcommand*{\acrfullplfmt}[3]{%
7051   \glslink{##1}{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
7052   (\glsentrylongpl{##2})}%
7053 \renewcommand*{\Acrfullplfmt}[3]{%
7054   \glslink{##1}{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
7055   (\glsentrylongpl{##2})}%
7056 \renewcommand*{\ACRfullplfmt}[3]{%
7057   \glslink{##1}{##2}{%
7058     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
7059     (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

7060 \renewcommand*{\glsentryfull}[1]{%
7061   \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
7062 \renewcommand*{\Glsentryfull}[1]{%
7063   \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
7064 \renewcommand*{\glsentryfullpl}[1]{%
7065   \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7066 \renewcommand*{\Glsentryfullpl}[1]{%
7067   \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7068 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

7069 \newacronymstyle{footnote-sc}%
7070 {%
7071   \GlsUseAcrEntryDispStyle{footnote}%
7072 }%
7073 {%
7074   \GlsUseAcrStyleDefs{footnote}%
7075   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
7076   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
7077   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7078 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

7079 \newacronymstyle{footnote-sm}%
7080 {%
7081   \GlsUseAcrEntryDispStyle{footnote}%
7082 }%
7083 {%
7084   \GlsUseAcrStyleDefs{footnote}%
7085   \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
7086   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%

```

```

7087 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7088 }%

```

footnote-desc *<short>*\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

7089 \newacronymstyle{footnote-desc}%
7090 {%
7091 \GlsUseAcrEntryDisplayStyle{footnote}%
7092 }%
7093 {%
7094 \GlsUseAcrStyleDefs{footnote}%
7095 \renewcommand*{\GenericAcronymFields}{}%
7096 \renewcommand*{\acronymsort}[2]{##2}%
7097 \renewcommand*{\acronymentry}[1]{%
7098 \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
7099 }

```

footnote-sc-desc \textsc{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

7100 \newacronymstyle{footnote-sc-desc}%
7101 {%
7102 \GlsUseAcrEntryDisplayStyle{footnote-sc}%
7103 }%
7104 {%
7105 \GlsUseAcrStyleDefs{footnote-sc}%
7106 \renewcommand*{\GenericAcronymFields}{}%
7107 \renewcommand*{\acronymsort}[2]{##2}%
7108 \renewcommand*{\acronymentry}[1]{%
7109 \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
7110 }

```

footnote-sm-desc \textsmaller{*<short>*}\footnote{*<long>*} acronym style that has an accompanying description (which the user needs to supply).

```

7111 \newacronymstyle{footnote-sm-desc}%
7112 {%
7113 \GlsUseAcrEntryDisplayStyle{footnote-sm}%
7114 }%
7115 {%
7116 \GlsUseAcrStyleDefs{footnote-sm}%
7117 \renewcommand*{\GenericAcronymFields}{}%
7118 \renewcommand*{\acronymsort}[2]{##2}%
7119 \renewcommand*{\acronymentry}[1]{%
7120 \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
7121 }

```

AcronymSynonyms

```

7122 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

`\acs`
 7123 `\let\acs\acrshort`
 First letter uppercase short form

`\Acs`
 7124 `\let\Acs\Acrshort`
 Plural short form

`\acsp`
 7125 `\let\acsp\acrshortpl`
 First letter uppercase plural short form

`\Acsp`
 7126 `\let\Acsp\Acrshortpl`
 Long form

`\acl`
 7127 `\let\acl\acllong`
 Plural long form

`\aclp`
 7128 `\let\aclp\acllongpl`
 First letter upper case long form

`\Acl`
 7129 `\let\Acl\Acllong`
 First letter upper case plural long form

`\Aclp`
 7130 `\let\Aclp\Acllongpl`
 Full form

`\acf`
 7131 `\let\acf\acrfull`
 Plural full form

`\acfp`
 7132 `\let\acfp\acrfullpl`
 First letter upper case full form

`\Acf`
 7133 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

```
7134 \let\Acfp\Acrfullpl
```

Standard form

`\ac`

```
7135 \let\ac\gls
```

First upper case standard form

`\Ac`

```
7136 \let\Ac\Gls
```

Standard plural form

`\acp`

```
7137 \let\acp\glspl
```

Standard first letter upper case plural form

`\Acp`

```
7138 \let\Acp\Glspl
```

```
7139 }
```

Define synonyms if required

```
7140 \ifglsacrshortcuts
```

```
7141 \DefineAcronymSynonyms
```

```
7142 \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.

```
7143 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
```

```
7144 \defglsentryfmt[#1]{\glsentryfmt}%
```

```
7145 }
```

`\glsNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\gslongtok` and `\glskeylisttok`.

```
7146 \newcommand*{\DefaultNewAcronymDef}{%
```

```
7147 \edef\@do@newglossaryentry{%
```

```
7148 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
7149 {%
```

```
7150 type=\acronymtype,%
```

```
7151 name={\the\glsshorttok},%
```

```
7152 sort={\the\glsshorttok},%
```

```
7153 text={\the\glsshorttok},%
```

```
7154 first={\acrfullformat{\the\gslongtok}{\the\glsshorttok}},%
```

```
7155 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
```

```

7156     firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7157                               {\noexpand\expandonce\noexpand\@glo@shortpl}},%
7158     short={\the\glsshorttok},%
7159     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7160     long={\the\glslongtok},%
7161     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7162     description={\the\glslongtok},%
7163     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

Remaining options specified by the user:

```

7164     \the\glskeylisttok
7165   }%
7166 }%
7167 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7168 \let\@org@gls@assign@plural\gls@assign@plural
7169 \let\@org@gls@assign@descplural\gls@assign@descplural
7170 \def\gls@assign@firstpl##1##2{%
7171   \@gls@expand@field{##1}{firstpl}{##2}%
7172 }%
7173 \def\gls@assign@plural##1##2{%
7174   \@gls@expand@field{##1}{plural}{##2}%
7175 }%
7176 \def\gls@assign@descplural##1##2{%
7177   \@gls@expand@field{##1}{descplural}{##2}%
7178 }%
7179 \@do@newglossaryentry
7180 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7181 \let\gls@assign@plural\@org@gls@assign@plural
7182 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7183 }

```

ultAcronymStyle Set up the default acronym style:

```

7184 \newcommand*{\SetDefaultAcronymStyle}{%
  Set the display style:
7185   \@for\@gls@type:=\@glsacronymlists\do{%
7186     \SetDefaultAcronymDisplayStyle{\@gls@type}%
7187   }%

```

Set up the definition of `\newacronym`:

```

7188 \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).

```

7189   \ifx\@glsacronymlists\@empty
7190     \def\@glo@type{\acronymtype}%
7191     \setkeys{glossentry}{##1}%
7192     \DeclareAcronymList{\@glo@type}%
7193     \SetDefaultAcronymDisplayStyle{\@glo@type}%
7194   \fi
7195   \glskeylisttok{##1}%

```

```

7196 \glslabeltok{##2}%
7197 \glsshorttok{##3}%
7198 \glslongtok{##4}%
7199 \newacronymhook
7200 \DefaultNewAcronymDef
7201 }%
7202 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7203 }

```

`\acrfootnote` Used by the footnote acronym styles.

```

7204 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}

```

`acrlinkfootnote`

```

7205 \newcommand*{\acrlinkfootnote}[3]{%
7206 \footnote{\glslink[#1]{#2}{#3}}%
7207 }

```

`acrnolinkfootnote`

```

7208 \newcommand*{\acrnolinkfootnote}[3]{%
7209 \footnote{#3}%
7210 }

```

`acronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```

7211 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7212 \defglsentryfmt[#1]{%
7213 \ifdefempty\glscustomtext
7214 {%
7215 \ifglsused{\glslabel}%
7216 {%
7217 \acronymfont{\glsgenentryfmt}%
7218 }%
7219 {%
7220 \firstacronymfont{\glsgenentryfmt}%
7221 \ifgls hassymbol{\glslabel}%
7222 {%
7223 \expandafter\protect\expandafter\acrfootnote\expandafter
7224 {\@gls@link@opts}{\@gls@link@label}%
7225 {%
7226 \glsifplural
7227 {\glsentrysymbolplural{\glslabel}}%
7228 {\glsentrysymbol{\glslabel}}%
7229 }%
7230 }%
7231 }%
7232 }%
7233 {\glscustomtext\glsinsert}%
7234 }%
7235 }

```


teNewAcronymDef

```

7236 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7237   \edef\@do@newglossaryentry{%
7238     \noexpand\newglossaryentry{\the\glslabeltok}%
7239     {%
7240       type=\acronymtype,%
7241       name={\noexpand\acronymfont{\the\glsshorttok}},%
7242       sort={\the\glsshorttok},%
7243       first={\the\glsshorttok},%
7244       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7245       text={\the\glsshorttok},%
7246       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7247       short={\the\glsshorttok},%
7248       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7249       long={\the\glslongtok},%
7250       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7251       symbol={\the\glslongtok},%
7252       symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7253       \the\glskeylisttok
7254     }%
7255   }%
7256   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7257   \let\@org@gls@assign@plural\gls@assign@plural
7258   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7259   \def\gls@assign@firstpl##1##2{%
7260     \@@gls@expand@field{##1}{firstpl}{##2}%
7261   }%
7262   \def\gls@assign@plural##1##2{%
7263     \@@gls@expand@field{##1}{plural}{##2}%
7264   }%
7265   \def\gls@assign@symbolplural##1##2{%
7266     \@@gls@expand@field{##1}{symbolplural}{##2}%
7267   }%
7268   \@do@newglossaryentry
7269   \let\gls@assign@plural\@org@gls@assign@plural
7270   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7271   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7272 }

```

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.

```

7273 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7274   \renewcommand{\newacronym}[4][\]{%
7275     \ifx\@glsacronymlists\@empty
7276       \def\@glo@type{\acronymtype}%
7277       \setkeys{glossentry}{##1}%
7278       \DeclareAcronymList{\@glo@type}%

```

```

7279     \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7280     \fi
7281     \glskeylisttok{##1}%
7282     \glslabeltok{##2}%
7283     \glsshorttok{##3}%
7284     \glslongtok{##4}%
7285     \newacronymhook
7286     \DescriptionFootnoteNewAcronymDef
7287 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7288 \@for\@gls@type:=\@glsacronymlists\do{%
7289     \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7290 }%

```

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.

```

7291 \ifglsacrsmallcaps
7292     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7293     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7294 \else
7295     \ifglsacrsmaller
7296         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7297     \fi
7298 \fi

```

Check for package option clash

```

7299 \ifglsacrdua
7300     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7301     can’t both be set}{}%
7302 \fi
7303 }%

```

`nymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.

```

7304 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7305     \defglsentryfmt[##1]{\glsngenentryfmt}%
7306 }

```

`UANewAcronymDef`

```

7307 \newcommand*{\DescriptionDUANewAcronymDef}{%
7308     \edef\@do@newglossaryentry{%
7309         \noexpand\newglossaryentry{\the\glslabeltok}%
7310         {%
7311             type=\acronymtype,%
7312             name={\the\glslongtok},%
7313             sort={\the\glslongtok},%
7314             text={\the\glslongtok},%
7315             first={\the\glslongtok},%

```

```

7316 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7317 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7318 short={\the\glsshorttok},%
7319 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7320 long={\the\glslongtok},%
7321 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7322 symbol={\the\glsshorttok},%
7323 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7324 \the\glskeylisttok
7325 }%
7326 }%
7327 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7328 \let\@org@gls@assign@plural\gls@assign@plural
7329 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7330 \def\gls@assign@firstpl##1##2{%
7331   \@gls@expand@field{##1}{firstpl}{##2}%
7332 }%
7333 \def\gls@assign@plural##1##2{%
7334   \@gls@expand@field{##1}{plural}{##2}%
7335 }%
7336 \def\gls@assign@symbolplural##1##2{%
7337   \@gls@expand@field{##1}{symbolplural}{##2}%
7338 }%
7339 \do@newglossaryentry
7340 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7341 \let\gls@assign@plural\@org@gls@assign@plural
7342 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7343 }

```

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.

```

7344 \newcommand*{\SetDescriptionDUAACronymStyle}{%
7345   \ifglssacrsmallcaps
7346     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7347       can't both be set}{}%
7348   \else
7349     \ifglssacrsmaller
7350       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7351         can't both be set}{}%
7352     \fi
7353   \fi
7354   \renewcommand{\newacronym}[4][{}]{%
7355     \ifx\@glsacronymlists\@empty
7356       \def\@glo@type{\acronymtype}%
7357       \setkeys{glossentry}{##1}%
7358       \DeclareAcronymList{\@glo@type}%
7359       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
7360     \fi

```

```

7361 \glskeylisttok{##1}%
7362 \glslabeltok{##2}%
7363 \glsshorttok{##3}%
7364 \glslongtok{##4}%
7365 \newacronymhook
7366 \DescriptionDUANewAcronymDef
7367 }%

```

Set display.

```

7368 \@for\@gls@type:=\@glsacronymlists\do{%
7369 \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
7370 }%
7371 }%

```

`\glsDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7372 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7373 \defglsentryfmt[#1]{%

7374 \ifdefempty\glscustomtext
7375 {%
7376 \ifglsused{\glslabel}%
7377 {%

```

Move the inserted text outside of `\acronymfont`

```

7378 \let\gls@org@insert\glsinsert
7379 \let\glsinsert\@empty
7380 \acronymfont{\glsentryfmt}\gls@org@insert
7381 }%
7382 {%
7383 \glsentryfmt
7384 \ifgls hassymbol{\glslabel}%
7385 {%
7386 \glsifplural
7387 {%
7388 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7389 }%
7390 {%
7391 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7392 }%
7393 \space(\protect\firstacronymfont
7394 {\glsupercase
7395 {\@glo@symbol}
7396 {\@glo@symbol}
7397 {\mfirstucMakeUppercase{\@glo@symbol}}}))%
7398 }%
7399 {}%
7400 }%
7401 }%
7402 {\glscustomtext\glsinsert}%

```

```

7403 }%
7404 }

```

onNewAcronymDef

```

7405 \newcommand*{\DescriptionNewAcronymDef}{%
7406   \edef\@do@newglossaryentry{%
7407     \noexpand\newglossaryentry{\the\glslabeltok}%
7408     {%
7409       type=\acronymtype,%
7410       name={\noexpand
7411         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
7412       sort={\the\glsshorttok},%
7413       first={\the\glslongtok},%
7414       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7415       text={\the\glsshorttok},%
7416       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7417       short={\the\glsshorttok},%
7418       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7419       long={\the\glslongtok},%
7420       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7421       symbol={\noexpand\@glo@text},%
7422       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7423       \the\glskeylisttok}%
7424   }%
7425   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7426   \let\@org@gls@assign@plural\gls@assign@plural
7427   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7428   \def\gls@assign@firstpl##1##2{%
7429     \@@gls@expand@field{##1}{firstpl}{##2}%
7430   }%
7431   \def\gls@assign@plural##1##2{%
7432     \@@gls@expand@field{##1}{plural}{##2}%
7433   }%
7434   \def\gls@assign@symbolplural##1##2{%
7435     \@@gls@expand@field{##1}{symbolplural}{##2}%
7436   }%
7437   \@do@newglossaryentry
7438   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7439   \let\gls@assign@plural\@org@gls@assign@plural
7440   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7441 }

```

ionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using \acrnameformat to allow the user to override the way the name is displayed in the list of acronyms.

```

7442 \newcommand*{\SetDescriptionAcronymStyle}{%
7443   \renewcommand{\newacronym}[4][\]{%
7444     \ifx\@glsacronymlists\empty
7445       \def\@glo@type{\acronymtype}%

```

```

7446     \setkeys{glossentry}{##1}%
7447     \DeclareAcronymList{\@glo@type}%
7448     \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7449     \fi
7450     \glskeylisttok{##1}%
7451     \glslabeltok{##2}%
7452     \glsshorttok{##3}%
7453     \glslongtok{##4}%
7454     \newacronymhook
7455     \DescriptionNewAcronymDef
7456 }%

```

Set display.

```

7457 \@for\@gls@type:=\@glsacronymlists\do{%
7458   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7459 }%

```

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.

```

7460 \ifglsacrsmallcaps
7461   \renewcommand{\acronymfont}[1]{\textsc{##1}}
7462   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7463 \else
7464   \ifglsacrsmaller
7465     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7466   \fi
7467 \fi
7468 }%

```

`\nymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7469 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7470   \defglsentryfmt[#1]{%
7471     \ifdefempty\glscustomtext
7472     {%

```

Move the inserted text outside of `\acronymfont`

```

7473     \let\gls@org@insert\glsinsert
7474     \let\glsinsert\@empty
7475     \ifglsused{\glslabel}%
7476     {%
7477       \acronymfont{\glsentryfmt}\gls@org@insert
7478     }%
7479     {%
7480       \firstacronymfont{\glsentryfmt}\gls@org@insert
7481       \ifglschaslong{\glslabel}%
7482       {%
7483         \expandafter\protect\expandafter\acrfootnote\expandafter
7484         {\@gls@link@opts}{\@gls@link@label}%

```

```

7485         {%
7486         \glsifplural
7487         {\glsentrylongpl{\glslabel}}}%
7488         {\glsentrylong{\glslabel}}}%
7489     }%
7490 }%

7491     {%
7492     }%
7493 }%
7494 {\glscustomtext\glsinsert}%
7495 }%
7496 }

```

teNewAcronymDef

```

7497 \newcommand*{\FootnoteNewAcronymDef}{%
7498 \edef\@do@newglossaryentry{%
7499 \noexpand\newglossaryentry{\the\glslabeltok}%
7500 {%
7501     type=\acronymtype,%
7502     name={\noexpand\acronymfont{\the\glsshorttok}},%
7503     sort={\the\glsshorttok},%
7504     text={\the\glsshorttok},%
7505     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7506     first={\the\glsshorttok},%
7507     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7508     short={\the\glsshorttok},%
7509     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7510     long={\the\glslongtok},%
7511     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7512     description={\the\glslongtok},%
7513     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7514     \the\glskeylisttok
7515 }%
7516 }%
7517 \let\@org@gls@assign@plural\gls@assign@plural
7518 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7519 \let\@org@gls@assign@descplural\gls@assign@descplural
7520 \def\gls@assign@firstpl##1##2{%
7521 \@@gls@expand@field{##1}{firstpl}{##2}%
7522 }%
7523 \def\gls@assign@plural##1##2{%
7524 \@@gls@expand@field{##1}{plural}{##2}%
7525 }%
7526 \def\gls@assign@descplural##1##2{%
7527 \@@gls@expand@field{##1}{descplural}{##2}%
7528 }%
7529 \@do@newglossaryentry
7530 \let\gls@assign@plural\@org@gls@assign@plural
7531 \let\gls@assign@firstpl\@org@gls@assign@firstpl

```

```

7532 \let\gls@assign@descplural\@org@gls@assign@descplural
7533 }

```

`\setfootnoteacronymstyle` 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.

```

7534 \newcommand*{\SetFootnoteAcronymStyle}{%
7535   \renewcommand{\newacronym}[4][]{%
7536     \ifx\@glsacronymlists\@empty
7537       \def\@glo@type{\acronymtype}%
7538       \setkeys{glossentry}{##1}%
7539       \DeclareAcronymList{\@glo@type}%
7540       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7541     \fi
7542     \glskeylisttok{##1}%
7543     \glslabeltok{##2}%
7544     \glsshorttok{##3}%
7545     \glslongtok{##4}%
7546     \newacronymhook
7547     \FootnoteNewAcronymDef
7548   }%

```

Set display

```

7549 \@for\@gls@type:=\@glsacronymlists\do{%
7550   \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7551 }%

```

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.

```

7552 \ifglsacrsmallcaps
7553   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7554   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7555 \else
7556   \ifglsacrsmaller
7557     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7558   \fi
7559 \fi

```

Check for option clash

```

7560 \ifglsacrdua
7561   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7562     can’t both be set}{}%
7563 \fi
7564 }%

```

`\doparenifnotempty` 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.

```

7565 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7566   \protected@edef\gls@tmp{##1}%
7567   \ifdefempty\gls@tmp
7568     {}%

```



```

7569  {%
7570    \ifx\gls@tmp\@gls@default@value
7571    \else
7572      \space (#2{#1})%
7573    \fi
7574  }%
7575 }

```

nymDisplayStyle Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7576 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7577   \defglsentryfmt[#1]{%

```

```

7578     \ifdefempty\glscustomtext
7579     {%

```

Move the inserted text outside of \acronymfont

```

7580     \let\gls@org@insert\glsinsert
7581     \let\glsinsert\@empty
7582     \ifglsused{\glslabel}%
7583     {%
7584       \acronymfont{\glsentryfmt}\gls@org@insert
7585     }%
7586     {%
7587       \glsentryfmt
7588       \ifgls hassymbol{\glslabel}%
7589       {%
7590         \glsifplural
7591         {%
7592           \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7593         }%
7594         {%
7595           \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7596         }%
7597         \space
7598         (\glscapscase
7599           {\firstacronymfont{\@glo@symbol}}%
7600           {\firstacronymfont{\@glo@symbol}}%
7601           {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7602         }%
7603       }%
7604     }%
7605   }%
7606   {\glscustomtext\glsinsert}%
7607 }%
7608 }

```

l1NewAcronymDef

```

7609 \newcommand*{\SmallNewAcronymDef}{%

```

```

7610 \edef\@do@newglossaryentry{%
7611   \noexpand\newglossaryentry{\the\glslabeltok}%
7612   {%
7613     type=\acronymtype,%
7614     name={\noexpand\acronymfont{\the\glsshorttok}},%
7615     sort={\the\glsshorttok},%
7616     text={\the\glsshorttok},%

    Default to the short plural.
7617     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7618     first={\the\glslongtok},%

    Default to the long plural.
7619     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7620     short={\the\glsshorttok},%
7621     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7622     long={\the\glslongtok},%
7623     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7624     description={\noexpand\@glo@first},%
7625     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7626     symbol={\the\glsshorttok},%

    Default to the short plural.
7627     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7628     \the\glskeylisttok
7629   }%
7630 }%
7631 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7632 \let\@org@gls@assign@plural\gls@assign@plural
7633 \let\@org@gls@assign@descplural\gls@assign@descplural
7634 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7635 \def\gls@assign@firstpl##1##2{%
7636   \@gls@expand@field{##1}{firstpl}{##2}%
7637 }%
7638 \def\gls@assign@plural##1##2{%
7639   \@gls@expand@field{##1}{plural}{##2}%
7640 }%
7641 \def\gls@assign@descplural##1##2{%
7642   \@gls@expand@field{##1}{descplural}{##2}%
7643 }%
7644 \def\gls@assign@symbolplural##1##2{%
7645   \@gls@expand@field{##1}{symbolplural}{##2}%
7646 }%
7647 \@do@newglossaryentry
7648 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7649 \let\gls@assign@plural\@org@gls@assign@plural
7650 \let\gls@assign@descplural\@org@gls@assign@descplural
7651 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7652 }

```

`\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.

```

7653 \newcommand*{\SetSmallAcronymStyle}{%
7654   \renewcommand{\newacronym}[4][\{%
7655     \ifx\@glsacronymlists\@empty
7656       \def\@glo@type{\acronymtype}%
7657       \setkeys{glossentry}{##1}%
7658       \DeclareAcronymList{\@glo@type}%
7659       \SetSmallAcronymDisplayStyle{\@glo@type}%
7660     \fi
7661     \glskeylisttok{##1}%
7662     \glslabeltok{##2}%
7663     \glsshorttok{##3}%
7664     \glslongtok{##4}%
7665     \newacronymhook
7666     \SmallNewAcronymDef
7667   }%

```

Change the display since first only contains long form.

```

7668   \@for\@gls@type:=\@glsacronymlists\do{%
7669     \SetSmallAcronymDisplayStyle{\@gls@type}%
7670   }%

```

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.

```

7671   \ifglsacrsmallcaps
7672     \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7673     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7674   \else
7675     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
7676   \fi

```

check for option clash

```

7677   \ifglsacrdua
7678     \ifglsacrsmallcaps
7679       \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7680         can’t both be set}{}%
7681     \else
7682       \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7683         can’t both be set}{}%
7684     \fi
7685   \fi
7686 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7687 \newcommand*{\SetDUADisplayStyle}[1]{%
7688   \defglsentryfmt[##1]{\glsentryfmt}%
7689 }

```

UANewAcronymDef

```

7690 \newcommand*{\DUANewAcronymDef}{%

```

```

7691 \edef\@do@newglossaryentry{%
7692   \noexpand\newglossaryentry{\the\glslabeltok}%
7693   {%
7694     type=\acronymtype,%
7695     name={\the\glsshorttok},%
7696     text={\the\glslongtok},%
7697     first={\the\glslongtok},%
7698     plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7699     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7700     short={\the\glsshorttok},%
7701     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7702     long={\the\glslongtok},%
7703     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7704     description={\the\glslongtok},%
7705     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7706     symbol={\the\glsshorttok},%
7707     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7708     \the\glskeylisttok
7709   }%
7710 }%
7711 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7712 \let\@org@gls@assign@plural\gls@assign@plural
7713 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7714 \let\@org@gls@assign@descplural\gls@assign@descplural
7715 \def\gls@assign@firstpl##1##2{%
7716   \@gls@expand@field{##1}{firstpl}{##2}%
7717 }%
7718 \def\gls@assign@plural##1##2{%
7719   \@gls@expand@field{##1}{plural}{##2}%
7720 }%
7721 \def\gls@assign@symbolplural##1##2{%
7722   \@gls@expand@field{##1}{symbolplural}{##2}%
7723 }%
7724 \def\gls@assign@descplural##1##2{%
7725   \@gls@expand@field{##1}{descplural}{##2}%
7726 }%
7727 \@do@newglossaryentry
7728 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7729 \let\gls@assign@plural\@org@gls@assign@plural
7730 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7731 \let\gls@assign@descplural\@org@gls@assign@descplural
7732 }

```

\SetDUASStyle Always expand acronyms.

```

7733 \newcommand*{\SetDUASStyle}{%
7734   \renewcommand{\newacronym}[4][]{%
7735     \ifx\@glsacronymlists\empty
7736       \def\@glo@type{\acronymtype}%
7737       \setkeys{glossentry}{##1}%

```

```

7738     \DeclareAcronymList{\@glo@type}%
7739     \SetDUADisplayStyle{\@glo@type}%
7740     \fi
7741     \glskeylisttok{##1}%
7742     \glslabeltok{##2}%
7743     \glsshorttok{##3}%
7744     \glslongtok{##4}%
7745     \newacronymhook
7746     \DUANewAcronymDef
7747 }%

Set the display
7748 \@for\@gls@type:=\@glsacronymlists\do{%
7749     \SetDUADisplayStyle{\@gls@type}%
7750 }%
7751 }

```

SetAcronymStyle

```

7752 \newcommand*{\SetAcronymStyle}{%
7753     \SetDefaultAcronymStyle
7754     \ifglsacrdescription
7755         \ifglsacrfootnote
7756             \SetDescriptionFootnoteAcronymStyle
7757         \else
7758             \ifglsacrdua
7759                 \SetDescriptionDUAAcronymStyle
7760             \else
7761                 \SetDescriptionAcronymStyle
7762             \fi
7763         \fi
7764     \else
7765         \ifglsacrfootnote
7766             \SetFootnoteAcronymStyle
7767         \else
7768             \ifthenelse{\boolean{glsacrsmalldcaps}\OR
7769                 \boolean{glsacrsmaller}}{%
7770                 {%
7771                     \SetSmallAcronymStyle
7772                 }%
7773                 {%
7774                     \ifglsacrdua
7775                         \SetDUASyle
7776                     \fi
7777                 }%
7778             \fi
7779         \fi
7780 }

```

Set the acronym style according to the package options

```

7781 \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.

```
7782 \newcommand*{\SetCustomDisplayStyle}[1]{%
7783   \def\glsentryfmt[#1]{\glsentryfmt}%
7784 }
```

`\CustomAcronymFields`

```
7785 \newcommand*{\CustomAcronymFields}{%
7786   name={\the\glsshorttok},%
7787   description={\the\glslongtok},%
7788   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7789   firstplural={\acrfullformat
7790     {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7791     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

7792   text={\the\glsshorttok},%
7793   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7794 }
```

`\CustomNewAcronymDef`

```
7795 \newcommand*{\CustomNewAcronymDef}{%
7796   \protected@edef\@do@newglossaryentry{%
7797     \noexpand\newglossaryentry{\the\glslabeltok}%
7798     {%
7799       type=\acronymtype,%
7800       short={\the\glsshorttok},%
7801       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7802       long={\the\glslongtok},%
7803       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7804       user1={\the\glsshorttok},%
7805       user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7806       user3={\the\glslongtok},%
7807       user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7808       \CustomAcronymFields,%
7809       \the\glskeylisttok
7810     }%
7811   }%
7812   \@do@newglossaryentry
7813 }
```

`\SetCustomStyle`

```
7814 \newcommand*{\SetCustomStyle}{%
7815   \renewcommand{\newacronym}[4][1]{%
7816     \ifx\@glsacronymlists\@empty
7817       \def\@glo@type{\acronymtype}%
7818     \fi
7819   }
```

```

7818     \setkeys{glossentry}{##1}%
7819     \DeclareAcronymList{\@glo@type}%
7820     \SetCustomDisplayStyle{\@glo@type}%
7821   \fi
7822   \glskeylisttok{##1}%
7823   \glslabeltok{##2}%
7824   \glsshorttok{##3}%
7825   \glslongtok{##4}%
7826   \newacronymhook
7827   \CustomNewAcronymDef
7828 }%

Set the display
7829 \@for\@gls@type:=\@glsacronymlists\do{%
7830   \SetCustomDisplayStyle{\@gls@type}%
7831 }%
7832 }

```

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.

```
7833 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7834 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `nolong` package option is used.

```
7835 \@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.

```
7836 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
7837 \@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.

```

7838 \ifx\@glossary@default@style\relax
7839 \else
7840   \setglossarystyle{\@glossary@default@style}
7841 \fi

```

1.20 Debugging Commands

`\showgloparent` `\showgloparent{<label>}`

```
7842 \newcommand*{\showgloparent}[1]{%
7843   \expandafter\show\csname glo@glstetoklabel{#1}@parent\endcsname
7844 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7845 \newcommand*{\showglolevel}[1]{%
7846   \expandafter\show\csname glo@glstetoklabel{#1}@level\endcsname
7847 }
```

`\showglotext` `\showglotext{<label>}`

```
7848 \newcommand*{\showglotext}[1]{%
7849   \expandafter\show\csname glo@glstetoklabel{#1}@text\endcsname
7850 }
```

`\showgloplural` `\showgloplural{<label>}`

```
7851 \newcommand*{\showgloplural}[1]{%
7852   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7853 }
```

`\showglofirst` `\showglofirst{<label>}`

```
7854 \newcommand*{\showglofirst}[1]{%
7855   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7856 }
```

`\showglofirstpl` `\showglofirstpl{<label>}`

```
7857 \newcommand*{\showglofirstpl}[1]{%
7858   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7859 }
```


\showglotype \showglotype{<label>}

```
7860 \newcommand*{\showglotype}[1]{%
7861   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7862 }
```

\showglocounter \showglocounter{<label>}

```
7863 \newcommand*{\showglocounter}[1]{%
7864   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7865 }
```

\showglouser \showglouser{<label>}

```
7866 \newcommand*{\showglouser}[1]{%
7867   \expandafter\show\csname glo@glstetoklabel{#1}@user\endcsname
7868 }
```

\showglouserii \showglouserii{<label>}

```
7869 \newcommand*{\showglouserii}[1]{%
7870   \expandafter\show\csname glo@glstetoklabel{#1}@userii\endcsname
7871 }
```

\showglouseriii \showglouseriii{<label>}

```
7872 \newcommand*{\showglouseriii}[1]{%
7873   \expandafter\show\csname glo@glstetoklabel{#1}@useriii\endcsname
7874 }
```

\showglouseriv \showglouseriv{<label>}

```
7875 \newcommand*{\showglouseriv}[1]{%
7876   \expandafter\show\csname glo@glstetoklabel{#1}@useriv\endcsname
7877 }
```

\showglouserv \showglouserv{<label>}

```
7878 \newcommand*{\showglouserv}[1]{%
7879   \expandafter\show\csname glo@glstdetoklabel{#1}@user\endcsname
7880 }
```

\showglouservi \showglouservi{<label>}

```
7881 \newcommand*{\showglouservi}[1]{%
7882   \expandafter\show\csname glo@glstdetoklabel{#1}@user\endcsname
7883 }
```

\showgloname \showgloname{<label>}

```
7884 \newcommand*{\showgloname}[1]{%
7885   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7886 }
```

\showglodesc \showglodesc{<label>}

```
7887 \newcommand*{\showglodesc}[1]{%
7888   \expandafter\show\csname glo@glstdetoklabel{#1}@desc\endcsname
7889 }
```

\showglodescplural \showglodescplural{<label>}

```
7890 \newcommand*{\showglodescplural}[1]{%
7891   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7892 }
```

\showglosort \showglosort{<label>}

```
7893 \newcommand*{\showglosort}[1]{%
7894   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7895 }
```

`\showglosymbol` `\showglosymbol{<label>}`

```
7896 \newcommand*{\showglosymbol}[1]{%
7897   \expandafter\show\csname glo@glstetoklabel{#1}@symbol\endcsname
7898 }
```

`wglosymbolplural` `\showglosymbolplural{<label>}`

```
7899 \newcommand*{\showglosymbolplural}[1]{%
7900   \expandafter\show\csname glo@glstetoklabel{#1}@symbolplural\endcsname
7901 }
```

`\showgloshort` `\showgloshort{<label>}`

```
7902 \newcommand*{\showgloshort}[1]{%
7903   \expandafter\show\csname glo@glstetoklabel{#1}@short\endcsname
7904 }
```

`\showglolong` `\showglolong{<label>}`

```
7905 \newcommand*{\showglolong}[1]{%
7906   \expandafter\show\csname glo@glstetoklabel{#1}@long\endcsname
7907 }
```

`\showgloindex` `\showgloindex{<label>}`

```
7908 \newcommand*{\showgloindex}[1]{%
7909   \expandafter\show\csname glo@glstetoklabel{#1}@index\endcsname
7910 }
```

`\showgloflag` `\showgloflag{<label>}`

```
7911 \newcommand*{\showgloflag}[1]{%
7912   \expandafter\show\csname ifglo@glstetoklabel{#1}@flag\endcsname
7913 }
```

`\showgloclist` `\showgloclist{<label>}`

```
7914 \newcommand*{\showgloclist}[1]{%
7915   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7916 }
```

`\showglofield` `\showglofield{<label>}{<field>}`

```
7917 \newcommand*{\showglofield}[2]{%
7918   \csshow{glo@\glsdetoklabel{#1}@#2}%
7919 }
```

`showacronymlists` `\showacronymlists`

Show list of glossaries that have been flagged as a list of acronyms.

```
7920 \newcommand*{\showacronymlists}{%
7921   \show\@glsacronymlists
7922 }
```

`\showglossaries` `\showglossaries`

Show list of defined glossaries.

```
7923 \newcommand*{\showglossaries}{%
7924   \show\@glo@types
7925 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7926 \newcommand*{\showglossaryin}[1]{%
7927   \expandafter\show\csname @glotype@#1@in\endcsname
7928 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7929 \newcommand*{\showglossaryout}[1]{%
7930   \expandafter\show\csname @glotype@#1@out\endcsname
7931 }
```

showglossarytitle

```
\showglossarytitle{<glossary-label>}
```

Show the title for the given glossary.

```
7932 \newcommand*{\showglossarytitle}[1]{%
7933   \expandafter\show\csname @glotype@#1@title\endcsname
7934 }
```

showglossarycounter

```
\showglossarycounter{<glossary-label>}
```

Show the counter for the given glossary.

```
7935 \newcommand*{\showglossarycounter}[1]{%
7936   \expandafter\show\csname @glotype@#1@counter\endcsname
7937 }
```

showglossaryentries

```
\showglossaryentries{<glossary-label>}
```

Show the list of entry labels for the given glossary.

```
7938 \newcommand*{\showglossaryentries}[1]{%
7939   \expandafter\show\csname glolist@#1\endcsname
7940 }
```

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.

```
7941 \csname ifglcompatible-2.07\endcsname
7942 \RequirePackage{glossaries-compatible-207}
7943 \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 “`\a\gls{<label>}`” on first use but use “`\an\gls{<label>}`” on subsequent use.

```
7944 \NeedsTeXFormat{LaTeX2e}
```

```
7945 \ProvidesPackage{glossaries-prefix}[2018/07/23 v4.41 (NLCT)]
```

Pass all options to glossaries:

```
7946 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7947 \ProcessOptions
```

Load glossaries:

```
7948 \RequirePackage{glossaries}
```

Add the new keys:

```
7949 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7950 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7951 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7952 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7953 \appto\@gls@keymap{,%
```

```
7954   {prefixfirst}{prefixfirst},%
```

```
7955   {prefixfirstplural}{prefixfirstplural},%
```

```
7956   {prefix}{prefix},%
```

```
7957   {prefixplural}{prefixplural}}%
```

```
7958 }
```

Set the default values:

```
7959 \appto\@newglossaryentryprehook{%
```

```
7960   \def\@glo@entryprefix{}
```

```
7961   \def\@glo@entryprefixplural{}
```

```
7962   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7963   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7964 }
```

Set the assignment code:

```
7965 \appto\@newglossaryentryposthook{%
```

```
7966   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}%
```

```
7967   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7968   \expandafter\gls@assign@field\expandafter
```

```
7969     {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
```

```
7970     {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7971 \expandafter\gls@assign@field\expandafter
7972   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7973   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7974 }

```

Define commands to access these fields:

entryprefixfirst

```

7975 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7976 \newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7977 \newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7978 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7979 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
7980   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7981   \xmakefirstuc\@glo@text
7982 }

```

entryfirstplural

```

7983 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7984   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7985   \xmakefirstuc\@glo@text
7986 }

```

\Glsentryprefix

```

7987 \newrobustcmd*{\Glsentryprefix}[1]{%
7988   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7989   \xmakefirstuc\@glo@text
7990 }

```

entryprefixplural

```

7991 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7992   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7993   \xmakefirstuc\@glo@text
7994 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7995 \newcommand*{\ifglshasprefix}[3]{%
7996   \ifcempty{glo@#1@prefix}%
7997   {#3}%
7998   {#2}%
7999 }
```

hasprefixplural

```
8000 \newcommand*{\ifglshasprefixplural}[3]{%
8001   \ifcempty{glo@#1@prefixplural}%
8002   {#3}%
8003   {#2}%
8004 }
```

shasprefixfirst

```
8005 \newcommand*{\ifglshasprefixfirst}[3]{%
8006   \ifcempty{glo@#1@prefixfirst}%
8007   {#3}%
8008   {#2}%
8009 }
```

efixfirstplural

```
8010 \newcommand*{\ifglshasprefixfirstplural}[3]{%
8011   \ifcempty{glo@#1@prefixfirstplural}%
8012   {#3}%
8013   {#2}%
8014 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
8015 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
8016 \newcommand*{\@pgls}[2][ ]{%
8017   \new@ifnextchar[%
8018   {\@pgls@{#1}{#2}}%
8019   {\@pgls@{#1}{#2}[ ]}%
8020 }
```

\@pgls@ Read in the final optional argument:

```
8021 \def\@pgls@#1#2[#3]{%
8022   \glsdoifexists{#2}%
8023   {%
8024     \ifglsused{#2}%
8025     {%
8026       \glstryprefix{#2}%
8027     }%
8028   }
```



```

8028     {%
8029     \glsentryprefixfirst{#2}%
8030     }%
8031     \@gls@{#1}{#2}[#3]%
8032     }%
8033 }

```

Similarly for the plural version:

```

\pglsp1
8034 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

8035 \newcommand*{\@pglsp1}[2][ ]{%
8036   \new@ifnextchar[%
8037   {\@pglsp1@{#1}{#2}}%
8038   {\@pglsp1@{#1}{#2}[]}%
8039 }

```

\@pglsp1@ Read in the final optional argument:

```

8040 \def\@pglsp1@#1#2[#3]{%
8041   \glsdoifexists{#2}%
8042   {%
8043     \ifglsused{#2}%
8044     {%
8045       \glsentryprefixplural{#2}%
8046     }%
8047     {%
8048       \glsentryprefixfirstplural{#2}%
8049     }%
8050     \@glspl@{#1}{#2}[#3]%
8051   }%
8052 }

```

Now for the first letter upper case versions:

```

\Pgls
8053 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

8054 \newcommand*{\@Pgls}[2][ ]{%
8055   \new@ifnextchar[%
8056   {\@Pgls@{#1}{#2}}%
8057   {\@Pgls@{#1}{#2}[]}%
8058 }

```

\@Pgls@ Read in the final optional argument:

```

8059 \def\@Pgls@#1#2[#3]{%

```

```

8060 \glsdoifexists{#2}%
8061 {%
8062   \ifglsused{#2}%
8063   {%
8064     \ifglshasprefix{#2}%
8065     {%
8066       \Glsentryprefix{#2}%
8067       \@gls@{#1}{#2}[#3]%
8068     }%
8069     {\@Gls@{#1}{#2}[#3]}%
8070   }%
8071   {%
8072     \ifglshasprefixfirst{#2}%
8073     {%
8074       \Glsentryprefixfirst{#2}%
8075       \@gls@{#1}{#2}[#3]%
8076     }%
8077     {\@Gls@{#1}{#2}[#3]}%
8078   }%
8079 }%
8080 }

```

Similarly for the plural version:

```

\Pglspl
8081 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\Pglspl}

```

\@Pglspl Unstarred version.

```

8082 \newcommand*{\@Pglspl}[2] [] {%
8083   \new@ifnextchar[%
8084   {\@Pglspl@{#1}{#2}}%
8085   {\@Pglspl@{#1}{#2} []}%
8086 }

```

\@Pglspl@ Read in the final optional argument:

```

8087 \def\@Pglspl@#1#2[#3] {%
8088   \glsdoifexists{#2}%
8089   {%
8090     \ifglsused{#2}%
8091     {%
8092       \ifglshasprefixplural{#2}%
8093       {%
8094         \Glsentryprefixplural{#2}%
8095         \@glspl@{#1}{#2}[#3]%
8096       }%
8097       {\@Glspl@{#1}{#2}[#3]}%
8098     }%
8099     {%
8100       \ifglshasprefixfirstplural{#2}%

```

```

8101      {%
8102          \Glsentryprefixfirstplural{#2}%
8103          \@glspl@{#1}{#2}[#3]%
8104      }%
8105      {\@Glspl@{#1}{#2}[#3]}%
8106  }%
8107 }%
8108 }

```

Finally the all upper case versions:

\PGLS

```

8109 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

8110 \newcommand*{\@PGLS}[2][{}]{%
8111   \new@ifnextchar[%
8112   {\@PGLS@{#1}{#2}}%
8113   {\@PGLS@{#1}{#2}[]}%
8114 }

```

\@PGLS@ Read in the final optional argument:

```

8115 \def\@PGLS@#1#2[#3]{%
8116   \glsdoifexists{#2}%
8117   {%
8118     \ifglsused{#2}%
8119     {%
8120       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
8121     }%
8122     {%
8123       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
8124     }%
8125     \@GLS@{#1}{#2}[#3]%
8126   }%
8127 }

```

Plural version:

\PGLSp1

```

8128 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

8129 \newcommand*{\@PGLSp1}[2][{}]{%
8130   \new@ifnextchar[%
8131   {\@PGLSp1@{#1}{#2}}%
8132   {\@PGLSp1@{#1}{#2}[]}%
8133 }

```

\@PGLSp1@ Read in the final optional argument:

```
8134 \def\@PGLSp1@#1#2[#3]{%
8135   \glsdoifexists{#2}%
8136   {%
8137     \ifglsused{#2}%
8138     {%
8139       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
8140     }%
8141     {%
8142       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
8143     }%
8144     \@GLSp1@{#1}{#2}[#3]%
8145   }%
8146 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8147 \ProvidesPackage{glossary-hypernav}[2018/07/23 v4.41 (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`

```
8148 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8149   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8150   \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

`avhyperlinkname`

Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
8151 \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.

`snavhypertarget`

```
8152 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
8153   \@glsnavhypertarget{#1}{#2}{#3}%
8154 }
```

The actual code is now in an internal command that doesn't have an optional argument, which makes it easier to save and restore the original behaviour.

`snavhypertarget`

```
8155 \newcommand*{\@glsnavhypertarget}[3]{%
```

Add this group to the aux file for re-run check.

```
8156 \protected@write\auxout{}\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
8157 \glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8158 \expandafter\let
```

```
8159 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8160 \@for\@gls@elem:=\@gls@list\do{%
```

```
8161 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
8162 \if@endfor
```

```
8163 \else
```

This group was not included in the list, so issue a warning.

```
8164 \GlossariesWarningNoLine{Navigation panel
```

```
8165 for glossary type ‘#1’~Jmissing group ‘#2’}%
```

```
8166 \gdef\gls@hypergroup@rerun{%
```

```
8167 \GlossariesWarningNoLine{Navigation panel
```

```
8168 has changed. Rerun LaTeX}}%
```

```
8169 \fi
```

```
8170 }
```

`hypergroup@rerun` Give a warning at the end if re-run required

```
8171 \let\gls@hypergroup@rerun\relax
```

```
8172 \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.

```
8173 \newcommand*{\@gls@hypergroup}[2]{%
```

```
8174 \@ifundefined{\@gls@hypergroup@list@#1}{%
```

```
8175 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}}%
```

```
8176 }{%
```

```
8177 \expandafter\let\expandafter\@gls@tmp
```

```
8178 \csname @gls@hypergroup@list@#1\endcsname
```

```
8179 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
```

```
8180 \@gls@tmp,#2}}%
```

```
8181 }%
```

```
8182 }
```

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`

```
8183 \newcommand*{\glsnavigation}{%
8184   \def\@gls@between{}%
8185   \ifcsundef{\@gls@hypergroupplist@\@glo@type}%
8186   {%
8187     \def\@gls@list{}%
8188   }%
8189   {%
8190     \expandafter\let\expandafter\@gls@list
8191       \csname \@gls@hypergroupplist@\@glo@type\endcsname
8192   }%
8193   \@for\@gls@tmp:=\@gls@list\do{%
8194     \@gls@between
8195     \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8196     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8197     \let\@gls@between\glshypernavsep
8198   }%
8199 }
```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8200 \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`

```
8201 \newcommand*{\glssymbolnav}{%
8202   \glsnavhyperlink{glssymbols}{\@gls@getgrouptitle{glssymbols}}%
8203   \glshypernavsep
8204   \glsnavhyperlink{glsnumbers}{\@gls@getgrouptitle{glsnumbers}}%
8205   \glshypernavsep
8206 }
```

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.

```
8207 \ProvidesPackage{glossary-inline}[2018/07/23 v4.41 (NLCT)]
```

`inline` Define the inline style.

```
8208 \newglossarystyle{inline}{%
  Start of glossary sets up first empty separator between entries. (This is then changed by
  \glossentry)
8209   \renewenvironment{theglossary}%
8210   {%
```

```

8211     \def\gls@inlinesep{}%
8212     \def\gls@inlinesubsep{}%
8213     \def\gls@inlinepostchild{}%
8214     }%
8215     {\glspostinline}%

```

No header:

```

8216 \renewcommand*{\glossaryheader}{}%

```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```

8217 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

8218 \renewcommand{\glossentry}[2]{%
8219   \glsinlinedopostchild
8220   \gls@inlinesep
8221   \glsentryitem{##1}%
8222   \glsinlinenameformat{##1}%
8223   \glossentryname{##1}%
8224   }%
8225   \ifglshasdescsuppressed{##1}%
8226   {%
8227     \glsinlineemptydescformat
8228     {%
8229       \glossentrysymbol{##1}%
8230     }%
8231     {%
8232       ##2%
8233     }%
8234   }%
8235   {%
8236     \ifglshasdesc{##1}%
8237     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8238     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8239   }%
8240   \ifglshaschildren{##1}%
8241   {%
8242     \glsresetsubentrycounter
8243     \glsinlineparentchildseparator
8244     \def\gls@inlinesubsep{}%
8245     \def\gls@inlinepostchild{\glsinlinepostchild}%
8246   }%
8247   {}%
8248   \def\gls@inlinesep{\glsinlineseparator}%
8249 }%

```

Sub-entries display description:

```

8250 \renewcommand{\subglossentry}[3]{%
8251   \gls@inlinesubsep%
8252   \glsinlinesubnameformat{##2}%

```



```

8253      \glossentryname{##2}}}%
8254      \glssubentryitem{##2}%
8255      \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8256      \def\gls@inlinesubsep{\glsinlinesubseparator}%
8257  }%

```

Nothing special between groups:

```

8258  \renewcommand*{\glsgroupskip}{}%
8259 }

```

linedopostchild

```

8260 \newcommand*{\glsinlinedopostchild}{%
8261      \gls@inlinepostchild
8262      \def\gls@inlinepostchild{}%
8263 }

```

inlineseparator Separator to use between entries.

```

8264 \newcommand*{\glsinlineseparator}{;\space}

```

inlinesubseparator Separator to use between sub-entries.

```

8265 \newcommand*{\glsinlinesubseparator}{,\space}

```

parentchildseparator Separator to use between parent and children.

```

8266 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

inlinepostchild Hook to use between child and next entry

```

8267 \newcommand*{\glsinlinepostchild}{}

```

\glspostinline Terminator for inline glossary.

```

8268 \newcommand*{\glspostinline}{\glspostdescription\space}

```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```

8269 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

inlinedescformat Formats the entry's description, symbol and location list:

```

8270 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```

8271 \newcommand*{\glsinlineemptydescformat}[2]{}

```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```

8272 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}

```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```

8273 \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.

8274 \ProvidesPackage{glossary-list}[2018/07/23 v4.41 (NLCT)]

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8275 \providecommand{\indexspace}{%
8276   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8277 }
```

`tgrouphaderfmt` Provide a way of adjusting the format of the group headings.

```
8278 \newcommand*{\glslistgrouphaderfmt}[1]{#1}
```

`tnavigationitem` Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of `item` to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```
8279 \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.

```
8280 \newglossarystyle{list}{%
```

Use description environment:

```
8281 \renewenvironment{theglossary}%
8282   {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8283 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8284 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8285 \renewcommand*{\glossentry}[2]{%
8286   \item[\glsentryitem{##1}%
8287     \glstarget{##1}{\glossentryname{##1}}]
8288   \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
8289 \renewcommand*{\subglossentry}[3]{%
8290   \glssubentryitem{##2}%
```

```

8291 \glstarget{##2}{\strut}\space
8292 \glossentrydesc{##2}\glspostdescription\space ##3.}%

Add vertical space between groups:

8293 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8294 }

```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```

8295 \newglossarystyle{listgroup}{%

Base it on the list style:

8296 \setglossarystyle{list}%

Each group has a heading:

8297 \renewcommand*{\glsgroupheading}[1]{%
8298 \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.

```

8299 \newglossarystyle{listhypergroup}{%

Base it on the list style:

8300 \setglossarystyle{list}%

Add navigation links at the start of the environment.

8301 \renewcommand*{\glossaryheader}{%
8302 \glslistnavigationitem{\glslnavigation}}%

Each group has a heading with a hypertarget:

8303 \renewcommand*{\glsgroupheading}[1]{%
8304 \item[\glslistgroupheaderfmt
8305 {\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.

```

8306 \newglossarystyle{altlist}{%

Base it on the list style:

8307 \setglossarystyle{list}%

Main (level 0) entries start a new item in the list with a line break after the entry name:

8308 \renewcommand*{\glossentry}[2]{%
8309 \item[\glssentryitem{##1}%
8310 \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.

```

8311 \mbox{}\par\nobreak\@afterheading
8312 \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries start a new paragraph:

```
8313 \renewcommand{\subglossentry}[3]{%
8314   \par
8315   \glssubentryitem{##2}%
8316   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
8317 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8318 \newglossarystyle{altlistgroup}{%
      Base it on the altlist style:
8319   \setglossarystyle{altlist}%
      Each group has a heading:
8320   \renewcommand*{\glsgroupeheading}[1]{%
8321     \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.

```
8322 \newglossarystyle{altlisthypergroup}{%
      Base it on the altlist style:
8323   \setglossarystyle{altlist}%
      Add navigation links at the start of the environment.
8324   \renewcommand*{\glossaryheader}{%
8325     \glslistnavigationitem{\glslistnavigation}}%
      Each group has a heading with a hypertarget:
8326   \renewcommand*{\glsgroupeheading}[1]{%
8327     \item[\glslistgroupheaderfmt
8328       {\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.

```
8329 \newglossarystyle{listdotted}{%
      Base it on the list style:
8330   \setglossarystyle{list}%
      Each main (level 0) entry starts a new item:
8331   \renewcommand*{\glossentry}[2]{%
8332     \item[\makebox[\glslistdottedwidth][l]{%
8333       \glssentryitem{##1}%
8334       \glstarget{##1}{\glossentryname{##1}}}%
8335     \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```

8336 \renewcommand*{\subglossentry}[3]{%
8337   \item[\makebox[\glslstdottedwidth][l]{%
8338     \glssubentryitem{##2}%
8339     \glstarget{##2}{\glossentryname{##2}}}%
8340   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8341 }
```

`listdottedwidth`

```

8342 \newlength\glslstdottedwidth
8343 \setlength{\glslstdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8344 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
8345 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```

8346 \renewcommand*{\glossentry}[2]{%
8347   \item[\glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
8348 }
```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8349 \ProvidesPackage{glossary-long}[2018/07/23 v4.41 (NLCT)]
```

Requires the package:

```
8350 \RequirePackage{longtable}
```

`\glstdescwidth` 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 `\glstdescwidth` may have already been defined by . The same goes for `\glspagelistwidth`.)

```

8351 \@ifundefined{glstdescwidth}{%
8352   \newlength\glstdescwidth
8353   \setlength{\glstdescwidth}{0.6\hsize}
8354 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column.

```

8355 \@ifundefined{glspagelistwidth}{%
8356   \newlength\glspagelistwidth
8357   \setlength{\glspagelistwidth}{0.1\hsize}
8358 }{}
```

long The long glossary style command which uses the longtable environment:

```
8359 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
8360 \renewenvironment{theglossary}{%
8361     {\begin{longtable}{lp{\glstdescwidth}}}%
8362     {\end{longtable}}}%
```

Do nothing at the start of the environment:

```
8363 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8364 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8365 \renewcommand{\glossentry}[2]{%
8366     \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8367     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8368 }%
```

Sub entries displayed on the following row without the name:

```
8369 \renewcommand{\subglossentry}[3]{%
8370     &
8371     \glssubentryitem{##2}%
8372     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8373     ##3\tabularnewline
8374 }%
```

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>)

```
8375 \ifglsgroupskip
8376     \renewcommand*{\glsgroupskip}{}%
8377 \else
8378     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8379 \fi
8380 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
8381 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
8382 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
8383 \renewenvironment{theglossary}{%
8384     \begin{longtable}{|lp{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8385 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8386 }
```

longheader The longheader style is like the long style but with a header:

```
8387 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8388 \setglossarystyle{long}%
```

Set the table's header:

```
8389 \renewcommand*{\glossaryheader}{%
8390   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8391 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8392 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8393 \setglossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8394 \renewcommand*{\glossaryheader}{%
8395   \hline\bfseries \entryname & \bfseries
8396   \descriptionname\tabularnewline\hline
8397   \endhead
8398   \hline\endfoot}%
8399 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
8400 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
8401 \renewenvironment{theglossary}%
8402   {\begin{longtable}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8403   {\end{longtable}}%
```

No table header:

```
8404 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8405 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8406 \renewcommand{\glossentry}[2]{%
8407   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8408   \glossentrydesc{##1} & ##2\tabularnewline
8409   }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8410 \renewcommand{\subglossentry}[3]{%
8411   &
8412   \glssubentryitem{##2}%
8413   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8414   ##3\tabularnewline
8415   }%
```

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>)

```
8416 \ifglsgroupskip
8417 \renewcommand*{\glsgroupskip}{}%
8418 \else
8419 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8420 \fi
8421 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
8422 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8423 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8424 \renewenvironment{theglossary}%
8425 {\begin{longtable}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8426 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8427 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8428 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
8429 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
8430 \setglossarystyle{long3col}%
    Set the table's header:
8431 \renewcommand*{\glossaryheader}{%
8432 \bfseries\entryname&\bfseries\descriptionname&
8433 \bfseries\pagelistname\tabularnewline\endhead}%
8434 }
```

colheaderborder The long3colheaderborder style is like the above but with a border

```
8435 \newglossarystyle{long3colheaderborder}{%
    Base it on the glostylelong3colborder style:
8436 \setglossarystyle{long3colborder}%
    Set the table's header and add horizontal line at table's foot:
8437 \renewcommand*{\glossaryheader}{%
8438 \hline
8439 \bfseries\entryname&\bfseries\descriptionname&
8440 \bfseries\pagelistname\tabularnewline\hline\endhead
8441 \hline\endfoot}%
8442 }
```


`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8443 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8444 \renewenvironment{theglossary}{%
```

```
8445 {\begin{longtable}{llll}}%
```

```
8446 {\end{longtable}}}%
```

No table header:

```
8447 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8448 \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):

```
8449 \renewcommand{\glossentry}[2]{%
```

```
8450 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
8451 \glossentrydesc{##1} &
```

```
8452 \glossentrysymbol{##1} &
```

```
8453 ##2\tabularnewline
```

```
8454 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8455 \renewcommand{\subglossentry}[3]{%
```

```
8456 &
```

```
8457 \glssubentryitem{##2}%
```

```
8458 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
8459 \glossentrysymbol{##2} & ##3\tabularnewline
```

```
8460 }%
```

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>)

```
8461 \ifglsgnogroupskip
```

```
8462 \renewcommand*{\glsgroupskip}{}%
```

```
8463 \else
```

```
8464 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
```

```
8465 \fi
```

```
8466 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8467 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8468 \setglossarystyle{long4col}%
```

Table has a header:

```
8469 \renewcommand*{\glossaryheader}{%
```

```
8470 \bfseries\entryname&\bfseries\descriptionname&
```

```
8471 \bfseries \symbolname&
```

```

8472 \bfseries\pagelistname\tabularnewline\endhead}%
8473 }

```

long4colborder The long4colborder style is like long4col but with a border.

```

8474 \newglossarystyle{long4colborder}{%
    Base it on the glostylelong4col style:
8475 \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8476 \renewenvironment{theglossary}%
8477 {\begin{longtable}{|l|l|l|l|}}%
8478 {\end{longtable}}%
    Add horizontal lines to the head and foot of the table:
8479 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8480 }

```

colheaderborder The long4colheaderborder style is like the above but with a border.

```

8481 \newglossarystyle{long4colheaderborder}{%
    Base it on the glostylelong4col style:
8482 \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8483 \renewenvironment{theglossary}%
8484 {\begin{longtable}{|l|l|l|l|}}%
8485 {\end{longtable}}%
    Add table header and horizontal line at the table's foot:
8486 \renewcommand*{\glossaryheader}{%
8487 \hline\bfseries\entryname&\bfseries\descriptionname&
8488 \bfseries \symbolname&
8489 \bfseries\pagelistname\tabularnewline\hline\endhead
8490 \hline\endfoot}%
8491 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8492 \newglossarystyle{altlong4col}{%
    Base it on the glostylelong4col style:
8493 \setglossarystyle{long4col}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8494 \renewenvironment{theglossary}%
8495 {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8496 {\end{longtable}}%
8497 }

```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```

8498 \newglossarystyle{altlong4colheader}{%
      Base it on the glostylelong4colheader style:
8499   \setglossarystyle{long4colheader}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8500   \renewenvironment{theglossary}%
8501     {\begin{longtable}{\lp{\glsgdescwidth}\lp{\glspagelistwidth}}}%
8502     {\end{longtable}}}%
8503 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```

8504 \newglossarystyle{altlong4colborder}{%
      Base it on the glostylelong4colborder style:
8505   \setglossarystyle{long4colborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8506   \renewenvironment{theglossary}%
8507     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8508     {\end{longtable}}}%
8509 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```

8510 \newglossarystyle{altlong4colheaderborder}{%
      Base it on the glostylelong4colheaderborder style:
8511   \setglossarystyle{long4colheaderborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8512   \renewenvironment{theglossary}%
8513     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8514     {\end{longtable}}}%
8515 }
```

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>

```

8516 \ProvidesPackage{glossary-longbooktabs}[2018/07/23 v4.41 (NLCT)]
```

Requires booktabs package:

```

8517 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8518 \RequirePackage{glossary-long}
8519 \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.

```
8520 \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.

```
8521 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8522 \setglossarystyle{long}%
```

Add a header with rules.

```
8523 \renewcommand*{\glossaryheader}{%
8524   \toprule \bfseries \entryname & \bfseries
8525   \descriptionname\tabularnewline\midrule\endhead
8526   \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.

```
8527 \ifglsgnogroupskip
8528   \renewcommand*{\glsgroupskip}{}%
8529 \else
8530   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8531 \fi
8532 }
```

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.

```
8533 \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.

```
8534 \glspatchLToutput
```

Use the long3col style as a base.

```
8535 \setglossarystyle{long3col}%
```

Add a header with rules.

```
8536 \renewcommand*{\glossaryheader}{%
8537   \toprule \bfseries \entryname &
8538   \bfseries \descriptionname &
8539   \bfseries \pagelistname
8540   \tabularnewline\midrule\endhead
8541   \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.

```
8542 \ifglsgroupskip
8543 \renewcommand*{\glsgroupskip}{}%
8544 \else
8545 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8546 \fi
8547 }
```

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.

```
8548 \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.

```
8549 \glspatchLToutput
```

Use the long4col style as a base.

```
8550 \setglossarystyle{long4col}{%
```

Add a header with rules.

```
8551 \renewcommand*{\glossaryheader}{%
8552 \toprule \bfseries \entryname &
8553 \bfseries \descriptionname &
8554 \bfseries \symbolname &
8555 \bfseries \pagelistname
8556 \tabularnewline\midrule\endhead
8557 \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.

```
8558 \ifglsgroupskip
8559 \renewcommand*{\glsgroupskip}{}%
8560 \else
8561 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8562 \fi
8563 }
```

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.

```
8564 \newglossarystyle{altlong4col-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.

```
8565 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8566 \setglossarystyle{long4col-booktabs}{%
```

Change the column specifications:

```
8567 \renewenvironment{theglossary}%  
8568   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%  
8569   {\end{longtable}}}%  
8570 }
```

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.

```
8571 \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.

```
8572 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8573 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8574 \renewenvironment{theglossary}%  
8575   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%  
8576   {\end{longtable}}}%  
8577 }
```

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.

```
8578 \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.

```
8579 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8580 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8581 \renewenvironment{theglossary}%  
8582   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%  
8583     >{\raggedright}p{\glspagelistwidth}}}%  
8584   {\end{longtable}}}%  
8585 }
```

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.

```
8586 \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.

```
8587 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8588 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8589 \renewenvironment{theglossary}%
8590   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8591     >{\raggedright}p{\glspagelistwidth}}}%
8592   {\end{longtable}}%
8593 }
```

sLTpenaltycheck

```
8594 \newcommand*{\glslTpenaltycheck}{%
8595   \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi
8596 }
```

enaltygroupskip

```
8597 \newcommand{\glspenaltygroupskip}{%
8598   \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8599 \let\@gls@org@LT@output\LT@output
8600 \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

```
8601 \newcommand*{\glspatchLToutput}{%
8602   \renewcommand*{\LT@output}{%
8603     \ifnum\outputpenalty < -\@Mi
8604       \ifnum\outputpenalty > -\LT@end@pen
8605         \LT@err{floats and marginpars not allowed in a longtable}\@ehc
8606       \else
8607         \setbox\z@\vbox{\unvbox\@cclv}%
8608         \ifdim \ht\LT@lastfoot>\ht\LT@foot
8609           \dimen@pagegoal
8610           \advance\dimen@-\ht\LT@lastfoot
8611           \ifdim\dimen@<\ht\z@
8612             \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
8613             \@makecol
8614             \@outputpage
8615             \setbox\z@\vbox{\box\LT@head\glslTpenaltycheck}%
8616           \fi
8617         \fi
8618         \global\@colroom\@colht
8619         \global\vsizel\@colht
8620         {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8621       \fi
8622     \else
```

```

8623 \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8624 \@makecol
8625 \@outputpage
8626 \global\ysize\@colroom
8627 \copy\LT@head
8628 \glsLTpenaltycheck
8629 \nobreak
8630 \fi
8631 }%
8632 }

```

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.

```
8633 \ProvidesPackage{glossary-longragged}[2018/07/23 v4.41 (NLCT)]
```

Requires the package:

```
8634 \RequirePackage{array}
```

Requires the package:

```
8635 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8636 \@ifundefined{glsdescwidth}{%
8637 \newlength\glsdescwidth
8638 \setlength{\glsdescwidth}{0.6\hsize}
8639 }{}

```

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8640 \@ifundefined{glspagelistwidth}{%
8641 \newlength\glspagelistwidth
8642 \setlength{\glspagelistwidth}{0.1\hsize}
8643 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8644 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8645 \renewenvironment{theglossary}%
8646 {\begin{longtable}{l>\raggedright}p{\glsdescwidth}}%
8647 {\end{longtable}}%

```

Do nothing at the start of the environment:

```
8648 \renewcommand*{\glossaryheader}{}%
```


No heading between groups:

```
8649 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8650 \renewcommand{\glossentry}[2]{%
8651   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8652   \glossentrydesc{##1}\glspostdescription\space ##2%
8653   \tabularnewline
8654 }%
```

Sub entries displayed on the following row without the name:

```
8655 \renewcommand{\subglossentry}[3]{%
8656   &
8657   \glssubentryitem{##2}%
8658   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8659   \glspostdescription\space ##3%
8660   \tabularnewline
8661 }%
```

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>)

```
8662 \ifglsgroupskip
8663 \renewcommand*{\glsgroupskip}{}%
8664 \else
8665 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8666 \fi
8667 }
```

`ongraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8668 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8669 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8670 \renewenvironment{theglossary}{%
8671   \begin{longtable}{|l|>{\raggedright}p{\glsglwidth}|}%
8672   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8673 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8674 }
```

`ongraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8675 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8676 \setglossarystyle{longragged}%
```

Set the table's header:

```
8677 \renewcommand*{\glossaryheader}{%
8678   \bfseries \entryname & \bfseries \descriptionname
```

```

8679 \tabularnewline\endhead}%
8680 }

```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```

8681 \newglossarystyle{longraggedheaderborder}{%
      Base it on the glostylelongraggedborder style:
8682 \setglossarystyle{longraggedborder}%
      Set the table's header and add horizontal line to table's foot:
8683 \renewcommand*{\glossaryheader}{%
8684 \hline\bfseries \entryname & \bfseries \descriptionname
8685 \tabularnewline\hline
8686 \endhead
8687 \hline\endfoot}%
8688 }

```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```

8689 \newglossarystyle{longragged3col}{%
      Use a longtable with 3 columns:
8690 \renewenvironment{theglossary}%
8691 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8692 >{\raggedright}p{\glspagelistwidth}}}%
8693 {\end{longtable}}%

```

No table header:

```

8694 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```

8695 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8696 \renewcommand{\glossentry}[2]{%
8697 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8698 \glossentrydesc{##1} & ##2\tabularnewline
8699 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8700 \renewcommand{\subglossentry}[3]{%
8701 &
8702 \glssubentryitem{##2}%
8703 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8704 ##3\tabularnewline
8705 }%

```

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>)

```

8706 \ifglsnogroupskip
8707 \renewcommand*{\glsgroupskip}{}%

```

```

8708 \else
8709 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8710 \fi
8711 }

```

ragged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```

8712 \newglossarystyle{longragged3colborder}{%
    Base it on the glostylelongragged3col style:
8713 \setglossarystyle{longragged3col}%
    Use a longtable with 3 columns with vertical lines around them:
8714 \renewenvironment{theglossary}%
8715 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%
8716 >{\raggedright}p{\glspagelistwidth}|}%
8717 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8718 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8719 }

```

ragged3colheader The longragged3colheader style is like longragged3col but with a header row:

```

8720 \newglossarystyle{longragged3colheader}{%
    Base it on the glostylelongragged3col style:
8721 \setglossarystyle{longragged3col}%
    Set the table's header:
8722 \renewcommand*{\glossaryheader}{%
8723 \bfseries\entryname&\bfseries\descriptionname&
8724 \bfseries\pagelistname\tabularnewline\endhead}%
8725 }

```

colheaderborder The longragged3colheaderborder style is like the above but with a border

```

8726 \newglossarystyle{longragged3colheaderborder}{%
    Base it on the glostylelongragged3colborder style:
8727 \setglossarystyle{longragged3colborder}%
    Set the table's header and add horizontal line at table's foot:
8728 \renewcommand*{\glossaryheader}{%
8729 \hline
8730 \bfseries\entryname&\bfseries\descriptionname&
8731 \bfseries\pagelistname\tabularnewline\hline\endhead
8732 \hline\endfoot}%
8733 }

```

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.

```

8734 \newglossarystyle{altlongragged4col}{%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8735 \renewenvironment{theglossary}%
8736   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8737     >{\raggedright}p{\glspagelistwidth}}}%
8738   {\end{longtable}}%
```

No table header:

```
8739 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8740 \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):

```
8741 \renewcommand{\glossentry}[2]{%
8742   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8743   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8744   ##2\tabularnewline
8745 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8746 \renewcommand{\subglossentry}[3]{%
8747   &
8748   \glssubentryitem{##2}%
8749   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8750   \glossentrysymbol{##2} & ##3\tabularnewline
8751 }%
```

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>)

```
8752 \ifglsgroupskip
8753   \renewcommand*{\glsgroupskip}{}%
8754 \else
8755   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8756 \fi
8757 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8758 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8759 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8760 \renewenvironment{theglossary}%
8761   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8762     >{\raggedright}p{\glspagelistwidth}}}%
8763   {\end{longtable}}%
```

Table has a header:

```
8764 \renewcommand*{\glossaryheader}{%
8765 \bfseries\entryname&\bfseries\descriptionname&
8766 \bfseries \symbolname&
8767 \bfseries\pagelistname\tabularnewline\endhead}%
8768 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8769 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8770 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8771 \renewenvironment{theglossary}%
8772 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8773 >\raggedright}p{\glspagelistwidth}|}}%
8774 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8775 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8776 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8777 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8778 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8779 \renewenvironment{theglossary}%
8780 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8781 >\raggedright}p{\glspagelistwidth}|}}%
8782 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8783 \renewcommand*{\glossaryheader}{%
8784 \hline\bfseries\entryname&\bfseries\descriptionname&
8785 \bfseries \symbolname&
8786 \bfseries\pagelistname\tabularnewline\hline\endhead
8787 \hline\endfoot}%
8788 }
```

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.

```
8789 \ProvidesPackage{glossary-mcols}[2018/07/23 v4.41 (NLCT)]
```

Required packages:

```
8790 \RequirePackage{multicol}
8791 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8792 \providecommand{\indexspace}{%
8793   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8794 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8795 \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.)

```
8796 \newglossarystyle{mcolindex}{%
8797   \setglossarystyle{index}%
8798   \renewenvironment{theglossary}%
8799     {%
8800       \begin{multicols}{\glsmcols}
8801       \setlength{\parindent}{0pt}%
8802       \setlength{\parskip}{0pt plus 0.3pt}%
8803       \let\item\glstreeitem
8804       \let\subitem\glstreesubitem
8805       \let\subsubitem\glstreesubsubitem
8806     }%
8807     {\end{multicols}}}%
8808 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8809 \newglossarystyle{mcolindexgroup}{%
8810   \setglossarystyle{mcolindex}%
8811   \renewcommand*{\glsgroupheading}[1]{%
8812     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8813 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8814 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8815   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8816   \renewcommand*{\glossaryheader}{%
8817     \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.

```
8818 \renewcommand*{\glsgroupheading}[1]{%
8819 \item\glstreegroupheaderfmt
8820 {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8821 \indexspace}%
8822 }
```

colindexspannav Similar to **colindexhypergroup**, but puts the navigation line in the optional argument of **multicols**.

```
8823 \newglossarystyle{colindexspannav}{%
8824 \setglossarystyle{index}%
8825 \renewenvironment{theglossary}%
8826 {%
8827 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8828 \setlength{\parindent}{0pt}%
8829 \setlength{\parskip}{0pt plus 0.3pt}%
8830 \let\item\glstreeitem}%
8831 {\end{multicols}}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8832 \renewcommand*{\glsgroupheading}[1]{%
8833 \item\glstreegroupheaderfmt
8834 {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
8835 \indexspace}%
8836 }
```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8837 \newglossarystyle{mcoltree}{%
8838 \setglossarystyle{tree}%
8839 \renewenvironment{theglossary}%
8840 {%
8841 \begin{multicols}{\glsmcols}
8842 \setlength{\parindent}{0pt}%
8843 \setlength{\parskip}{0pt plus 0.3pt}%
8844 }%
8845 {\end{multicols}}%
8846 }
```

mcoltreegroup Like the **mcoltree** style but the glossary groups have headings.

```
8847 \newglossarystyle{mcoltreegroup}{%
Base it on the glostylemcoltree style:
8848 \setglossarystyle{mcoltree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```
8849 \renewcommand{\glsgroupheading}[1]{\par
8850 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8851 }
```

ltreehypergroup The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8852 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the `glostylemcoltree` style:

```
8853 \setglossarystyle{mcoltree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8854 \renewcommand*{\glossaryheader}{%
8855 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8856 \renewcommand*{\glsgroupheading}[1]{%
8857 \par\noindent
8858 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8859 \indexspace}%
8860 }
```

mcoltreespannav Similar to the `mcoltreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8861 \newglossarystyle{mcoltreespannav}{%
8862 \setglossarystyle{tree}%
8863 \renewenvironment{theglossary}%
8864 {%
8865 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8866 \setlength{\parindent}{0pt}%
8867 \setlength{\parskip}{0pt plus 0.3pt}%
8868 }%
8869 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8870 \renewcommand*{\glsgroupheading}[1]{%
8871 \par\noindent
8872 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8873 \indexspace}%
8874 }
```

mcoltreenoname Multi-column index style. Same as the `treenoname`, but puts the glossary in multiple columns.

```
8875 \newglossarystyle{mcoltreenoname}{%
8876 \setglossarystyle{treenoname}%
8877 \renewenvironment{theglossary}%
8878 {%
```



```

8879      \begin{multicols}{\glsmcols}
8880      \setlength{\parindent}{0pt}%
8881      \setlength{\parskip}{0pt plus 0.3pt}%
8882    }%
8883    {\end{multicols}}%
8884  }

```

treenonamegroup Like the `mcoltreenoname` style but the glossary groups have headings.

```

8885 \newglossarystyle{mcoltreenonamegroup}{%
      Base it on the glostylemcoltreenoname style:
8886   \setglossarystyle{mcoltreenoname}%
      Give each group a heading:
8887   \renewcommand{\glsgroupheading}[1]{\par
8888     \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8889 }

```

onamehypergroup The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

8890 \newglossarystyle{mcoltreenonamehypergroup}{%
      Base it on the glostylemcoltreenoname style:
8891   \setglossarystyle{mcoltreenoname}%
      Put navigation links to the groups at the start of the theglossary environment:
8892   \renewcommand*{\glossaryheader}{%
8893     \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8894   \renewcommand*{\glsgroupheading}[1]{%
8895     \par\noindent
8896     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8897     \indexspace}%
8898 }

```

eenonamespannav Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8899 \newglossarystyle{mcoltreenonamespannav}{%
8900   \setglossarystyle{treenoname}%
8901   \renewenvironment{theglossary}%
8902   {%
8903     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8904     \setlength{\parindent}{0pt}%
8905     \setlength{\parskip}{0pt plus 0.3pt}%
8906   }%
8907   {\end{multicols}}%
      Each group has a heading (in bold with a target) followed by a vertical gap):
8908   \renewcommand*{\glsgroupheading}[1]{%
8909     \par\noindent

```

```

8910 \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8911 \indexspace}%
8912 }

```

mcolalttree Multi-column index style. Same as the alttree, but puts the glossary in multiple columns.

```

8913 \newglossarystyle{mcolalttree}{%
8914 \setglossarystyle{alttree}%
8915 \renewenvironment{theglossary}%
8916 {%
8917 \begin{multicols}{\glsmcols}
8918 \def\@gls@prevlevel{-1}%
8919 \mbox{}}\par
8920 }%
8921 {\par\end{multicols}}}%
8922 }

```

colalttreegroup Like the mcolalttree style but the glossary groups have headings.

```

8923 \newglossarystyle{colalttreegroup}{%
      Base it on the glostylemcolalttree style:
8924 \setglossarystyle{mcolalttree}%
      Give each group a heading.
8925 \renewcommand{\glsgroupheading}[1]{\par
8926 \def\@gls@prevlevel{-1}%
8927 \hangindent0pt\relax
8928 \parindent0pt\relax
8929 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8930 }

```

treehypergroup The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```

8931 \newglossarystyle{mcolalttreehypergroup}{%
      Base it on the glostylemcolalttree style:
8932 \setglossarystyle{mcolalttree}%
      Put the navigation links in the header
8933 \renewcommand*{\glossaryheader}{%
8934 \par
8935 \def\@gls@prevlevel{-1}%
8936 \hangindent0pt\relax
8937 \parindent0pt\relax
8938 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8939 \renewcommand*{\glsgroupheading}[1]{%
8940 \par
8941 \def\@gls@prevlevel{-1}%
8942 \hangindent0pt\relax

```

```

8943 \parindent0pt\relax
8944 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8945 \indexspace}%
8946 }

```

`lalttreespannav` Similar to the `mcolalttreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8947 \newglossarystyle{mcolalttreespannav}{%
8948 \setglossarystyle{alttree}%
8949 \renewenvironment{theglossary}%
8950 {%
8951 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glshnavigation}]
8952 \def\@glsp@prevlevel{-1}%
8953 \mbox{}\par
8954 }%
8955 {\par\end{multicols}}}%

```

Put a hypertarget at the start of each group

```

8956 \renewcommand*{\glsgroupheading}[1]{%
8957 \par
8958 \def\@glsp@prevlevel{-1}%
8959 \hangindent0pt\relax
8960 \parindent0pt\relax
8961 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8962 \indexspace}%
8963 }

```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the `supertabular` environment.

```

8964 \ProvidesPackage{glossary-super}[2018/07/23 v4.41 (NLCT)]

```

Requires the package:

```

8965 \RequirePackage{supertabular}

```

`\glsdscwidth` This is a length that governs the width of the description column. This may already have been defined if `has` has been loaded.

```

8966 \@ifundefined{glsdscwidth}{%
8967 \newlength{glsdscwidth}
8968 \setlength{glsdscwidth}{0.6\hsize}
8969 }{}

```

`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.

```

8970 \@ifundefined{glspagelistwidth}{%
8971 \newlength{glspagelistwidth}
8972 \setlength{glspagelistwidth}{0.1\hsize}

```

8973 }{}

super The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

8974 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

8975 \renewenvironment{theglossary}{%

8976 {\tablehead{}\tabletail{}}%

8977 \begin{supertabular}{lp{\glsdescwidth}}}%

8978 {\end{supertabular}}}%

Do nothing at the start of the table:

8979 \renewcommand*{\glossaryheader}{}%

No group headings:

8980 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

8981 \renewcommand{\glossentry}[2]{%

8982 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

8983 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline

8984 }%

Sub entries put in a row (no name, description and page list in second column):

8985 \renewcommand{\subglossentry}[3]{%

8986 &

8987 \glssubentryitem{##2}%

8988 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space

8989 ##3\tabularnewline

8990 }%

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>)

8991 \ifglsnogroupskip

8992 \renewcommand*{\glsgroupskip}{}%

8993 \else

8994 \renewcommand*{\glsgroupskip}{& \tabularnewline}%

8995 \fi

8996 }

superborder The superborder style is like the above, but with horizontal and vertical lines:

8997 \newglossarystyle{superborder}{%

Base it on the glostylesuper style:

8998 \setglossarystyle{super}%

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

8999 \renewenvironment{theglossary}{%

9000 {\tablehead{\hline}\tabletail{\hline}%

```

9001      \begin{supertabular}{|l|p{\glsdescwidth}|}%
9002      {\end{supertabular}}%
9003 }

```

superheader The superheader style is like the super style, but with a header:

```

9004 \newglossarystyle{superheader}{%
      Base it on the glostylesuper style:
9005   \setglossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and no tail:
9006 \renewenvironment{theglossary}%
9007   {\tablehead{\bfseries \entryname &
9008     \bfseries\descriptionname\tabularnewline}%
9009     \tabletail{}}%
9010   \begin{supertabular}{lp{\glsdescwidth}}%
9011   {\end{supertabular}}%
9012 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```

9013 \newglossarystyle{superheaderborder}{%
      Base it on the glostylesuper style:
9014   \setglossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and horizontal
      lines above and below the table:
9015   \renewenvironment{theglossary}%
9016     {\tablehead{\hline\bfseries \entryname &
9017       \bfseries \descriptionname\tabularnewline\hline}%
9018       \tabletail{\hline}}
9019     \begin{supertabular}{|l|p{\glsdescwidth}|}%
9020     {\end{supertabular}}%
9021 }

```

super3col The super3col style is like the super style, but with 3 columns:

```

9022 \newglossarystyle{super3col}{%
      Put the glossary in a supertabular environment with three columns and no head or tail:
9023   \renewenvironment{theglossary}%
9024     {\tablehead{}\tabletail{}}%
9025     \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
9026     {\end{supertabular}}%
      Do nothing at the start of the table:
9027   \renewcommand*{\glossaryheader}{}%
      No group headings:
9028   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9029 \renewcommand{\glossentry}[2]{%
9030   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9031   \glossentrydesc{##1} & ##2\tabularnewline
9032 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
9033 \renewcommand{\subglossentry}[3]{%
9034   &
9035   \glssubentryitem{##2}%
9036   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9037   ##3\tabularnewline
9038 }%
```

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>)

```
9039 \ifglsgroupskip
9040   \renewcommand*{\glsgroupskip}{}%
9041 \else
9042   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9043 \fi
9044 }
```

super3colborder The super3colborder style is like the super3col style, but with a border:

```
9045 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
9046 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
9047 \renewenvironment{theglossary}%
9048   {\tablehead{\hline}\tabletail{\hline}%
9049   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth|}}%
9050   {\end{supertabular}}%
9051 }
```

super3colheader The super3colheader style is like the super3col style but with a header row:

```
9052 \newglossarystyle{super3colheader}{%
```

Base it on the glostylesuper3col style:

```
9053 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
9054 \renewenvironment{theglossary}%
9055   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9056   \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9057   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
9058   {\end{supertabular}}%
9059 }
```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
9060 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
9061 \setglossarystyle{super3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9062 \renewenvironment{theglossary}{%
9063   {\tablehead{\hline
9064     \bfseries\entryname&\bfseries\descriptionname&
9065     \bfseries\pagelistname\tabularnewline\hline}%
9066   \tabletail{\hline}%
9067   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
9068   {\end{supertabular}}}%
9069 }
```

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.

```
9070 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9071 \renewenvironment{theglossary}{%
9072   {\tablehead{}\tabletail{}}%
9073   \begin{supertabular}{|l|l|l|l|}%
9074   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9075 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9076 \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:

```
9077 \renewcommand{\glossentry}[2]{%
9078   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9079   \glossentrydesc{##1} &
9080   \glossentrysymbol{##1} & ##2\tabularnewline
9081   }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9082 \renewcommand{\subglossentry}[3]{%
9083   &
9084   \glssubentryitem{##2}%
9085   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9086   \glossentrysymbol{##2} & ##3\tabularnewline
9087   }%
```

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>)

```

9088 \ifglsgroupskip
9089 \renewcommand*{\glsgroupskip}{}%
9090 \else
9091 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9092 \fi
9093 }

```

super4colheader The super4colheader style is like the super4col but with a header row.

```

9094 \newglossarystyle{super4colheader}{%
    Base it on the glostylesuper4col style:
9095 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns, a header and no tail:
9096 \renewenvironment{theglossary}%
9097 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9098 \bfseries\symbolname &
9099 \bfseries\pagelistname\tabularnewline}%
9100 \tabletail{}}%
9101 \begin{supertabular}{1111}}%
9102 {\end{supertabular}}%
9103 }

```

super4colborder The super4colborder style is like the super4col but with a border.

```

9104 \newglossarystyle{super4colborder}{%
    Base it on the glostylesuper4col style:
9105 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns and a horizontal line in the
    head and tail:
9106 \renewenvironment{theglossary}%
9107 {\tablehead{\hline}\tabletail{\hline}%
9108 \begin{supertabular}{11111111}}%
9109 {\end{supertabular}}%
9110 }

```

colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```

9111 \newglossarystyle{super4colheaderborder}{%
    Base it on the glostylesuper4col style:
9112 \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:
9113 \renewenvironment{theglossary}%
9114 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
9115 \bfseries\symbolname &

```



```

9116      \bfseries\pagelistname\tabularnewline\hline}%
9117      \tabletail{\hline}%
9118      \begin{supertabular}{|l|l|l|l|}%
9119      {\end{supertabular}}%
9120 }

```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```

9121 \newglossarystyle{altsuper4col}{%
      Base it on the glostylesuper4col style:
9122   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and no head or tail:
9123   \renewenvironment{theglossary}%
9124     {\tablehead{}\tabletail{}}%
9125     \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
9126     {\end{supertabular}}%
9127 }

```

super4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```

9128 \newglossarystyle{altsuper4colheader}{%
      Base it on the glostylesuper4colheader style:
9129   \setglossarystyle{super4colheader}%
      Put the glossary in a supertabular environment with four columns, a header and no tail:
9130   \renewenvironment{theglossary}%
9131     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9132       \bfseries\symbolname &
9133       \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9134     \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
9135     {\end{supertabular}}%
9136 }

```

super4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```

9137 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
9138   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a horizontal line in the
      head and tail:
9139   \renewenvironment{theglossary}%
9140     {\tablehead{\hline}\tabletail{\hline}%
9141     \begin{supertabular}%
9142       {|l|p{\glsgdescwidth}|l|p{\glspagelistwidth}|}%
9143     {\end{supertabular}}%
9144 }

```

colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```

9145 \newglossarystyle{altsuper4colheaderborder}{%

```

Base it on the `glostylessuper4colheaderborder` style:

```
9146 \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:

```
9147 \renewenvironment{theglossary}%
9148   {\tablehead{\hline
9149     \bfseries\entryname &
9150     \bfseries\descriptionname &
9151     \bfseries\symbolname &
9152     \bfseries\pagelistname\tabularnewline\hline}%
9153   \tabletail{\hline}%
9154   \begin{supertabular}%
9155     {lllp{\glsdescwidth}llp{\glspagelistwidth}}}%
9156   {\end{supertabular}}%
9157 }
```

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.

```
9158 \ProvidesPackage{glossary-superragged}[2018/07/23 v4.41 (NLCT)]
```

Requires the package:

```
9159 \RequirePackage{array}
```

Requires the package:

```
9160 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
9161 \@ifundefined{glsdescwidth}{%
9162   \newlength\glsdescwidth
9163   \setlength{\glsdescwidth}{0.6\hsize}
9164 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
9165 \@ifundefined{glspagelistwidth}{%
9166   \newlength\glspagelistwidth
9167   \setlength{\glspagelistwidth}{0.1\hsize}
9168 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
9169 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9170 \renewenvironment{theglossary}%
9171   {\tablehead{}\tabletail{}}%
9172   \begin{supertabular}{|l>{\raggedright}p{\glsgdescwidth}}}%
9173   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9174 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9175 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9176 \renewcommand{\glossentry}[2]{%
9177   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9178   \glossentrydesc{##1}\glspostdescription\space ##2%
9179   \tabularnewline
9180 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9181 \renewcommand{\subglossentry}[3]{%
9182   &
9183   \glssubentryitem{##2}%
9184   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9185   ##3%
9186   \tabularnewline
9187 }%
```

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>)

```
9188 \ifglsgroupskip
9189   \renewcommand*{\glsgroupskip}{}%
9190 \else
9191   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9192 \fi
9193 }
```

`\perraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
9194 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylesuperragged` style:

```
9195 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9196 \renewenvironment{theglossary}%
9197   {\tablehead{\hline}\tabletail{\hline}%
9198   \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}%
9199   {\end{supertabular}}%
9200 }
```

`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
9201 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
9202 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
9203 \renewenvironment{theglossary}%
```

```
9204 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
```

```
9205 \tabularnewline}%
```

```
9206 \tabletail{}}%
```

```
9207 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}%
```

```
9208 {\end{supertabular}}%
```

```
9209 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
9210 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
9211 \setglossarystyle{superragged}%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
9212 \renewenvironment{theglossary}%
```

```
9213 {\tablehead{\hline\bfseries \entryname &
```

```
9214 \bfseries \descriptionname\tabularnewline\hline}%
```

```
9215 \tabletail{\hline}
```

```
9216 \begin{supertabular}{ll>{\raggedright}p{\glsgdescwidth}}%
```

```
9217 {\end{supertabular}}%
```

```
9218 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
9219 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
9220 \renewenvironment{theglossary}%
```

```
9221 {\tablehead{}\tabletail{}}%
```

```
9222 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
```

```
9223 >{\raggedright}p{\glspagelistwidth}}%
```

```
9224 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9225 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9226 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9227 \renewcommand{\glossentry}[2]{%
```

```
9228 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9229 \glossentrydesc{##1} &
```

```

9230    ##2\tabularnewline
9231  }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9232  \renewcommand{\subglossentry}[3]{%
9233    &
9234    \glssubentryitem{##2}%
9235    \glstarget{##2}{\strut}\glossentrydesc{##2} &
9236    ##3\tabularnewline
9237  }%

```

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>)

```

9238  \ifglsnogroupskip
9239  \renewcommand*{\glsgroupskip}{}%
9240  \else
9241  \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9242  \fi
9243 }

```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```

9244 \newglossarystyle{superragged3colborder}{%

```

Base it on the glostylesuperragged3col style:

```

9245  \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9246  \renewenvironment{theglossary}%
9247  {\tablehead{\hline}\tabletail{\hline}%
9248    \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}||%
9249    >{\raggedright}p{\glspagerlistwidth}||}%
9250    {\end{supertabular}}%
9251 }

```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```

9252 \newglossarystyle{superragged3colheader}{%

```

Base it on the glostylesuperragged3col style:

```

9253  \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9254  \renewenvironment{theglossary}%
9255  {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9256    \bfseries\pagerlistname\tabularnewline}\tabletail{}}%
9257  \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9258  >{\raggedright}p{\glspagerlistwidth}}}%
9259  {\end{supertabular}}%
9260 }

```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9261 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9262 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9263 \renewenvironment{theglossary}{%
```

```
9264 {\tablehead{\hline
```

```
9265 \bfseries\entryname&\bfseries\descriptionname&
```

```
9266 \bfseries\pagelistname\tabularnewline\hline}%
```

```
9267 \tabletail{\hline}%
```

```
9268 \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
```

```
9269 >{\raggedright}p{\glspagelistwidth}|}}%
```

```
9270 {\end{supertabular}}%
```

```
9271 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9272 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9273 \renewenvironment{theglossary}{%
```

```
9274 {\tablehead{}\tabletail{}%
```

```
9275 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
```

```
9276 >{\raggedright}p{\glspagelistwidth}}}%
```

```
9277 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9278 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9279 \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:

```
9280 \renewcommand{\glossentry}[2]{%
```

```
9281 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9282 \glossentrydesc{##1} &
```

```
9283 \glossentrysymbol{##1} & ##2\tabularnewline
```

```
9284 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9285 \renewcommand{\subglossentry}[3]{%
```

```
9286 &
```

```
9287 \glssubentryitem{##2}%
```

```
9288 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
9289 \glossentrysymbol{##2} & ##3\tabularnewline
```

```
9290 }%
```

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>)

```

9291 \ifglsgroupskip
9292 \renewcommand*{\glsgroupskip}{}%
9293 \else
9294 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9295 \fi
9296 }

```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```

9297 \newglossarystyle{altsuperragged4colheader}{%

```

Base it on the glostylealtsuperragged4col style:

```

9298 \setglossarystyle{altsuperragged4col}%

```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

9299 \renewenvironment{theglossary}%
9300 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9301 \bfseries\symbolname &
9302 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9303 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9304 >{\raggedright}p{\glspagelistwidth}}}%
9305 {\end{supertabular}}%
9306 }

```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```

9307 \newglossarystyle{altsuperragged4colborder}{%

```

Base it on the glostylealtsuperragged4col style:

```

9308 \setglossarystyle{altsuper4col}%

```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

9309 \renewenvironment{theglossary}%
9310 {\tablehead{\hline}\tabletail{\hline}%
9311 \begin{supertabular}%
9312 {l|>{\raggedright}p{\glsgdescwidth}l|}%
9313 >{\raggedright}p{\glspagelistwidth}l}}%
9314 {\end{supertabular}}%
9315 }

```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```

9316 \newglossarystyle{altsuperragged4colheaderborder}{%

```

Base it on the glostylealtsuperragged4col style:

```

9317 \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:

```

9318 \renewenvironment{theglossary}%
9319   {\tablehead{\hline
9320     \bfseries\entryname &
9321     \bfseries\descriptionname &
9322     \bfseries\symbolname &
9323     \bfseries\pagelistname\tabularnewline\hline}%
9324   \tabletail{\hline}%
9325   \begin{supertabular}%
9326     {||>{\raggedright}p{\glsdescwidth}||}%
9327     >{\raggedright}p{\glspagelistwidth}||}%
9328   {\end{supertabular}}%
9329 }

```

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.

```

9330 \ProvidesPackage{glossary-tree}[2018/07/23 v4.41 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9331 \providecommand{\indexspace}{%
9332   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9333 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command was previously also used to format the group headings.

```

9334 \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.

```

9335 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9336 \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.

```

9337 \ifdef\@idxitem
9338 {\newcommand{\glstreeitem}{\@idxitem}}
9339 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```


`\glstreesubitem` Level 1 item used in index style.

```
9340 \ifdef\subitem
9341 {\let\glstreesubitem\subitem}
9342 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

`\glstreesubsubitem` Level 1 item used in index style.

```
9343 \ifdef\subsubitem
9344 {\let\glstreesubsubitem\subsubitem}
9345 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `alttree` style).

```
9346 \newcommand{\glstreepredesc}{\space}
```

`\glstreechildpredesc` Allow the user to adjust the space before the description for sub-entries (except for the `treename` and `alttree` style).

```
9347 \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.

```
9348 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9349 \renewenvironment{theglossary}%
9350 {\setlength{\parindent}{0pt}}%
9351 {\setlength{\parskip}{0pt plus 0.3pt}}%
9352 \let\item\glstreeitem
9353 \let\subitem\glstreesubitem
9354 \let\subsubitem\glstreesubsubitem
9355 }%
```

```
9356 {\par}}%
```

Do nothing at the start of the environment:

```
9357 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9358 \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.

```
9359 \renewcommand*{\glossentry}[2]{%
9360 \item\glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
9361 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9362 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9363 }%
```

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.

```

9364 \renewcommand{\subglossentry}[3]{%
9365   \ifcase##1\relax
9366     % level 0
9367     \item
9368   \or
9369     % level 1
9370     \subitem
9371     \glssubentryitem{##2}%
9372   \else
9373     % all other levels
9374     \subsubitem
9375   \fi
9376   \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}%
9377   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9378   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9379 }%
```

Vertical gap between groups is the same as that used by indices:

```

9380 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The `indexgroup` style is like the `index` style but has headings.

```

9381 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```

9382 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9383 \renewcommand*{\glsgroupheading}[1]{%
9384   \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9385   \indexspace
9386 }%
9387 }
```

indexhypergroup The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

9388 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```

9389 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```

9390 \renewcommand*{\glossaryheader}{%
9391   \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.

```

9392 \renewcommand*{\glsgroupheading}[1]{%
9393   \item\glstreegroupheaderfmt
```

```

9394      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
9395      \indexspace}%
9396 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9397 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9398   \renewenvironment{theglossary}%
9399     {\setlength{\parindent}{0pt}%
9400     \setlength{\parskip}{0pt plus 0.3pt}}%
9401   {}%

```

Do nothing at the start of the theglossary environment:

```

9402   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9403   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

9404   \renewcommand{\glossentry}[2]{%
9405     \hangindent0pt\relax
9406     \parindent0pt\relax
9407     \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9408     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9409     \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9410   }%

```

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.

```

9411   \renewcommand{\subglossentry}[3]{%
9412     \hangindent##1\glstreeindent\relax
9413     \parindent##1\glstreeindent\relax
9414     \ifnum##1=1\relax
9415       \glssubentryitem{##2}%
9416       \fi
9417       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9418       \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9419       \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9420   }%

```

Vertical gap between groups is the same as that used by indices:

```

9421   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

9422 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

9423   \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```
9424 \renewcommand{\glsgrouphheading}[1]{\par
9425 \noindent\glstreegroupheaderfmt{\glsggetgrouptitle{##1}}\par
9426 \indexspace}%
9427 }
```

treehypergroup The **treehypergroup** style is like the **treegroup** style, but has a set of links to the groups at the start of the glossary.

```
9428 \newglossarystyle{treehypergroup}{%
```

Base it on the **glostyletree** style:

```
9429 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the **theglossary** environment:

```
9430 \renewcommand*{\glossaryheader}{%
9431 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9432 \renewcommand*{\glsgrouphheading}[1]{%
9433 \par\noindent
9434 \glstreegroupheaderfmt
9435 {\glsnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
9436 \indexspace}%
9437 }
```

\glstreeindent Length governing left indent for each level of the tree style.

```
9438 \newlength\glstreeindent
9439 \setlength{\glstreeindent}{10pt}
```

treenoname The **treenoname** glossary style is like the **tree** style, but doesn't print the name or symbol for sub-levels.

```
9440 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9441 \renewenvironment{theglossary}%
9442 {\setlength{\parindent}{0pt}%
9443 \setlength{\parskip}{0pt plus 0.3pt}}%
9444 {}%
```

No header:

```
9445 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9446 \renewcommand*{\glsgrouphheading}[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.

```
9447 \renewcommand{\glossentry}[2]{%
9448 \hangindent0pt\relax
9449 \parindent0pt\relax
9450 \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```

9451 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9452 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9453 }%

```

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.

```

9454 \renewcommand{\subglossentry}[3]{%
9455 \hangindent##1\glstreeindent\relax
9456 \parindent##1\glstreeindent\relax
9457 \ifnum##1=1\relax
9458 \glssubentryitem{##2}%
9459 \fi
9460 \glstarget{##2}{\strut}%
9461 \glossentrydesc{##2}\glspostdescription\space##3\par
9462 }%

```

Vertical gap between groups is the same as that used by indices:

```

9463 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9464 }

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```

9465 \newglossarystyle{treenonamegroup}{%

```

Base it on the `glostyletreenoname` style:

```

9466 \setglossarystyle{treenoname}%

```

Give each group a heading:

```

9467 \renewcommand{\glsgroupheading}[1]{\par
9468 \noindent\glstreegroupheaderfmt
9469 {\glsgrouptitle{##1}}\par\indexspace}%
9470 }

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9471 \newglossarystyle{treenonamehypergroup}{%

```

Base it on the `glostyletreenoname` style:

```

9472 \setglossarystyle{treenoname}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

9473 \renewcommand*{\glossaryheader}{%
9474 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9475 \renewcommand*{\glsgroupheading}[1]{%
9476 \par\noindent
9477 \glstreegroupheaderfmt
9478 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9479 \indexspace}%
9480 }

```

`esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```

9481 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9482   \dimen@=0pt\relax
9483   \gls@tmplen=0pt\relax
9484   \forallglossaries[#1]{\@gls@type}%
9485   {%
9486     \forallglsentries[\@gls@type]{\@glo@label}%
9487     {%
9488       \ifglsahasparent{\@glo@label}%
9489       }%
9490       {%
9491         \settowidth{\dimen@}%
9492           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
9493         \ifdim\dimen@>\gls@tmplen
9494           \gls@tmplen=\dimen@
9495           \letcs{\@glswidestname}{glo@\glsdetoklabel{\@glo@label}@name}%
9496         \fi
9497       }%
9498     }%
9499   }%
9500 }
```

`\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.

```

9501 \newcommand*{\glssetwidest}[2][0]{%
9502   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9503     #2}%
9504 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```

9505 \newcommand*{\@glswidestname}{}
```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9506 \newcommand*{\glstreenamebox}[2]{%
9507   \makebox[#1][l]{#2}%
9508 }
```

`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`.

```

9509 \newglossarystyle{alttree}{%
  Redefine theglossary environment.
9510   \renewenvironment{theglossary}%
9511     {\def\@gls@prevlevel{-1}%
9512      \mbox{}\par}%
9513     {\par}%
  Set the header and group headers to nothing.
9514   \renewcommand*{\glossaryheader}{}%
9515   \renewcommand*{\glsgroupheading}[1]{}%
}
```

Redefine the way that the level 0 entries are displayed.

```
9516 \renewcommand{\glossentry}[2]{%
9517   \ifnum\@gls@prevlevel=0\relax
9518   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9519     \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9520   \fi
```

Set the hangindent and paragraph indent.

```
9521   \hangindent\glstreeindent
9522   \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9523   \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
9524     \glssentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9525   \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9526   \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9527   \def\@gls@prevlevel{0}%
9528 }%
```

Redefine the way sub-entries are displayed.

```
9529 \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.

```
9530   \ifnum##1=1\relax
9531     \glssubentryitem{##2}%
9532   \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust \glstreeindent accordingly.

```
9533   \ifnum\@gls@prevlevel=##1\relax
9534   \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in \gls@tmplen

```
9535     \@ifundefined{@glswidestname\romannumeral##1}{%
9536       \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%
9537       \settowidth{\gls@tmplen}{\glstreenamefmt{%
9538         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9539   \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9540      \setlength\glstreeindent\gls@tmplen
9541      \addtolength\glstreeindent\parindent
9542      \parindent\glstreeindent
9543      \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`.

```
9544      \ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9545      \settowidth{\glstreeindent}{\glstreenamfmt{%
9546      \@glswidestname\space}}}{%
9547      \settowidth{\glstreeindent}{\glstreenamfmt{%
9548      \csname @glswidestname\romannumeral\@gls@prevlevel
9549      \endcsname\space}}}{%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9550      \addtolength\parindent{-\glstreeindent}%
9551      \setlength\glstreeindent\parindent
9552      \fi
9553      \fi
```

Set the hanging indentation.

```
9554      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9555      \makebox[0pt][r]{\glstreenambox{\gls@tmplen}{%
9556      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9557      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9558      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9559      \def\@gls@prevlevel{##1}%
9560      }%
```

Vertical gap between groups is the same as that used by indices:

```
9561      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9562 }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
9563 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
9564 \setglossarystyle{almtree}%
```

Give each group a heading.

```
9565 \renewcommand{\glsgroupheading}[1]{\par
9566 \def\@gls@prevlevel{-1}%
9567 \hangindent0pt\relax
```



```

9568     \parindent0pt\relax
9569     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
9570     \par\indexspace}%
9571 }

```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9572 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9573   \setglossarystyle{alttree}%
    Put the navigation links in the header
9574   \renewcommand*{\glossaryheader}{%
9575     \par
9576     \def\@gls@prevlevel{-1}%
9577     \hangindent0pt\relax
9578     \parindent0pt\relax
9579     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9580   \renewcommand*{\glsgroupheading}[1]{%
9581     \par
9582     \def\@gls@prevlevel{-1}%
9583     \hangindent0pt\relax
9584     \parindent0pt\relax
9585     \glstreegroupheaderfmt
9586     {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9587     \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.

```
9588 \NeedsTeXFormat{LaTeX2e}
9589 \ProvidesPackage{glossaries-compatible-207}[2018/07/23 v4.41 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9590 \ifglsxindy
9591   \renewcommand*\GlsAddXdyAttribute[1]{%
9592     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9593     \expandafter\toks@\expandafter{\@xdylocref}%
9594     \edef\@xdylocref{\the\toks@ ^^J}%
9595     (markup-locref
9596     :open \string"\string~n\string\setentrycounter
9597       {\noexpand\glscounter}%
9598       \expandafter\string\csname#1\endcsname
9599       \expandafter\@gobble\string\{\string" ^^J
9600     :close \string"\expandafter\@gobble\string\}\string" ^^J
9601     :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
9602 \fi
```

sAddXdyCounters

```
9603 \renewcommand*\GlsAddXdyCounters[1]{%
9604   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9605     in compatibility mode.}%
9606 }
```

Add predefined attributes

```
9607 \GlsAddXdyAttribute{glsnumberformat}
9608 \GlsAddXdyAttribute{textrm}
9609 \GlsAddXdyAttribute{textsf}
9610 \GlsAddXdyAttribute{texttt}
9611 \GlsAddXdyAttribute{textbf}
9612 \GlsAddXdyAttribute{textmd}
9613 \GlsAddXdyAttribute{textit}
9614 \GlsAddXdyAttribute{textup}
9615 \GlsAddXdyAttribute{textsl}
```

```

9616 \GlsAddXdyAttribute{textsc}
9617 \GlsAddXdyAttribute{emph}
9618 \GlsAddXdyAttribute{glshypernumber}
9619 \GlsAddXdyAttribute{hyperrm}
9620 \GlsAddXdyAttribute{hypersf}
9621 \GlsAddXdyAttribute{hypertt}
9622 \GlsAddXdyAttribute{hyperbf}
9623 \GlsAddXdyAttribute{hypermd}
9624 \GlsAddXdyAttribute{hyperit}
9625 \GlsAddXdyAttribute{hyperup}
9626 \GlsAddXdyAttribute{hypersl}
9627 \GlsAddXdyAttribute{hypersc}
9628 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9629 \ifglxindy
9630 \renewcommand*{\GlsAddXdyLocation}[2]{%
9631   \edef\@xdyuserlocationdefs{%
9632     \@xdyuserlocationdefs ^^J%
9633     (define-location-class \string"#1\string"^^J\space\space
9634     \space(#2))
9635   }%
9636   \edef\@xdyuserlocationnames{%
9637     \@xdyuserlocationnames^^J\space\space\space
9638     \string"#1\string"}%
9639 }
9640 \fi

```

\@do@wrglossary

```

9641 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9642 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9643 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9644 \def\@glo@range{}%
9645 \expandafter\if\@glo@prefix(\relax
9646 \def\@glo@range{:open-range}%
9647 \else
9648 \expandafter\if\@glo@prefix)\relax
9649 \def\@glo@range{:close-range}%
9650 \fi
9651 \fi

  Get the location and escape any special characters
9652 \protected@edef\@glslocref{\theglsentrycounter}%
9653 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9654 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9655 (indexentry :tkey (\csname glo@#1@index\endcsname)
9656 :locoref \string"\@glslocoref\string" %
9657 :attr \string"\@glo@suffix\string" \@glo@range
9658 )
9659 }%
9660 \else

```

Convert the format information into the format required for makeindex

```

9661 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9662 \glossary[\csname glo@#1@type\endcsname]{%
9663 \string\glossaryentry{\csname glo@#1@index\endcsname
9664 \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9665 \fi
9666 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9667 \def\@set@glo@numformat#1#2#3{%
9668 \expandafter\@glo@check@mkidxrangechar#3\@nil
9669 \protected@edef#1{%
9670 \@glo@prefix setentrycounter[] {#2}%
9671 \expandafter\string\csname\@glo@suffix\endcsname
9672 }%
9673 \@gls@checkmkidxchars#1%
9674 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9675 \ifglxsindy
9676 \def\writeist{%
9677 \openout\glswrite=\istfilename
9678 \write\glswrite{;; xindy style file created by the glossaries
9679 package in compatible-2.07 mode}%
9680 \write\glswrite{;; for document '\jobname' on
9681 \the\year-\the\month-\the\day}%
9682 \write\glswrite{^^J; required styles^^J}
9683 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9684 \ifx\@xdystyle\@empty
9685 \else
9686 \protected@write\glswrite{{(require
9687 \string"\@xdystyle.xdy\string")}}%
9688 \fi
9689 }%
9690 \write\glswrite{^^J%
9691 ; list of allowed attributes (number formats)^^J}%
9692 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9693 \write\glswrite{^^J; user defined alphabets^^J}%
9694 \write\glswrite{\@xdyuseralphabets}%
9695 \write\glswrite{^^J; location class definitions^^J}%
9696 \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9697     \string"roman-numbers-lowercase\string" :sep \string"}}%
9698 \@onelevel@sanitize\@gls@roman
9699 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9700     :sep \string"}}%
9701 \@onelevel@sanitize\@tmp
9702 \ifx\@tmp\@gls@roman
9703     \write\glswrite{(define-location-class
9704         \string"roman-page-numbers\string"^^J\space\space\space
9705         (\string"roman-numbers-lowercase\string")
9706         :min-range-length \@glsminrange)}}%
9707 \else
9708     \write\glswrite{(define-location-class
9709         \string"roman-page-numbers\string"^^J\space\space\space
9710         (:sep "\@gls@roman")
9711         :min-range-length \@glsminrange)}}%
9712 \fi
9713 \write\glswrite{(define-location-class
9714     \string"Roman-page-numbers\string"^^J\space\space\space
9715     (\string"roman-numbers-uppercase\string")
9716     :min-range-length \@glsminrange)}}%
9717 \write\glswrite{(define-location-class
9718     \string"arabic-page-numbers\string"^^J\space\space\space
9719     (\string"arabic-numbers\string")
9720     :min-range-length \@glsminrange)}}%
9721 \write\glswrite{(define-location-class
9722     \string"alpha-page-numbers\string"^^J\space\space\space
9723     (\string"alpha\string")
9724     :min-range-length \@glsminrange)}}%
9725 \write\glswrite{(define-location-class
9726     \string"Alpha-page-numbers\string"^^J\space\space\space
9727     (\string"ALPHA\string")
9728     :min-range-length \@glsminrange)}}%
9729 \write\glswrite{(define-location-class
9730     \string"Appendix-page-numbers\string"^^J\space\space\space
9731     (\string"ALPHA\string"
9732     :sep \string"\@glsAlphacompositor\string"
9733     \string"arabic-numbers\string")
9734     :min-range-length \@glsminrange)}}%
9735 \write\glswrite{(define-location-class
9736     \string"arabic-section-numbers\string"^^J\space\space\space
9737     (\string"arabic-numbers\string"
9738     :sep \string"\glscompositor\string"
9739     \string"arabic-numbers\string")
9740     :min-range-length \@glsminrange)}}%
9741 \write\glswrite{^^J; user defined location classes}%
9742 \write\glswrite{\@xdyuserlocationdefs}%
9743 \write\glswrite{^^J; define cross-reference class^^J}%
9744 \write\glswrite{(define-crossref-class \string"see\string"
9745     :unverified )}%

```

```

9746 \write\glswrite{(markup-crossref-list
9747 :class \string"see\string"^^J\space\space\space
9748 :open \string"\string\glseeformat\string"
9749 :close \string"{}\string")}%
9750 \write\glswrite{^^J; define the order of the location classes}%
9751 \write\glswrite{(define-location-class-order
9752 (\@xdylocationclassorder))}%
9753 \write\glswrite{^^J; define the glossary markup^^J}%
9754 \write\glswrite{(markup-index^^J\space\space\space
9755 :open \string"\string
9756 \glossarysection[\string\glossarytoctitle]{\string
9757 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9758 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9759 \space\space:close \string"\expandafter\@gobble
9760 \string%\string~n\string
9761 \end{theglossary}\string\glossarypostamble
9762 \string~n\string" ^^J\space\space\space
9763 :tree)}}%
9764 \write\glswrite{(markup-letter-group-list
9765 :sep \string"\string\glsgroupskip\string~n\string")}%
9766 \write\glswrite{(markup-indexentry
9767 :open \string"\string\relax \string\glresetentrylist
9768 \string~n\string")}%
9769 \write\glswrite{(markup-locclass-list :open
9770 \string"\glsoopenbrace\string\glossaryentrynumbers
9771 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9772 :sep \string", \string"
9773 :close \string"\glsclosebrace\glsclosebrace\string")}%
9774 \write\glswrite{(markup-locref-list
9775 :sep \string"\string\delimN\space\string")}%
9776 \write\glswrite{(markup-range
9777 :sep \string"\string\delimR\space\string")}%
9778 \@onelevel@sanitize\gls@suffixF
9779 \@onelevel@sanitize\gls@suffixFF
9780 \ifx\gls@suffixF\@empty
9781 \else
9782 \write\glswrite{(markup-range
9783 :close "\gls@suffixF" :length 1 :ignore-end)}%
9784 \fi
9785 \ifx\gls@suffixFF\@empty
9786 \else
9787 \write\glswrite{(markup-range
9788 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9789 \fi
9790 \write\glswrite{^^J; define format to use for locations^^J}%
9791 \write\glswrite{\@xdylocref}%
9792 \write\glswrite{^^J; define letter group list format^^J}%
9793 \write\glswrite{(markup-letter-group-list
9794 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9795 \write\glswrite{^^J; letter group headings^^J}%
9796 \write\glswrite{(markup-letter-group
9797   :open-head \string"\string\glsgroupheading
9798   \glsoopenbrace\string"^^J\space\space\space
9799   :close-head \string"\glsclosebrace\string")}%
9800 \write\glswrite{^^J; additional letter groups^^J}%
9801 \write\glswrite{\@xdylettergroups}%
9802 \write\glswrite{^^J; additional sort rules^^J}
9803 \write\glswrite{\@xdysortrules}%
9804 \noist}
9805 \else
9806 \edef\@gls@actualchar{\string?}
9807 \edef\@gls@encapchar{\string|}
9808 \edef\@gls@levelchar{\string!}
9809 \edef\@gls@quotechar{\string"}
9810 \def\writeist{\relax
9811   \openout\glswrite=\istfilename
9812   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9813     created by the glossaries package}
9814   \write\glswrite{\expandafter\@gobble\string\% for document
9815     '\jobname' on \the\year-\the\month-\the\day}
9816   \write\glswrite{actual '\@gls@actualchar'}
9817   \write\glswrite{encap '\@gls@encapchar'}
9818   \write\glswrite{level '\@gls@levelchar'}
9819   \write\glswrite{quote '\@gls@quotechar'}
9820   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9821   \write\glswrite{preamble \string"\string\glossarysection[\string
9822     \glossarytoctitle]{\string\glossarytitle}\string
9823     \glossarypreamble\string\n\string\begin{theglossary}\string
9824     \glossaryheader\string\n\string"}
9825   \write\glswrite{postamble \string"\string%\string\n\string
9826     \end{theglossary}\string\glossarypostamble\string\n
9827     \string"}
9828   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9829     \string"}
9830   \write\glswrite{item_0 \string"\string%\string\n\string"}
9831   \write\glswrite{item_1 \string"\string%\string\n\string"}
9832   \write\glswrite{item_2 \string"\string%\string\n\string"}
9833   \write\glswrite{item_01 \string"\string%\string\n\string"}
9834   \write\glswrite{item_x1
9835     \string"\string\relax \string\glsresetentrylist\string\n
9836     \string"}
9837   \write\glswrite{item_12 \string"\string%\string\n\string"}
9838   \write\glswrite{item_x2
9839     \string"\string\relax \string\glsresetentrylist\string\n
9840     \string"}
9841   \write\glswrite{delim_0 \string"\string\{\string
9842     \glossaryentrynumbers\string\{\string\relax \string"}
9843   \write\glswrite{delim_1 \string"\string\{\string

```

```

9844      \glossaryentrynumbers\string\{\string\relax \string"}
9845      \write\glswrite{delim_2 \string"\string\{\string
9846      \glossaryentrynumbers\string\{\string\relax \string"}
9847      \write\glswrite{delim_t \string"\string\}\string\}\string"}
9848      \write\glswrite{delim_n \string"\string\delimN \string"}
9849      \write\glswrite{delim_r \string"\string\delimR \string"}
9850      \write\glswrite{headings_flag 1}
9851      \write\glswrite{heading_prefix
9852      \string"\string\glsgroupheading\string\{\string"}
9853      \write\glswrite{heading_suffix
9854      \string"\string\}\string\relax
9855      \string\glsgroupresetentrylist \string"}
9856      \write\glswrite{symhead_positive \string"glssymbols\string"}
9857      \write\glswrite{numhead_positive \string"glslnumbers\string"}
9858      \write\glswrite{page_compositor \string"glscpositor\string"}
9859      \@gls@escbsdq\gls@suffixF
9860      \@gls@escbsdq\gls@suffixFF
9861      \ifx\gls@suffixF\@empty
9862      \else
9863      \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
9864      \fi
9865      \ifx\gls@suffixFF\@empty
9866      \else
9867      \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
9868      \fi
9869      \noist
9870  }
9871 \fi

```

\noist

```

9872 \renewcommand*{\noist}{\let\writeist\relax}

```

4.2 glossaries-compatible-307

```

9873 \NeedsTeXFormat{LaTeX2e}
9874 \ProvidesPackage{glossaries-compatible-307}[2018/07/23 v4.41 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9875 \newcommand{\compatglossarystyle}[2]{%
9876   \ifcsundef{@glscompstyle@#1}%
9877   {%
9878     \csdef{@glscompstyle@#1}{#2}%
9879   }%
9880   {%
9881     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9882   }%
9883 }

```


Backward compatible inline style.

```

9884 \compatglossarystyle{inline}{%
9885   \renewcommand{\glossaryentryfield}[5]{%
9886     \glsinlinedopostchild
9887     \gls@inlinesep
9888     \def\glo@desc{##3}%
9889     \def\@no@post@desc{\nopostdesc}%
9890     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
9891     \ifx\glo@desc\@no@post@desc
9892       \glsinlineemptydescformat{##4}{##5}%
9893     \else
9894       \ifstrepty{##3}%
9895         {\glsinlineemptydescformat{##4}{##5}}%
9896         {\glsinlinedescformat{##3}{##4}{##5}}%
9897     \fi
9898     \ifglshaschildren{##1}%
9899     {%
9900       \glsresetsubentrycounter
9901       \glsinlineparentchildseparator
9902       \def\gls@inlinesubsep{}%
9903       \def\gls@inlinepostchild{\glsinlinepostchild}%
9904     }%
9905   }%
9906   \def\gls@inlinesep{\glsinlineseparator}%
9907 }%
```

Sub-entries display description:

```

9908 \renewcommand{\glossarysubentryfield}[6]{%
9909   \gls@inlinesubsep%
9910   \glsinlinesubnameformat{##2}{##3}%
9911   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9912   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9913 }%
9914 }
```

Backward compatible list style.

```

9915 \compatglossarystyle{list}{%
9916   \renewcommand*{\glossaryentryfield}[5]{%
9917     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
9918     ##3\glspostdescription\space ##5}%
9919 }
```

Sub-entries continue on the same line:

```

9919 \renewcommand*{\glossarysubentryfield}[6]{%
9920   \glssubentryitem{##2}%
9921   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9922 }
```

Backward compatible listgroup style.

```

9923 \compatglossarystyle{listgroup}{%
9924   \csuse{@glscompstyle@list}%
9925 }%
```

Backward compatible listhypergroup style.

```
9926 \compatglossarystyle{listhypergroup}{%
9927   \csuse{@glscompstyle@list}%
9928 }%
```

Backward compatible altlist style.

```
9929 \compatglossarystyle{altlist}{%
9930   \renewcommand*{\glossaryentryfield}[5]{%
9931     \item[\glstentryitem{##1}\glstarget{##1}{##2}]%
9932       \mbox{}\par\nobreak\@afterheading
9933       ##3\glspostdescription\space ##5}%
9934   \renewcommand{\glossarysubentryfield}[6]{%
9935     \par
9936     \glssubentryitem{##2}%
9937     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9938 }%
```

Backward compatible altlistgroup style.

```
9939 \compatglossarystyle{altlistgroup}{%
9940   \csuse{@glscompstyle@altlist}%
9941 }%
```

Backward compatible altlisthypergroup style.

```
9942 \compatglossarystyle{altlisthypergroup}{%
9943   \csuse{@glscompstyle@altlist}%
9944 }%
```

Backward compatible listdotted style.

```
9945 \compatglossarystyle{listdotted}{%
9946   \renewcommand*{\glossaryentryfield}[5]{%
9947     \item[]\makebox[\glslistdottedwidth][l]{%
9948       \glstentryitem{##1}\glstarget{##1}{##2}%
9949       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9950   \renewcommand*{\glossarysubentryfield}[6]{%
9951     \item[]\makebox[\glslistdottedwidth][l]{%
9952       \glssubentryitem{##2}%
9953       \glstarget{##2}{##3}%
9954       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9955 }%
```

Backward compatible sublistdotted style.

```
9956 \compatglossarystyle{sublistdotted}{%
9957   \csuse{@glscompstyle@listdotted}%
9958   \renewcommand*{\glossaryentryfield}[5]{%
9959     \item[\glstentryitem{##1}\glstarget{##1}{##2}]}%
9960 }%
```

Backward compatible long style.

```
9961 \compatglossarystyle{long}{%
9962   \renewcommand*{\glossaryentryfield}[5]{%
9963     \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9964   \renewcommand*{\glossarysubentryfield}[6]{%
9965     \glssubentryitem{##2}\glstarget{##2}{##3} & ##4\glspostdescription\space ##5\\}%
9966 }%
```

```

9965      &
9966      \glssubentryitem{##2}%
9967      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9968 }%

```

Backward compatible longborder style.

```

9969 \compatglossarystyle{longborder}{%
9970 \csuse{@glscmpstyle@long}%
9971 }%

```

Backward compatible longheader style.

```

9972 \compatglossarystyle{longheader}{%
9973 \csuse{@glscmpstyle@long}%
9974 }%

```

Backward compatible longheaderborder style.

```

9975 \compatglossarystyle{longheaderborder}{%
9976 \csuse{@glscmpstyle@long}%
9977 }%

```

Backward compatible long3col style.

```

9978 \compatglossarystyle{long3col}{%
9979 \renewcommand*{\glossaryentryfield}[5]{%
9980 \glstarget{##1}{\strut}##4\glstarget{##1}{##2} & ##3 & ##5\\}%
9981 \renewcommand*{\glossarysubentryfield}[6]{%
9982 &
9983 \glssubentryitem{##2}%
9984 \glstarget{##2}{\strut}##4 & ##6\\}%
9985 }%

```

Backward compatible long3colborder style.

```

9986 \compatglossarystyle{long3colborder}{%
9987 \csuse{@glscmpstyle@long3col}%
9988 }%

```

Backward compatible long3colheader style.

```

9989 \compatglossarystyle{long3colheader}{%
9990 \csuse{@glscmpstyle@long3col}%
9991 }%

```

Backward compatible long3colheaderborder style.

```

9992 \compatglossarystyle{long3colheaderborder}{%
9993 \csuse{@glscmpstyle@long3col}%
9994 }%

```

Backward compatible long4col style.

```

9995 \compatglossarystyle{long4col}{%
9996 \renewcommand*{\glossaryentryfield}[5]{%
9997 \glstarget{##1}{\strut}##4\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9998 \renewcommand*{\glossarysubentryfield}[6]{%
9999 &
10000 \glssubentryitem{##2}%

```

10001 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10002 }%

Backward compatible long4colheader style.

10003 \compatglossarystyle{long4colheader}{%
10004 \csuse{@glscompstyle@long4col}%
10005 }%

Backward compatible long4colborder style.

10006 \compatglossarystyle{long4colborder}{%
10007 \csuse{@glscompstyle@long4col}%
10008 }%

Backward compatible long4colheaderborder style.

10009 \compatglossarystyle{long4colheaderborder}{%
10010 \csuse{@glscompstyle@long4col}%
10011 }%

Backward compatible altlong4col style.

10012 \compatglossarystyle{altlong4col}{%
10013 \csuse{@glscompstyle@long4col}%
10014 }%

Backward compatible altlong4colheader style.

10015 \compatglossarystyle{altlong4colheader}{%
10016 \csuse{@glscompstyle@long4col}%
10017 }%

Backward compatible altlong4colborder style.

10018 \compatglossarystyle{altlong4colborder}{%
10019 \csuse{@glscompstyle@long4col}%
10020 }%

Backward compatible altlong4colheaderborder style.

10021 \compatglossarystyle{altlong4colheaderborder}{%
10022 \csuse{@glscompstyle@long4col}%
10023 }%

Backward compatible long style.

10024 \compatglossarystyle{longragged}{%
10025 \renewcommand*{\glossaryentryfield}[5]{%
10026 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10027 \tabularnewline}%
10028 \renewcommand*{\glossarysubentryfield}[6]{%
10029 &
10030 \glssubentryitem{##2}%
10031 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10032 \tabularnewline}%
10033 }%

Backward compatible longraggedborder style.

10034 \compatglossarystyle{longraggedborder}{%
10035 \csuse{@glscompstyle@longragged}%
10036 }%

Backward compatible longraggedheader style.

```
10037 \compatglossarystyle{longraggedheader}{%  
10038 \csuse{@glscmpstyle@longragged}%  
10039 }%
```

Backward compatible longraggedheaderborder style.

```
10040 \compatglossarystyle{longraggedheaderborder}{%  
10041 \csuse{@glscmpstyle@longragged}%  
10042 }%
```

Backward compatible longragged3col style.

```
10043 \compatglossarystyle{longragged3col}{%  
10044 \renewcommand*{\glossaryentryfield}[5]{%  
10045 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
10046 \renewcommand*{\glossarysubentryfield}[6]{%  
10047 &  
10048 \glssubentryitem{##2}%  
10049 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
10050 }%
```

Backward compatible longragged3colborder style.

```
10051 \compatglossarystyle{longragged3colborder}{%  
10052 \csuse{@glscmpstyle@longragged3col}%  
10053 }%
```

Backward compatible longragged3colheader style.

```
10054 \compatglossarystyle{longragged3colheader}{%  
10055 \csuse{@glscmpstyle@longragged3col}%  
10056 }%
```

Backward compatible longragged3colheaderborder style.

```
10057 \compatglossarystyle{longragged3colheaderborder}{%  
10058 \csuse{@glscmpstyle@longragged3col}%  
10059 }%
```

Backward compatible altlongragged4col style.

```
10060 \compatglossarystyle{altlongragged4col}{%  
10061 \renewcommand*{\glossaryentryfield}[5]{%  
10062 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
10063 \renewcommand*{\glossarysubentryfield}[6]{%  
10064 &  
10065 \glssubentryitem{##2}%  
10066 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
10067 }%
```

Backward compatible altlongragged4colheader style.

```
10068 \compatglossarystyle{altlongragged4colheader}{%  
10069 \csuse{@glscmpstyle@altlong4col}%  
10070 }%
```

Backward compatible altlongragged4colborder style.

```
10071 \compatglossarystyle{altlongragged4colborder}{%
```

```
10072 \csuse{@glscompstyle@altlong4col}%
10073 }%
```

Backward compatible altlongragged4colheaderborder style.

```
10074 \compatglossarystyle{altlongragged4colheaderborder}{%
10075 \csuse{@glscompstyle@altlong4col}%
10076 }%
```

Backward compatible index style.

```
10077 \compatglossarystyle{index}{%
10078 \renewcommand*{\glossaryentryfield}[5]{%
10079 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10080 \ifx\relax##4\relax
10081 \else
10082 \space{##4}%
10083 \fi
10084 \space ##3\glspostdescription \space ##5}%
10085 \renewcommand*{\glossarysubentryfield}[6]{%
10086 \ifcase##1\relax
10087 % level 0
10088 \item
10089 \or
10090 % level 1
10091 \subitem
10092 \glssubentryitem{##2}%
10093 \else
10094 % all other levels
10095 \subsubitem
10096 \fi
10097 \textbf{\glstarget{##2}{##3}}%
10098 \ifx\relax##5\relax
10099 \else
10100 \space{##5}%
10101 \fi
10102 \space##4\glspostdescription\space ##6}%
10103 }%
```

Backward compatible indexgroup style.

```
10104 \compatglossarystyle{indexgroup}{%
10105 \csuse{@glscompstyle@index}%
10106 }%
```

Backward compatible indexhypergroup style.

```
10107 \compatglossarystyle{indexhypergroup}{%
10108 \csuse{@glscompstyle@index}%
10109 }%
```

Backward compatible tree style.

```
10110 \compatglossarystyle{tree}{%
10111 \renewcommand{\glossaryentryfield}[5]{%
10112 \hangindent0pt\relax
```

```

10113 \parindent0pt\relax
10114 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10115 \ifx\relax##4\relax
10116 \else
10117 \space(##4)%
10118 \fi
10119 \space ##3\glspostdescription \space ##5\par}%
10120 \renewcommand{\glossarysubentryfield}[6]{%
10121 \hangindent##1\glstreeindent\relax
10122 \parindent##1\glstreeindent\relax
10123 \ifnum##1=1\relax
10124 \glssubentryitem{##2}%
10125 \fi
10126 \textbf{\glstarget{##2}{##3}}%
10127 \ifx\relax##5\relax
10128 \else
10129 \space(##5)%
10130 \fi
10131 \space##4\glspostdescription\space ##6\par}%
10132 }%

```

Backward compatible treegroup style.

```

10133 \compatglossarystyle{treegroup}{%
10134 \csuse{@glscmpstyle@tree}%
10135 }%

```

Backward compatible treehypergroup style.

```

10136 \compatglossarystyle{treehypergroup}{%
10137 \csuse{@glscmpstyle@tree}%
10138 }%

```

Backward compatible treenoname style.

```

10139 \compatglossarystyle{treenoname}{%
10140 \renewcommand{\glossaryentryfield}[5]{%
10141 \hangindent0pt\relax
10142 \parindent0pt\relax
10143 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10144 \ifx\relax##4\relax
10145 \else
10146 \space(##4)%
10147 \fi
10148 \space ##3\glspostdescription \space ##5\par}%
10149 \renewcommand{\glossarysubentryfield}[6]{%
10150 \hangindent##1\glstreeindent\relax
10151 \parindent##1\glstreeindent\relax
10152 \ifnum##1=1\relax
10153 \glssubentryitem{##2}%
10154 \fi
10155 \glstarget{##2}{\strut}%
10156 ##4\glspostdescription\space ##6\par}%
10157 }%

```

Backward compatible treenonamegroup style.

```
10158 \compatglossarystyle{treenonamegroup}{%
10159   \csuse{@glscompstyle@treenoname}%
10160 }%
```

Backward compatible treenonamehypergroup style.

```
10161 \compatglossarystyle{treenonamehypergroup}{%
10162   \csuse{@glscompstyle@treenoname}%
10163 }%
```

Backward compatible alttree style.

```
10164 \compatglossarystyle{alttree}{%
10165   \renewcommand{\glossaryentryfield}[5]{%
10166     \ifnum\@gls@prevlevel=0\relax
10167     \else
10168       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10169       \hangindent\glstreeindent
10170       \parindent\glstreeindent
10171     \fi
10172     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10173       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10174     \ifx\relax##4\relax
10175     \else
10176       (##4)\space
10177     \fi
10178     ##3\glspostdescription \space ##5\par
10179     \def\@gls@prevlevel{0}%
10180   }%
10181   \renewcommand{\glossarysubentryfield}[6]{%
10182     \ifnum##1=1\relax
10183       \glssubentryitem{##2}%
10184     \fi
10185     \ifnum\@gls@prevlevel=##1\relax
10186     \else
10187       \@ifundefined{@glswidestname\romannumeral##1}{%
10188         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10189         \settowidth{\gls@tmplen}{\textbf{%
10190           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10191       \ifnum\@gls@prevlevel<##1\relax
10192         \setlength\glstreeindent\gls@tmplen
10193         \addtolength\glstreeindent\parindent
10194         \parindent\glstreeindent
10195       \else
10196         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10197         \settowidth{\glstreeindent}{\textbf{%
10198           \@glswidestname\space}}{%
10199         \settowidth{\glstreeindent}{\textbf{%
10200           \csname @glswidestname\romannumeral\@gls@prevlevel
10201             \endcsname\space}}}%
10202         \addtolength\parindent{-\glstreeindent}%

```



```

10203         \setlength\glstreeindent\parindent
10204     \fi
10205 \fi
10206 \hangindent\glstreeindent
10207 \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10208     \textbf{\glstarget{##2}{##3}}}%
10209 \ifx##5\relax\relax
10210 \else
10211     (##5)\space
10212 \fi
10213 ##4\glspostdescription\space ##6\par
10214 \def\@gls@prevlevel{##1}%
10215 }%
10216 }%

```

Backward compatible alttreegroup style.

```

10217 \compatglossarystyle{alttreegroup}{%
10218 \csuse{@glscompstyle@almtree}%
10219 }%

```

Backward compatible almtreehypergroup style.

```

10220 \compatglossarystyle{almtreehypergroup}{%
10221 \csuse{@glscompstyle@almtree}%
10222 }%

```

Backward compatible mcolindex style.

```

10223 \compatglossarystyle{mcolindex}{%
10224 \csuse{@glscompstyle@index}%
10225 }%

```

Backward compatible mcolindexgroup style.

```

10226 \compatglossarystyle{mcolindexgroup}{%
10227 \csuse{@glscompstyle@index}%
10228 }%

```

Backward compatible mcolindexhypergroup style.

```

10229 \compatglossarystyle{mcolindexhypergroup}{%
10230 \csuse{@glscompstyle@index}%
10231 }%

```

Backward compatible mcoltree style.

```

10232 \compatglossarystyle{mcoltree}{%
10233 \csuse{@glscompstyle@tree}%
10234 }%

```

Backward compatible mcoltreegroup style.

```

10235 \compatglossarystyle{mcolindextreegroup}{%
10236 \csuse{@glscompstyle@tree}%
10237 }%

```

Backward compatible mcoltreehypergroup style.

```

10238 \compatglossarystyle{mcolindextreehypergroup}{%

```

```
10239 \csuse{@glscompstyle@tree}%
10240 }%
```

Backward compatible mcoltreenoname style.

```
10241 \compatglossarystyle{mcoltreenoname}{%
10242 \csuse{@glscompstyle@tree}%
10243 }%
```

Backward compatible mcoltreenonamegroup style.

```
10244 \compatglossarystyle{mcoltreenonamegroup}{%
10245 \csuse{@glscompstyle@tree}%
10246 }%
```

Backward compatible mcoltreenonamehypergroup style.

```
10247 \compatglossarystyle{mcoltreenonamehypergroup}{%
10248 \csuse{@glscompstyle@tree}%
10249 }%
```

Backward compatible mcolalttree style.

```
10250 \compatglossarystyle{mcolalttree}{%
10251 \csuse{@glscompstyle@alttree}%
10252 }%
```

Backward compatible mcolalttreegroup style.

```
10253 \compatglossarystyle{mcolalttreegroup}{%
10254 \csuse{@glscompstyle@alttree}%
10255 }%
```

Backward compatible mcolalttreehypergroup style.

```
10256 \compatglossarystyle{mcolalttreehypergroup}{%
10257 \csuse{@glscompstyle@alttree}%
10258 }%
```

Backward compatible superragged style.

```
10259 \compatglossarystyle{superragged}{%
10260 \renewcommand*{\glossaryentryfield}[5]{%
10261 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10262 \tabularnewline}%
10263 \renewcommand*{\glossarysubentryfield}[6]{%
10264 &
10265 \glssubentryitem{##2}%
10266 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10267 \tabularnewline}%
10268 }%
```

Backward compatible superraggedborder style.

```
10269 \compatglossarystyle{superraggedborder}{%
10270 \csuse{@glscompstyle@superragged}%
10271 }%
```

Backward compatible superraggedheader style.

```
10272 \compatglossarystyle{superraggedheader}{%
10273 \csuse{@glscompstyle@superragged}%
10274 }%
```

Backward compatible superraggedheaderborder style.

```
10275 \compatglossarystyle{superraggedheaderborder}{%
10276   \csuse{@glscompstyle@superragged}%
10277 }%
```

Backward compatible superragged3col style.

```
10278 \compatglossarystyle{superragged3col}{%
10279   \renewcommand*{\glossaryentryfield}[5]{%
10280     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10281   \renewcommand*{\glossarysubentryfield}[6]{%
10282     &
10283     \glssubentryitem{##2}%
10284     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10285 }%
```

Backward compatible superragged3colborder style.

```
10286 \compatglossarystyle{superragged3colborder}{%
10287   \csuse{@glscompstyle@superragged3col}%
10288 }%
```

Backward compatible superragged3colheader style.

```
10289 \compatglossarystyle{superragged3colheader}{%
10290   \csuse{@glscompstyle@superragged3col}%
10291 }%
```

Backward compatible superragged3colheaderborder style.

```
10292 \compatglossarystyle{superragged3colheaderborder}{%
10293   \csuse{@glscompstyle@superragged3col}%
10294 }%
```

Backward compatible altsuperragged4col style.

```
10295 \compatglossarystyle{altsuperragged4col}{%
10296   \renewcommand*{\glossaryentryfield}[5]{%
10297     \glstentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10298   \renewcommand*{\glossarysubentryfield}[6]{%
10299     &
10300     \glssubentryitem{##2}%
10301     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10302 }%
```

Backward compatible altsuperragged4colheader style.

```
10303 \compatglossarystyle{altsuperragged4colheader}{%
10304   \csuse{@glscompstyle@altsuperragged4col}%
10305 }%
```

Backward compatible altsuperragged4colborder style.

```
10306 \compatglossarystyle{altsuperragged4colborder}{%
10307   \csuse{@glscompstyle@altsuperragged4col}%
10308 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10309 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10310 \csuse{@glscompstyle@altsuperragged4col}%
10311 }%
```

Backward compatible super style.

```
10312 \compatglossarystyle{super}{%
10313   \renewcommand*{\glossaryentryfield}[5]{%
10314     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10315   \renewcommand*{\glossarysubentryfield}[6]{%
10316     &
10317     \glssubentryitem{##2}%
10318     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10319 }%
```

Backward compatible superborder style.

```
10320 \compatglossarystyle{superborder}{%
10321   \csuse{@glscompstyle@super}%
10322 }%
```

Backward compatible superheader style.

```
10323 \compatglossarystyle{superheader}{%
10324   \csuse{@glscompstyle@super}%
10325 }%
```

Backward compatible superheaderborder style.

```
10326 \compatglossarystyle{superheaderborder}{%
10327   \csuse{@glscompstyle@super}%
10328 }%
```

Backward compatible super3col style.

```
10329 \compatglossarystyle{super3col}{%
10330   \renewcommand*{\glossaryentryfield}[5]{%
10331     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10332   \renewcommand*{\glossarysubentryfield}[6]{%
10333     &
10334     \glssubentryitem{##2}%
10335     \glstarget{##2}{\strut}##4 & ##6\\}%
10336 }%
```

Backward compatible super3colborder style.

```
10337 \compatglossarystyle{super3colborder}{%
10338   \csuse{@glscompstyle@super3col}%
10339 }%
```

Backward compatible super3colheader style.

```
10340 \compatglossarystyle{super3colheader}{%
10341   \csuse{@glscompstyle@super3col}%
10342 }%
```

Backward compatible super3colheaderborder style.

```
10343 \compatglossarystyle{super3colheaderborder}{%
10344   \csuse{@glscompstyle@super3col}%
10345 }%
```

Backward compatible super4col style.

```
10346 \compatglossarystyle{super4col}{%
10347   \renewcommand*{\glossaryentryfield}[5]{%
10348     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10349   \renewcommand*{\glossarysubentryfield}[6]{%
10350     &
10351     \glssubentryitem{##2}%
10352     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10353 }%
```

Backward compatible super4colheader style.

```
10354 \compatglossarystyle{super4colheader}{%
10355   \csuse{@glscompstyle@super4col}%
10356 }%
```

Backward compatible super4colborder style.

```
10357 \compatglossarystyle{super4colborder}{%
10358   \csuse{@glscompstyle@super4col}%
10359 }%
```

Backward compatible super4colheaderborder style.

```
10360 \compatglossarystyle{super4colheaderborder}{%
10361   \csuse{@glscompstyle@super4col}%
10362 }%
```

Backward compatible altsuper4col style.

```
10363 \compatglossarystyle{altsuper4col}{%
10364   \csuse{@glscompstyle@super4col}%
10365 }%
```

Backward compatible altsuper4colheader style.

```
10366 \compatglossarystyle{altsuper4colheader}{%
10367   \csuse{@glscompstyle@super4col}%
10368 }%
```

Backward compatible altsuper4colborder style.

```
10369 \compatglossarystyle{altsuper4colborder}{%
10370   \csuse{@glscompstyle@super4col}%
10371 }%
```

Backward compatible altsuper4colheaderborder style.

```
10372 \compatglossarystyle{altsuper4colheaderborder}{%
10373   \csuse{@glscompstyle@super4col}%
10374 }%
```

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.

```
10375 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10376 \ProvidesPackage{glossaries-accsupp}[2018/07/23 v4.41 (NLCT)]
```

```
10377 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10378 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10379 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10380 \ifpackageloaded{glossaries-extra}
```

```
10381 {%
```

If the accsupp option was used, \@glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10382 \ifx\@glsxtr@doaccsupp\empty
```

```
10383 \GlossariesWarning{The ‘glossaries-accsupp’
```

```
10384 package has been loaded\MessageBreak
```

```
10385 after the ‘glossaries-extra’ package. This\MessageBreak
```

```
10386 can cause a failure to integrate both packages. \MessageBreak
```

```
10387 Either use the ‘accsupp’ option when you load\MessageBreak
```

```
10388 ‘glossaries-extra’ or load ‘glossaries-accsupp’\MessageBreak
```

```
10389 before loading ‘glossaries-extra’}%
```

```
10390 \fi
```

```
10391 }
```

```
10392 {}
```

tibleglossentry Override style compatibility macros:

```
10393 \def\compatibleglossentry#1#2{%
```

```
10394 \toks@{#2}%
```

```
10395 \protected@edef\do@glossentry{%
```

```
10396 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10397 {\noexpand\glsnamefont
```

```
10398 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```

10399    {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10400    {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10401    {\the\toks@}%
10402  }%
10403  \@do@glossentry
10404 }

```

lesubglossentry

```

10405 \def\compatiblesubglossentry#1#2#3{%
10406   \toks@{#3}%
10407   \protected@edef\@do@subglossentry{%
10408     \noexpand\accsuppglossarysubentryfield{\number#1}%
10409     {#2}%
10410     {\noexpand\glsnamefont
10411      {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}}%
10412     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10413     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10414     {\the\toks@}%
10415   }%
10416   \@do@subglossentry
10417 }

```

Required packages:

```

10418 \RequirePackage{glossaries}
10419 \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:

```

10420 \define@key{glossentry}{access}{%
10421   \def\@glo@access{#1}%
10422 }

```

textaccess The replacement text corresponding to the text key:

```

10423 \define@key{glossentry}{textaccess}{%
10424   \def\@glo@textaccess{#1}%
10425 }

```

firstaccess The replacement text corresponding to the first key:

```

10426 \define@key{glossentry}{firstaccess}{%
10427   \def\@glo@firstaccess{#1}%
10428 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10429 \define@key{glossentry}{pluralaccess}{%
10430   \def\@glo@pluralaccess{#1}%
10431 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

10432 \define@key{glossentry}{firstpluralaccess}{%
10433   \def\@glo@firstpluralaccess{#1}%
10434 }
```

symbolaccess The replacement text corresponding to the symbol key:

```

10435 \define@key{glossentry}{symbolaccess}{%
10436   \def\@glo@symbolaccess{#1}%
10437 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```

10438 \define@key{glossentry}{symbolpluralaccess}{%
10439   \def\@glo@symbolpluralaccess{#1}%
10440 }
```

descriptionaccess The replacement text corresponding to the description key:

```

10441 \define@key{glossentry}{descriptionaccess}{%
10442   \def\@glo@descaccess{#1}%
10443 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```

10444 \define@key{glossentry}{descriptionpluralaccess}{%
10445   \def\@glo@descpluralaccess{#1}%
10446 }
```

shortaccess The replacement text corresponding to the short key:

```

10447 \define@key{glossentry}{shortaccess}{%
10448   \def\@glo@shortaccess{#1}%
10449 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```

10450 \define@key{glossentry}{shortpluralaccess}{%
10451   \def\@glo@shortpluralaccess{#1}%
10452 }
```

longaccess The replacement text corresponding to the long key:

```

10453 \define@key{glossentry}{longaccess}{%
10454   \def\@glo@longaccess{#1}%
10455 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```

10456 \define@key{glossentry}{longpluralaccess}{%
10457   \def\@glo@longpluralaccess{#1}%
10458 }
```


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={\glsacccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```

10459 \appto\@gls@keymap{,%
10460   {access}{access},%
10461   {textaccess}{textaccess},%
10462   {firstaccess}{firstaccess},%
10463   {pluralaccess}{pluralaccess},%
10464   {firstpluralaccess}{firstpluralaccess},%
10465   {symbolaccess}{symbolaccess},%
10466   {symbolpluralaccess}{symbolpluralaccess},%
10467   {descaccess}{descaccess},%
10468   {descpluralaccess}{descpluralaccess},%
10469   {shortaccess}{shortaccess},%
10470   {shortpluralaccess}{shortpluralaccess},%
10471   {longaccess}{longaccess},%
10472   {longpluralaccess}{longpluralaccess}%
10473 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```

10474 \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):

```

10475 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10476 \renewcommand*{\@newglossaryentryprehook}{%
10477   \@gls@oldnewglossaryentryprehook
10478   \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```

10479   \def\@glo@textaccess{\@glo@access}%
10480   \def\@glo@firstaccess{\@glo@access}%
10481   \def\@glo@pluralaccess{\@glo@textaccess}%
10482   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10483   \def\@glo@symbolaccess{\relax}%
10484   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10485   \def\@glo@descaccess{\relax}%
10486   \def\@glo@descpluralaccess{\@glo@descaccess}%
10487   \def\@glo@shortaccess{\relax}%
10488   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10489   \def\@glo@longaccess{\relax}%
10490   \def\@glo@longpluralaccess{\@glo@longaccess}%
10491 }
```

Add to the end hook:

```

10492 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10493 \renewcommand*{\@newglossaryentryposthook}{%
10494   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```

10495 \expandafter
10496   \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10497     \@glo@access}%
10498 \expandafter
10499   \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10500     \@glo@textaccess}%
10501 \expandafter
10502   \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10503     \@glo@firstaccess}%
10504 \expandafter
10505   \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10506     \@glo@pluralaccess}%
10507 \expandafter
10508   \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10509     \@glo@firstpluralaccess}%
10510 \expandafter
10511   \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10512     \@glo@symbolaccess}%
10513 \expandafter
10514   \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10515     \@glo@symbolpluralaccess}%
10516 \expandafter
10517   \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10518     \@glo@descaccess}%
10519 \expandafter
10520   \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10521     \@glo@descpluralaccess}%
10522 \expandafter
10523   \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10524     \@glo@shortaccess}%
10525 \expandafter
10526   \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10527     \@glo@shortpluralaccess}%
10528 \expandafter
10529   \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10530     \@glo@longaccess}%
10531 \expandafter
10532   \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10533     \@glo@longpluralaccess}%
10534 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10535 \newcommand*\glsentryaccess}[1]{%
10536   \@gls@entry@field{#1}{access}%
10537 }

```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
10538 \newcommand*{\glsentrytextaccess}[1]{%
10539   \@gls@entry@field{#1}{textaccess}%
10540 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10541 \newcommand*{\glsentryfirstaccess}[1]{%
10542   \@gls@entry@field{#1}{firstaccess}%
10543 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10544 \newcommand*{\glsentrypluralaccess}[1]{%
10545   \@gls@entry@field{#1}{pluralaccess}%
10546 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10547 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10548   \csname glo@#1@firstpluralaccess\endcsname
10549 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10550 \newcommand*{\glsentrysymbolaccess}[1]{%
10551   \@gls@entry@field{#1}{symbolaccess}%
10552 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10553 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10554   \@gls@entry@field{#1}{symbolpluralaccess}%
10555 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10556 \newcommand*{\glsentrydescaccess}[1]{%
10557   \@gls@entry@field{#1}{descaccess}%
10558 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10559 \newcommand*{\glsentrydescpluralaccess}[1]{%
10560   \@gls@entry@field{#1}{descaccess}%
10561 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10562 \newcommand*{\glsentryshortaccess}[1]{%
10563   \@gls@entry@field{#1}{shortaccess}%
10564 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10565 \newcommand*{\glsentryshortpluralaccess}[1]{%
10566   \@gls@entry@field{#1}{shortpluralaccess}%
10567 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10568 \newcommand*{\glsentrylongaccess}[1]{%
10569   \@gls@entry@field{#1}{longaccess}%
10570 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10571 \newcommand*{\glsentrylongpluralaccess}[1]{%
10572   \@gls@entry@field{#1}{longpluralaccess}%
10573 }
```

`\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.)

```
10574 \newcommand*{\glsaccsupp}[2]{%
10575   \BeginAccSupp{ActualText={#1}}#2\EndAccSupp{}}%
10576 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
10577 \newcommand*{\xglsaccsupp}[2]{%
10578   \protected@edef\@gls@replacementtext{#1}%
10579   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10580 }
```

`@access@display`

```
10581 \newcommand*{\@gls@access@display}[2]{%
10582   \protected@edef\@glo@access{#2}%
10583   \ifx\@glo@access\@gls@noaccess
10584     #1%
10585   \else
10586     \xglsaccsupp{\@glo@access}{#1}%
10587   \fi
10588 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10589 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10590   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10591 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10592 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10593   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10594 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```

10595 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
10596   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10597 }

```

staccessdisplay As above but for the firstaccess replacement text.

```

10598 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
10599   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
10600 }

```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```

10601 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
10602   \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
10603 }

```

olaccessdisplay As above but for the symbolaccess replacement text.

```

10604 \DeclareRobustCommand*\glsymbolaccessdisplay}[2]{%
10605   \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
10606 }

```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```

10607 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
10608   \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}%
10609 }

```

onaccessdisplay As above but for the descriptionaccess replacement text.

```

10610 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10611   \@gls@access@display{#1}{\glsentrydescaccess{#2}}%
10612 }

```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

10613 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10614   \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}%
10615 }

```

rtaccessdisplay As above but for the shortaccess replacement text.

```

10616 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10617   \@gls@access@display{#1}{\glsentryshortaccess{#2}}%
10618 }

```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```

10619 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10620   \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}%
10621 }

```

ngaccessdisplay As above but for the longaccess replacement text.

```

10622 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10623   \@gls@access@display{#1}{\glsentrylongaccess{#2}}%
10624 }

```

alaccessdisplay As above but for the longpluralaccess replacement text.

```
10625 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10626   \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}%
10627 }
```

lsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
10628 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10629   \ifundefined{gls#1accessdisplay}%
10630   {%
10631     \PackageError{glossaries-accsupp}{No accessibility support
10632       for key ‘#1’}{%
10633     }%
10634   {%
10635     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10636   }%
10637 }
```

efault@entryfmt Redefine the default entry format to use accessibility information

```
10638 \renewcommand*\@@gls@default@entryfmt}[2]{%
10639   \ifdefempty\glscustomtext
10640   {%
10641     \glsifplural
10642     {%
```

Plural form

```
10643     \glscapscase
10644     {%
```

Don't adjust case

```
10645     \ifglsused\glslabel
10646     {%
```

Subsequent use

```
10647     #2{\glspluralaccessdisplay
10648         {\glsentryplural{\glslabel}}{\glslabel}}%
10649     {\glsdescriptionpluralaccessdisplay
10650         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10651     {\glsymbolpluralaccessdisplay
10652         {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10653     {\glsinsert}%
10654   }%
10655   {%
```

First use

```
10656     #1{\glsfirstpluralaccessdisplay
10657         {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10658     {\glsdescriptionpluralaccessdisplay
10659         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10660     {\glsymbolpluralaccessdisplay
```

```

10661          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10662      {\glsinsert}%
10663  }%
10664  }%
10665  {%

```

Make first letter upper case

```

10666      \ifglsused\glslabel
10667  {%

```

Subsequent use.

```

10668      #2{\glspluralaccessdisplay
10669          {\Glsentryplural{\glslabel}}{\glslabel}}%
10670      {\glsdescriptionpluralaccessdisplay
10671          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10672      {\glsymbolpluralaccessdisplay
10673          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10674      {\glsinsert}%
10675  }%
10676  {%

```

First use

```

10677      #1{\glsfirstpluralaccessdisplay
10678          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10679      {\glsdescriptionpluralaccessdisplay
10680          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10681      {\glsymbolpluralaccessdisplay
10682          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10683      {\glsinsert}%
10684  }%
10685  }%
10686  {%

```

Make all upper case

```

10687      \ifglsused\glslabel
10688  {%

```

Subsequent use

```

10689      \MakeUppercase{%
10690      #2{\glspluralaccessdisplay
10691          {\glsentryplural{\glslabel}}{\glslabel}}%
10692      {\glsdescriptionpluralaccessdisplay
10693          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10694      {\glsymbolpluralaccessdisplay
10695          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10696      {\glsinsert}}%
10697  }%
10698  {%

```

First use

```

10699      \MakeUppercase{%
10700      #1{\glsfirstpluralaccessdisplay

```

```

10701          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10702          {\glsdescriptionpluralaccessdisplay
10703          {\glsentrydescplural{\glslabel}}{\glslabel}}%
10704          {\glssymbolpluralaccessdisplay
10705          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10706          {\glsinsert}}}%
10707      }%
10708  }%
10709 }%
10710 {%

```

Singular form

```

10711      \glscapscase
10712      {%

```

Don't adjust case

```

10713      \ifglused\glslabel
10714      {%

```

Subsequent use

```

10715      #2{\glstextaccessdisplay
10716          {\glsentrytext{\glslabel}}{\glslabel}}%
10717          {\glsdescriptionaccessdisplay
10718          {\glsentrydesc{\glslabel}}{\glslabel}}%
10719          {\glssymbolaccessdisplay
10720          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10721          {\glsinsert}}%
10722      }%
10723      {%

```

First use

```

10724      #1{\glsfirstaccessdisplay
10725          {\glsentryfirst{\glslabel}}{\glslabel}}%
10726          {\glsdescriptionaccessdisplay
10727          {\glsentrydesc{\glslabel}}{\glslabel}}%
10728          {\glssymbolaccessdisplay
10729          {\glsentrysymbol{\glslabel}}{\glslabel}}%
10730          {\glsinsert}}%
10731      }%
10732      }%
10733      {%

```

Make first letter upper case

```

10734      \ifglused\glslabel
10735      {%

```

Subsequent use

```

10736      #2{\glstextaccessdisplay
10737          {\Glsentrytext{\glslabel}}{\glslabel}}%
10738          {\glsdescriptionaccessdisplay
10739          {\Glsentrydesc{\glslabel}}{\glslabel}}%
10740          {\glssymbolaccessdisplay

```



```

10741         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10742     {\glsinsert}%
10743 }%
10744 {%

```

First use

```

10745     #1{\glsfirstaccessdisplay
10746         {\glsentryfirst{\glslabel}}{\glslabel}}%
10747     {\glsdescriptionaccessdisplay
10748         {\glsentrydesc{\glslabel}}{\glslabel}}%
10749     {\glsymbolaccessdisplay
10750         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10751     {\glsinsert}%
10752 }%
10753 }%
10754 {%

```

Make all upper case

```

10755     \ifglsused\glslabel
10756     {%

```

Subsequent use

```

10757     \MakeUppercase{%
10758     #2{\glsfirstaccessdisplay
10759         {\glsentrytext{\glslabel}}{\glslabel}}%
10760     {\glsdescriptionaccessdisplay
10761         {\glsentrydesc{\glslabel}}{\glslabel}}%
10762     {\glsymbolaccessdisplay
10763         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10764     {\glsinsert}}%
10765 }%
10766 {%

```

First use

```

10767     \MakeUppercase{%
10768     #1{\glsfirstaccessdisplay
10769         {\glsentryfirst{\glslabel}}{\glslabel}}%
10770     {\glsdescriptionaccessdisplay
10771         {\glsentrydesc{\glslabel}}{\glslabel}}%
10772     {\glsymbolaccessdisplay
10773         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10774     {\glsinsert}}%
10775 }%
10776 }%
10777 }%
10778 }%
10779 {%

```

Custom text provided in \glsdisp

```

10780     \ifglsused{\glslabel}%
10781     {%

```

Subsequent use

```

10782      #2{\glscustomtext}%
10783      {\glsdescriptionaccessdisplay
10784       {\glsentrydesc{\glslabel}}{\glslabel}}%
10785      {\glssymbolaccessdisplay
10786       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10787      {\glsinsert}%
10788    }%
10789    {%

```

First use

```

10790      #1{\glscustomtext}%
10791      {\glsdescriptionaccessdisplay
10792       {\glsentrydesc{\glslabel}}{\glslabel}}%
10793      {\glssymbolaccessdisplay
10794       {\glsentrysymbol{\glslabel}}{\glslabel}}%
10795      {\glsinsert}%
10796    }%
10797  }%
10798 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10799 \renewcommand*{\glsgenentryfmt}{%
10800   \ifdefempty\glscustomtext
10801   {%
10802     \glsifplural
10803     {%

```

Plural form

```

10804     \glscapscase
10805     {%

```

Don't adjust case

```

10806     \ifglsused\glslabel
10807     {%

```

Subsequent use

```

10808       \glspluralaccessdisplay
10809       {\glsentryplural{\glslabel}}{\glslabel}%
10810       \glsinsert
10811     }%
10812     {%

```

First use

```

10813       \glsfirstpluralaccessdisplay
10814       {\glsentryfirstplural{\glslabel}}{\glslabel}%
10815       \glsinsert
10816     }%
10817   }%
10818   {%

```

Make first letter upper case

```
10819      \ifglused\glslabel
10820      {%
```

Subsequent use.

```
10821      \glspluralaccessdisplay
10822      {\Glentryplural{\glslabel}}{\glslabel}%
10823      \glinsert
10824      }%
10825      {%
```

First use

```
10826      \glfirstpluralaccessdisplay
10827      {\Glentryfirstplural{\glslabel}}{\glslabel}%
10828      \glinsert
10829      }%
10830      }%
10831      {%
```

Make all upper case

```
10832      \ifglused\glslabel
10833      {%
```

Subsequent use

```
10834      \glspluralaccessdisplay
10835      {\mfirstucMakeUppercase{\glentryplural{\glslabel}}}%
10836      {\glslabel}%
10837      \mfirstucMakeUppercase{\glinsert}%
10838      }%
10839      {%
```

First use

```
10840      \glfirstpluralacessdisplay
10841      {\mfirstucMakeUppercase{\glentryfirstplural{\glslabel}}}%
10842      {\glslabel}%
10843      \mfirstucMakeUppercase{\glinsert}%
10844      }%
10845      }%
10846      }%
10847      {%
```

Singular form

```
10848      \glscapscase
10849      {%
```

Don't adjust case

```
10850      \ifglused\glslabel
10851      {%
```

Subsequent use

```
10852      \glstextaccessdisplay{\glentrytext{\glslabel}}{\glslabel}%
10853      \glinsert
```

```

10854      }%
10855      {%

```

First use

```

10856      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10857      \glsinsert
10858      }%
10859      }%
10860      {%

```

Make first letter upper case

```

10861      \ifglsused\glslabel
10862      {%

```

Subsequent use

```

10863      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10864      \glsinsert
10865      }%
10866      {%

```

First use

```

10867      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10868      \glsinsert
10869      }%
10870      }%
10871      {%

```

Make all upper case

```

10872      \ifglsused\glslabel
10873      {%

```

Subsequent use

```

10874      \glstextaccessdisplay
10875      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10876      \mfirstucMakeUppercase{\glsinsert}%
10877      }%
10878      {%

```

First use

```

10879      \glsfirstaccessdisplay
10880      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10881      \mfirstucMakeUppercase{\glsinsert}%
10882      }%
10883      }%
10884      }%
10885      }%
10886      {%

```

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.

```

10887      \glscustomtext\glsinsert
10888      }%
10889      }

```

`\glsgenacfmt` Redefine to include accessibility information.

```
10890 \renewcommand*{\glsgenacfmt}{%
10891   \ifdefempty\glscustomtext
10892   {%
10893     \ifglused\glslabel
10894     {%
```

Subsequent use:

```
10895     \glsifplural
10896     {%
```

Subsequent plural form:

```
10897     \glscapscase
10898     {%
```

Subsequent plural form, don't adjust case:

```
10899     \acronymfont
10900     {\glsshortpluralaccessdisplay
10901      {\glentryshortpl{\glslabel}}{\glslabel}}%
10902     \glsinsert
10903   }%
10904   {%
```

Subsequent plural form, make first letter upper case:

```
10905     \acronymfont
10906     {\glsshortpluralaccessdisplay
10907      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10908     \glsinsert
10909   }%
10910   {%
```

Subsequent plural form, all caps:

```
10911     \mfirstucMakeUppercase
10912     {\acronymfont
10913      {\glsshortpluralaccessdisplay
10914       {\glentryshortpl{\glslabel}}{\glslabel}}%
10915      \glsinsert}%
10916   }%
10917   }%
10918   {%
```

Subsequent singular form

```
10919     \glscapscase
10920     {%
```

Subsequent singular form, don't adjust case:

```
10921     \acronymfont
10922     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10923     \glsinsert
10924   }%
10925   {%
```

Subsequent singular form, make first letter upper case:

```
10926      \acronymfont
10927      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10928      \glsinsert
10929      }%
10930      {%
```

Subsequent singular form, all caps:

```
10931      \mfirstucMakeUppercase
10932      {\acronymfont{%
10933      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10934      \glsinsert}%
10935      }%
10936      }%
10937      }%
10938      {%
```

First use:

```
10939      \glsifplural
10940      {%
```

First use plural form:

```
10941      \glscapscase
10942      {%
```

First use plural form, don't adjust case:

```
10943      \genplacrfullformat{\glslabel}{\glsinsert}%
10944      }%
10945      {%
```

First use plural form, make first letter upper case:

```
10946      \Genplacrfullformat{\glslabel}{\glsinsert}%
10947      }%
10948      {%
```

First use plural form, all caps:

```
10949      \mfirstucMakeUppercase
10950      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10951      }%
10952      }%
10953      {%
```

First use singular form

```
10954      \glscapscase
10955      {%
```

First use singular form, don't adjust case:

```
10956      \genacrfullformat{\glslabel}{\glsinsert}%
10957      }%
10958      {%
```

First use singular form, make first letter upper case:

```
10959      \Genacrfullformat{\glslabel}{\glsinsert}%
10960      }%
10961      {%
```

First use singular form, all caps:

```
10962      \mfirstucMakeUppercase
10963      {\genacrfullformat{\glslabel}{\glsinsert}}%
10964      }%
10965      }%
10966      }%
10967      }%
10968      {%
```

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.

```
10969      \glscustomtext
10970      }%
10971 }
```

enacrfullformat Redefine to include accessibility information.

```
10972 \renewcommand*{\genacrfullformat}[2]{%
10973   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10974   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
10975 }
```

enacrfullformat Redefine to include accessibility information.

```
10976 \renewcommand*{\Genacrfullformat}[2]{%
10977   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10978   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
10979 }
```

placrfullformat Redefine to include accessibility information.

```
10980 \renewcommand*{\genplacrfullformat}[2]{%
10981   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10982   (\glsshortpluralaccessdisplay
10983     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10984 }
```

placrfullformat Redefine to include accessibility information.

```
10985 \renewcommand*{\Genplacrfullformat}[2]{%
10986   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10987   (\glsshortpluralaccessdisplay
10988     {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1}}%
10989 }
```

\@acrshort

```
10990 \def\@acrshort#1#2[#3]{%
10991   \glsdoifexists{#2}%
```

```

10992 {%
10993   \let\do@gl@s@link@checkfirsthyper\relax

10994   \let\gl@sifplural\@secondoftwo
10995   \let\gl@scapscase\@firstofthree
10996   \let\gl@sinsert\@empty
10997   \def\glscustomtext{%
10998     \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10999   }%

  Call \@gl@s@link
11000   \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
11001 }%

11002 \glspostlinkhook
11003 }

```

\@Acrshort

```

11004 \def\@Acrshort#1#2[#3]{%
11005   \gl@sdoifexists{#2}%
11006   {%
11007     \let\do@gl@s@link@checkfirsthyper\relax

11008     \let\gl@sifplural\@secondoftwo
11009     \let\gl@scapscase\@secondofthree
11010     \let\gl@sinsert\@empty
11011     \def\glscustomtext{%
11012       \acronymfont{\glsshortaccessdisplay{\Gl@sentryshort{#2}}{#2}}#3%
11013     }%

    Call \@gl@s@link
11014     \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
11015   }%

11016   \glspostlinkhook
11017 }

```

\@ACRshort

```

11018 \def\@ACRshort#1#2[#3]{%
11019   \gl@sdoifexists{#2}%
11020   {%
11021     \let\do@gl@s@link@checkfirsthyper\relax

11022     \let\gl@sifplural\@secondoftwo
11023     \let\gl@scapscase\@thirdofthree
11024     \let\gl@sinsert\@empty
11025     \def\glscustomtext{%
11026       \acronymfont{\glsshortaccessdisplay
11027         {\MakeUppercase{\gl@sentryshort{#2}}}{#2}}#3%
11028     }%

```



```

Call \@gls@link
11029   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11030   }%

11031   \glspostlinkhook
11032 }

```

\@acrlong

```

11033 \def\@acrlong#1#2[#3]{%
11034   \glsdoifexists{#2}%
11035   {%
11036     \let\do@gls@link@checkfirsthyper\relax

11037     \let\glsifplural\@secondoftwo
11038     \let\glscapscase\@firstofthree
11039     \let\glsinsert\@empty
11040     \def\glscustomtext{%
11041       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
11042     }%

```

```

Call \@gls@link
11043   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11044   }%

11045   \glspostlinkhook
11046 }

```

\@Acrlong

```

11047 \def\@Acrlong#1#2[#3]{%
11048   \glsdoifexists{#2}%
11049   {%
11050     \let\do@gls@link@checkfirsthyper\relax

11051     \let\glsifplural\@secondoftwo
11052     \let\glscapscase\@firstofthree
11053     \let\glsinsert\@empty
11054     \def\glscustomtext{%
11055       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
11056     }%

```

```

Call \@gls@link
11057   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11058   }%

11059   \glspostlinkhook
11060 }

```

\@ACRlong

```

11061 \def\@ACRlong#1#2[#3]{%
11062   \glsdoifexists{#2}%
11063   {%
11064     \let\do@gls@link@checkfirsthyper\relax

```

```

11065 \let\glsifplural\@secondoftwo
11066 \let\glsifscapscase\@firstofthree
11067 \let\glsinsert\@empty
11068 \def\glscustomtext{%
11069 \acronymfont{\glslongaccessdisplay{%
11070 \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
11071 }%

Call \@gls@link
11072 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11073 }%

11074 \glspostlinkhook
11075 }

```

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.

```

11076 \renewcommand*{\glossentryname}[1]{%
11077 \glsdoifexists{#1}%
11078 {%
11079 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
11080 }%
11081 }

11082 \renewcommand*{\glossentrydesc}[1]{%
11083 \glsdoifexists{#1}%
11084 {%
11085 \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
11086 }%
11087 }

11088 \renewcommand*{\glossentrydesc}[1]{%
11089 \glsdoifexists{#1}%
11090 {%
11091 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
11092 }%
11093 }

11094 \renewcommand*{\Glossentrydesc}[1]{%
11095 \glsdoifexists{#1}%
11096 {%
11097 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
11098 }%
11099 }

```

```

11100 \renewcommand*{\glossentrysymbol}[1]{%
11101   \glsdoifexists{#1}%
11102   {%
11103     \glssymbolaccessdisplay{\glentrysymbol{#1}}{#1}%
11104   }%
11105 }

11106 \renewcommand*{\Glossentrysymbol}[1]{%
11107   \glsdoifexists{#1}%
11108   {%
11109     \glssymbolaccessdisplay{\Glentrysymbol{#1}}{#1}%
11110   }%
11111 }

```

ssaryentryfield

```

11112 \newcommand*{\accsuppglossaryentryfield}[5]{%
11113   \glossaryentryfield{#1}%
11114   {\glsnameaccessdisplay{#2}{#1}}%
11115   {\glsdescriptionaccessdisplay{#3}{#1}}%
11116   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
11117 }

```

rysubentryfield

```

11118 \newcommand*{\accsuppglossarysubentryfield}[6]{%
11119   \glossarysubentryfield{#1}{#2}%
11120   {\glsnameaccessdisplay{#3}{#2}}%
11121   {\glsdescriptionaccessdisplay{#4}{#2}}%
11122   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
11123 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

11124 \renewacronymstyle{long-short}%
11125 {%

```

Check for long form in case this is a mixed glossary.

```

11126   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11127 }%
11128 {%
11129   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11130   \renewcommand*{\genacrfullformat}[2]{%
11131     \glslongaccessdisplay{\glentrylong{##1}}{##1}##2\space
11132     (\glsshortaccessdisplay
11133       {\protect\firstacronymfont{\glentryshort{##1}}}{##1})%
11134   }%
11135   \renewcommand*{\Genacrfullformat}[2]{%

```

```

11136 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11137 (\glsshortaccessdisplay
11138   {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
11139 }%
11140 \renewcommand*{\genplacrfullformat}[2]{%
11141   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11142   (\glsshortpluralaccessdisplay
11143     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11144   }%
11145 \renewcommand*{\Genplacrfullformat}[2]{%
11146   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11147   (\glsshortpluralaccessdisplay
11148     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11149   }%
11150 \renewcommand*{\acronymentry}[1]{%
11151   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
11152 \renewcommand*{\acronymsort}[2]{##1}%
11153 \renewcommand*{\acronymfont}[1]{##1}%
11154 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11155 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11156 }

```

short-long (*short*) (*long*) acronym style.

```

11157 \renewacronymstyle{short-long}%
11158 {%

```

Check for long form in case this is a mixed glossary.

```

11159 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11160 }%
11161 {%
11162 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11163 \renewcommand*{\genacrfullformat}[2]{%
11164   \glsshortaccessdisplay
11165     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
11166     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11167   }%
11168 \renewcommand*{\Genacrfullformat}[2]{%
11169   \glsshortaccessdisplay
11170     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
11171     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11172   }%
11173 \renewcommand*{\genplacrfullformat}[2]{%
11174   \glsshortpluralaccessdisplay
11175     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
11176     (\glslongpluralaccessdisplay
11177       {\glsentrylongpl{##1}}{##1}})%
11178   }%
11179 \renewcommand*{\Genplacrfullformat}[2]{%
11180   \glsshortpluralaccessdisplay
11181     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

11182 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11183 }%
11184 \renewcommand*{\acronymentry}[1]{%
11185   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11186 \renewcommand*{\acronymsort}[2]{##1}%
11187 \renewcommand*{\acronymfont}[1]{##1}%
11188 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11189 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11190 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

11191 \renewacronymstyle{long-short-desc}%
11192 {%
11193   \GlsUseAcrEntryDispStyle{long-short}%
11194 }%
11195 {%
11196   \GlsUseAcrStyleDefs{long-short}%
11197   \renewcommand*{\GenericAcronymFields}{}%
11198   \renewcommand*{\acronymsort}[2]{##2}%
11199   \renewcommand*{\acronymentry}[1]{%
11200     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11201     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11202 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11203 \renewacronymstyle{long-sc-short-desc}%
11204 {%
11205   \GlsUseAcrEntryDispStyle{long-sc-short}%
11206 }%
11207 {%
11208   \GlsUseAcrStyleDefs{long-sc-short}%
11209   \renewcommand*{\GenericAcronymFields}{}%
11210   \renewcommand*{\acronymsort}[2]{##2}%
11211   \renewcommand*{\acronymentry}[1]{%
11212     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11213     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11214 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11215 \renewacronymstyle{long-sm-short-desc}%
11216 {%
11217   \GlsUseAcrEntryDispStyle{long-sm-short}%
11218 }%
11219 {%
11220   \GlsUseAcrStyleDefs{long-sm-short}%
11221   \renewcommand*{\GenericAcronymFields}{}%

```

```

11222 \renewcommand*{\acronymsort}[2]{##2}%
11223 \renewcommand*{\acronymentry}[1]{%
11224   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11225   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11226 }

```

short-long-desc *(short)* (*long*) acronym style that has an accompanying description (which the user needs to supply).

```

11227 \renewacronymstyle{short-long-desc}%
11228 {%
11229   \GlsUseAcrEntryDispStyle{short-long}%
11230 }%
11231 {%
11232   \GlsUseAcrStyleDefs{short-long}%
11233   \renewcommand*{\GenericAcronymFields}{}%
11234   \renewcommand*{\acronymsort}[2]{##2}%
11235   \renewcommand*{\acronymentry}[1]{%
11236     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11237     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11238 }

```

short-long-desc *long* (\textsc{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11239 \renewacronymstyle{sc-short-long-desc}%
11240 {%
11241   \GlsUseAcrEntryDispStyle{sc-short-long}%
11242 }%
11243 {%
11244   \GlsUseAcrStyleDefs{sc-short-long}%
11245   \renewcommand*{\GenericAcronymFields}{}%
11246   \renewcommand*{\acronymsort}[2]{##2}%
11247   \renewcommand*{\acronymentry}[1]{%
11248     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11249     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11250 }

```

short-long-desc *long* (\textsmaller{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11251 \renewacronymstyle{sm-short-long-desc}%
11252 {%
11253   \GlsUseAcrEntryDispStyle{sm-short-long}%
11254 }%
11255 {%
11256   \GlsUseAcrStyleDefs{sm-short-long}%
11257   \renewcommand*{\GenericAcronymFields}{}%
11258   \renewcommand*{\acronymsort}[2]{##2}%
11259   \renewcommand*{\acronymentry}[1]{%
11260     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11261     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

11262 }

dua *<long>* only acronym style.

11263 \renewacronymstyle{dua}%

11264 {%

Check for long form in case this is a mixed glossary.

11265 \ifdefempty\glscustomtext

11266 {%

11267 \ifglshaslong{\glslabel}%

11268 {%

11269 \glsifplural

11270 {%

Plural form:

11271 \glscapscase

11272 {%

Plural form, don't adjust case:

11273 \gslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%

11274 \glsinsert

11275 }%

11276 {%

Plural form, make first letter upper case:

11277 \gslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%

11278 \glsinsert

11279 }%

11280 {%

Plural form, all caps:

11281 \gslongpluralaccessdisplay

11282 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%

11283 \mfirstucMakeUppercase{\glsinsert}%

11284 }%

11285 }%

11286 {%

Singular form

11287 \glscapscase

11288 {%

Singular form, don't adjust case:

11289 \gslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert

11290 }%

11291 {%

Subsequent singular form, make first letter upper case:

11292 \gslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert

11293 }%

11294 {%

Subsequent singular form, all caps:

```

11295         \glslongaccessdisplay
11296         {\mfirstucMakeUppercase
11297          {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11298         \mfirstucMakeUppercase{\glsinsert}%
11299     }%
11300 }%
11301 }%
11302 {%

```

Not an acronym:

```

11303     \glsgenentryfmt
11304 }%
11305 }%
11306 {\glscustomtext\glsinsert}%
11307}%
11308{%
11309 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11310 \renewcommand*{\acrfullfmt}[3]{%
11311     \glslink[##1]{##2}{%
11312         \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11313         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
11314 \renewcommand*{\Acrfullfmt}[3]{%
11315     \glslink[##1]{##2}{%
11316         \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11317         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}%
11318 \renewcommand*{\ACRfullfmt}[3]{%
11319     \glslink[##1]{##2}{%
11320         \glslongaccessdisplay
11321         {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11322         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}}}}}%
11323 \renewcommand*{\acrfullplfmt}[3]{%
11324     \glslink[##1]{##2}{%
11325         \glslongpluralaccessdisplay
11326         {\glsentrylongpl{##2}}{##2}##3\space
11327         (\glsshortpluralaccessdisplay
11328         {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
11329 \renewcommand*{\Acrfullplfmt}[3]{%
11330     \glslink[##1]{##2}{%
11331         \glslongpluralaccessdisplay
11332         {\Glsentrylongpl{##2}}{##2}##3\space
11333         (\glsshortpluralaccessdisplay
11334         {\acronymfont{\glsentryshortpl{##2}}}{##2}}}%
11335 \renewcommand*{\ACRfullplfmt}[3]{%
11336     \glslink[##1]{##2}{%
11337         \glslongpluralaccessdisplay
11338         {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11339         (\glsshortpluralaccessdisplay
11340         {\acronymfont{\glsentryshortpl{##2}}}{##2}}}}}%
11341 \renewcommand*{\glsentryfull}[1]{%

```



```

11342 \glslongaccessdisplay{\glsentrylong{##1}}\space
11343 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11344 }%
11345 \renewcommand*{\Glsentryfull}[1]{%
11346 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11347 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11348 }%
11349 \renewcommand*{\Glsentryfullpl}[1]{%
11350 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11351 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11352 }%
11353 \renewcommand*{\Glsentryfullpl}[1]{%
11354 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11355 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11356 }%
11357 \renewcommand*{\acronymentry}[1]{%
11358 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11359 \renewcommand*{\acronymsort}[2]{##1}%
11360 \renewcommand*{\acronymfont}[1]{##1}%
11361 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11362 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11363 \renewacronymstyle{dua-desc}%
11364 {%
11365 \GlsUseAcrEntryDispStyle{dua}%
11366 }%
11367 {%
11368 \GlsUseAcrStyleDefs{dua}%
11369 \renewcommand*{\GenericAcronymFields}{}%
11370 \renewcommand*{\acronymentry}[1]{%
11371 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11372 \renewcommand*{\acronymsort}[2]{##2}%
11373 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11374 \renewacronymstyle{footnote}%
11375 {%
11376 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11377 }%
11378 {%
11379 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11380 \glshyperfirstfalse
11381 \renewcommand*{\genacrfullformat}[2]{%
11382 \glsshortaccessdisplay
11383 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

11384 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11385 }%
11386 \renewcommand*{\Genacrfullformat}[2]{%
11387 \glsshortaccessdisplay
11388   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11389 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11390 }%
11391 \renewcommand*{\genplacrfullformat}[2]{%
11392 \glsshortpluralaccessdisplay
11393   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
11394 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11395 }%
11396 \renewcommand*{\Genplacrfullformat}[2]{%
11397 \glsshortpluralaccessdisplay
11398   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11399 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11400 }%
11401 \renewcommand*{\acronymentry}[1]{%
11402 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11403 \renewcommand*{\acronymsort}[2]{##1}%
11404 \renewcommand*{\acronymfont}[1]{##1}%
11405 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11406 \renewcommand*{\acrfullfmt}[3]{%
11407 \glslink{##1}{##2}{%
11408   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11409   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11410 \renewcommand*{\Acrfullfmt}[3]{%
11411 \glslink{##1}{##2}{%
11412   \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11413   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11414 \renewcommand*{\ACRfullfmt}[3]{%
11415 \glslink{##1}{##2}{%
11416   \glsshortaccessdisplay
11417     {\mfirstucMakeUppercase
11418     {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11419     (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11420 \renewcommand*{\acrfullplfmt}[3]{%
11421 \glslink{##1}{##2}{%
11422   \glsshortpluralaccessdisplay
11423     {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11424     (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11425 \renewcommand*{\Acrfullplfmt}[3]{%
11426 \glslink{##1}{##2}{%
11427   \glsshortpluralaccessdisplay
11428     {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11429     (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11430 \renewcommand*{\ACRfullplfmt}[3]{%
11431 \glslink{##1}{##2}{%

```

```

11432 \glsshortpluralaccessdisplay
11433 {\mfirstucMakeUppercase
11434 {\acronymfont{\glentryshortpl{##2}}}{##2}##3\space
11435 (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

Similarly for \glentryfull etc:
11436 \renewcommand*{\glentryfull}[1]{%
11437 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}\space
11438 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11439 \renewcommand*{\Glsentryfull}[1]{%
11440 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
11441 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11442 \renewcommand*{\glentryfullpl}[1]{%
11443 \glsshortpluralaccessdisplay
11444 {\acronymfont{\glentryshortpl{##1}}}{##1}\space
11445 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11446 \renewcommand*{\Glsentryfullpl}[1]{%
11447 \glsshortpluralaccessdisplay
11448 {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
11449 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11450 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11451 \renewacronymstyle{footnote-sc}%
11452 {%
11453 \GlsUseAcrEntryDispStyle{footnote}%
11454 }%
11455 {%
11456 \GlsUseAcrStyleDefs{footnote}%
11457 \renewcommand{\acronymentry}[1]{%
11458 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11459 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11460 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11461 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11462 \renewacronymstyle{footnote-sm}%
11463 {%
11464 \GlsUseAcrEntryDispStyle{footnote}%
11465 }%
11466 {%
11467 \GlsUseAcrStyleDefs{footnote}%
11468 \renewcommand{\acronymentry}[1]{%
11469 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11470 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11471 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11472 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11473 \renewacronymstyle{footnote-desc}%
11474 {%
11475   \GlsUseAcrEntryDisplayStyle{footnote}%
11476 }%
11477 {%
11478   \GlsUseAcrStyleDefs{footnote}%
11479   \renewcommand*{\GenericAcronymFields}{}%
11480   \renewcommand*{\acronymsort}[2]{##2}%
11481   \renewcommand*{\acronymentry}[1]{%
11482     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11483     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11484 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11485 \renewacronymstyle{footnote-sc-desc}%
11486 {%
11487   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11488 }%
11489 {%
11490   \GlsUseAcrStyleDefs{footnote-sc}%
11491   \renewcommand*{\GenericAcronymFields}{}%
11492   \renewcommand*{\acronymsort}[2]{##2}%
11493   \renewcommand*{\acronymentry}[1]{%
11494     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11495     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11496 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11497 \renewacronymstyle{footnote-sm-desc}%
11498 {%
11499   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11500 }%
11501 {%
11502   \GlsUseAcrStyleDefs{footnote-sm}%
11503   \renewcommand*{\GenericAcronymFields}{}%
11504   \renewcommand*{\acronymsort}[2]{##2}%
11505   \renewcommand*{\acronymentry}[1]{%
11506     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11507     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11508 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11509 \renewcommand*{\newacronymhook}{%
11510   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11511     \the\glskeylisttok}%
11512   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11513 }

1tNewAcronymDef Modify default style to use access text:

```
11514 \renewcommand*{\DefaultNewAcronymDef}{%
11515   \edef\@do@newglossaryentry{%
11516     \noexpand\newglossaryentry{\the\glslabeltok}%
11517     {%
11518       type=\acronymtype,%
11519       name={\the\glsshorttok},%
11520       description={\the\glslongtok},%
11521       descriptionaccess=\relax,%
11522       text={\the\glsshorttok},%
11523       access={\noexpand\@glo@textaccess},%
11524       sort={\the\glsshorttok},%
11525       short={\the\glsshorttok},%
11526       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11527       shortaccess={\the\glslongtok},%
11528       long={\the\glslongtok},%
11529       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11530       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11531       first={\noexpand\glslongaccessdisplay
11532         {\the\glslongtok}{\the\glslabeltok}\space
11533         (\noexpand\glsshortaccessdisplay
11534           {\the\glsshorttok}{\the\glslabeltok})},%
11535       plural={\the\glsshorttok\acrpluralsuffix},%
11536       firstplural={\noexpand\glslongpluralaccessdisplay
11537         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11538         (\noexpand\glsshortpluralaccessdisplay
11539           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11540       firstaccess=\relax,%
11541       firstpluralaccess=\relax,%
11542       textaccess={\noexpand\@glo@shortaccess},%
11543       \the\glskeylisttok
11544     }%
11545   }%
11546   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11547   \let\@org@gls@assign@plural\gls@assign@plural
11548   \let\@org@gls@assign@descplural\gls@assign@descplural
11549   \def\gls@assign@firstpl##1##2{%
11550     \@gls@expand@field{##1}{firstpl}{##2}%
11551   }%
11552   \def\gls@assign@plural##1##2{%
11553     \@gls@expand@field{##1}{plural}{##2}%
11554   }%
11555   \def\gls@assign@descplural##1##2{%
11556     \@gls@expand@field{##1}{descplural}{##2}%
11557   }%
11558   \@do@newglossaryentry
11559   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11560 \let\gls@assign@plural\@org@gls@assign@plural
11561 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11562 }

```

teNewAcronymDef

```

11563 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11564 \edef\@do@newglossaryentry{%
11565 \noexpand\newglossaryentry{\the\glslabeltok}%
11566 {%
11567 type=\acronymtype,%
11568 name={\noexpand\acronymfont{\the\glsshorttok}},%
11569 sort={\the\glsshorttok},%
11570 text={\the\glsshorttok},%
11571 short={\the\glsshorttok},%
11572 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11573 shortaccess={\the\glslongtok},%
11574 long={\the\glslongtok},%
11575 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11576 access={\noexpand\@glo@textaccess},%
11577 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11578 symbol={\the\glslongtok},%
11579 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11580 firstpluralaccess=\relax,
11581 textaccess={\noexpand\@glo@shortaccess},%
11582 \the\glskeylisttok
11583 }%
11584 }%
11585 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11586 \let\@org@gls@assign@plural\gls@assign@plural
11587 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11588 \def\gls@assign@firstpl##1##2{%
11589 \@@gls@expand@field{##1}{firstpl}{##2}%
11590 }%
11591 \def\gls@assign@plural##1##2{%
11592 \@@gls@expand@field{##1}{plural}{##2}%
11593 }%
11594 \def\gls@assign@symbolplural##1##2{%
11595 \@@gls@expand@field{##1}{symbolplural}{##2}%
11596 }%
11597 \do@newglossaryentry
11598 \let\gls@assign@plural\@org@gls@assign@plural
11599 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11600 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11601 }

```

onNewAcronymDef

```

11602 \renewcommand*{\DescriptionNewAcronymDef}{%
11603 \edef\@do@newglossaryentry{%
11604 \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11605 {%
11606     type=\acronymtype,%
11607     name={\noexpand
11608         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11609     access={\noexpand\@glo@textaccess},%
11610     sort={\the\glsshorttok},%
11611     short={\the\glsshorttok},%
11612     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11613     shortaccess={\the\glslongtok},%
11614     long={\the\glslongtok},%
11615     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11616     first={\the\glslongtok},%
11617     firstaccess=\relax,
11618     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11619     text={\the\glsshorttok},%
11620     textaccess={\the\glslongtok},%
11621     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11622     symbol={\noexpand\@glo@text},%
11623     symbolaccess={\noexpand\@glo@textaccess},%
11624     symbolplural={\noexpand\@glo@plural},%
11625     firstpluralaccess=\relax,
11626     textaccess={\noexpand\@glo@shortaccess},%
11627     \the\glskeylisttok}%
11628 }%
11629 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11630 \let\@org@gls@assign@plural\gls@assign@plural
11631 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11632 \def\gls@assign@firstpl##1##2{%
11633     \@gls@expand@field{##1}{firstpl}{##2}%
11634 }%
11635 \def\gls@assign@plural##1##2{%
11636     \@gls@expand@field{##1}{plural}{##2}%
11637 }%
11638 \def\gls@assign@symbolplural##1##2{%
11639     \@gls@expand@field{##1}{symbolplural}{##2}%
11640 }%
11641 \@do@newglossaryentry
11642 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11643 \let\gls@assign@plural\@org@gls@assign@plural
11644 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11645 }

```

teNewAcronymDef

```

11646 \renewcommand*{\FootnoteNewAcronymDef}{%
11647     \edef\@do@newglossaryentry{%
11648         \noexpand\newglossaryentry{\the\glslabeltok}%
11649         {%
11650             type=\acronymtype,%
11651             name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11652     sort={\the\glsshorttok},%
11653     text={\the\glsshorttok},%
11654     textaccess={\the\glslongtok},%
11655     access={\noexpand\@glo@textaccess},%
11656     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11657     short={\the\glsshorttok},%
11658     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11659     long={\the\glslongtok},%
11660     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11661     description={\the\glslongtok},%
11662     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11663     \the\glskeylisttok
11664   }%
11665 }%
11666 \let\@org@gls@assign@plural\gls@assign@plural
11667 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11668 \let\@org@gls@assign@descplural\gls@assign@descplural
11669 \def\gls@assign@firstpl##1##2{%
11670   \@@gls@expand@field{##1}{firstpl}{##2}%
11671 }%
11672 \def\gls@assign@plural##1##2{%
11673   \@@gls@expand@field{##1}{plural}{##2}%
11674 }%
11675 \def\gls@assign@descplural##1##2{%
11676   \@@gls@expand@field{##1}{descplural}{##2}%
11677 }%
11678 \do@newglossaryentry
11679 \let\gls@assign@plural\@org@gls@assign@plural
11680 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11681 \let\gls@assign@descplural\@org@gls@assign@descplural
11682 }

```

11NewAcronymDef

```

11683 \renewcommand*{\SmallNewAcronymDef}{%
11684   \edef\@do@newglossaryentry{%
11685     \noexpand\newglossaryentry{\the\glslabeltok}%
11686     {%
11687       type=\acronymtype,%
11688       name={\noexpand\acronymfont{\the\glsshorttok}},%
11689       access={\noexpand\@glo@symbolaccess},%
11690       sort={\the\glsshorttok},%
11691       short={\the\glsshorttok},%
11692       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11693       shortaccess={\the\glslongtok},%
11694       long={\the\glslongtok},%
11695       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11696       text={\noexpand\@glo@short},%
11697       textaccess={\noexpand\@glo@shortaccess},%
11698       plural={\noexpand\@glo@shortpl},%

```



```

11699     first={\the\glslongtok},%
11700     firstaccess=\relax,
11701     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11702     description={\noexpand\@glo@first},%
11703     descriptionplural={\noexpand\@glo@firstplural},%
11704     symbol={\the\glsshorttok},%
11705     symbolaccess={\the\glslongtok},%
11706     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11707     \the\glskeylisttok
11708   }%
11709 }%
11710 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11711 \let\@org@gls@assign@plural\gls@assign@plural
11712 \let\@org@gls@assign@descplural\gls@assign@descplural
11713 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11714 \def\gls@assign@firstpl##1##2{%
11715   \@@gls@expand@field{##1}{firstpl}{##2}%
11716 }%
11717 \def\gls@assign@plural##1##2{%
11718   \@@gls@expand@field{##1}{plural}{##2}%
11719 }%
11720 \def\gls@assign@descplural##1##2{%
11721   \@@gls@expand@field{##1}{descplural}{##2}%
11722 }%
11723 \def\gls@assign@symbolplural##1##2{%
11724   \@@gls@expand@field{##1}{symbolplural}{##2}%
11725 }%
11726 \do@newglossaryentry
11727 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11728 \let\gls@assign@plural\@org@gls@assign@plural
11729 \let\gls@assign@descplural\@org@gls@assign@descplural
11730 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11731 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
11732 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
11733 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
11734 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
11735 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

5.5 Debugging Commands

owglonameaccess

```
11736 \newcommand*{\showglonameaccess}[1]{%
11737   \expandafter\show\csname glo@\glsdetoklabel{#1}@access\endcsname
11738 }
```

owglotextaccess

```
11739 \newcommand*{\showglotextaccess}[1]{%
11740   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11741 }
```

glopluralaccess

```
11742 \newcommand*{\showglopluralaccess}[1]{%
11743   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11744 }
```

wglofirstaccess

```
11745 \newcommand*{\showglofirstaccess}[1]{%
11746   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11747 }
```

rstpluralaccess

```
11748 \newcommand*{\showglofirstpluralaccess}[1]{%
11749   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11750 }
```

glosymbolaccess

```
11751 \newcommand*{\showglosymbolaccess}[1]{%
11752   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11753 }
```

bolpluralaccess

```
11754 \newcommand*{\showglosymbolpluralaccess}[1]{%
11755   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11756 }
```

owglodescaccess

```
11757 \newcommand*{\showglodescaccess}[1]{%
11758   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11759 }
```

escpluralaccess

```
11760 \newcommand*{\showglodescpluralaccess}[1]{%
11761   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11762 }
```

wgloshortaccess

```
11763 \newcommand*{\showgloshortaccess}[1]{%  
11764   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11765 }
```

ortpluralaccess

```
11766 \newcommand*{\showgloshortpluralaccess}[1]{%  
11767   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11768 }
```

owglolongaccess

```
11769 \newcommand*{\showglolongaccess}[1]{%  
11770   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11771 }
```

ongpluralaccess

```
11772 \newcommand*{\showglolongpluralaccess}[1]{%  
11773   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11774 }
```

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.

```
11775 \NeedsTeXFormat{LaTeX2e}
11776 \ProvidesPackage{glossaries-babel}[2018/07/23 v4.41 (NLCT)]
```

Load tracklang to obtain language settings.

```
11777 \RequirePackage{tracklang}
11778 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11779 \AnyTrackedLanguages
11780 {%
11781   \ForEachTrackedDialect{\this@dialect}{%
11782     \IfTrackedLanguageFileExists{\this@dialect}%
11783       {glossaries-}% prefix
11784       {.ldf}%
11785       {%
11786         \RequireGlossariesLang{\CurrentTrackedTag}%
11787       }%
11788       {%
11789         \PackageWarningNoLine{glossaries}%
11790           {No language module detected for ‘\this@dialect’.\MessageBreak
11791             Language modules need to be installed separately.\MessageBreak
11792             Please check on CTAN for a bundle called\MessageBreak
11793             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11794       }%
11795     }%
11796   }%
11797 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11798 \NeedsTeXFormat{LaTeX2e}
11799 \ProvidesPackage{glossaries-polyglossia}[2018/07/23 v4.41 (NLCT)]
```

Load tracklang to obtain language settings.

```
11800 \RequirePackage{tracklang}
11801 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11802 \AnyTrackedLanguages
```

```

11803 {%
11804     \ForEachTrackedDialect{\this@dialect}{%
11805         \IfTrackedLanguageFileExists{\this@dialect}%
11806         {glossaries-}% prefix
11807         {.ldf}%
11808         {%
11809             \RequireGlossariesLang{\CurrentTrackedTag}%
11810         }%
11811         {%
11812             \PackageWarningNoLine{glossaries}%
11813             {No language module detected for ‘\this@dialect’.\MessageBreak
11814             Language modules need to be installed separately.\MessageBreak
11815             Please check on CTAN for a bundle called\MessageBreak
11816             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11817         }%
11818     }%
11819 }%
11820 {}%

```

Glossary

`makeindex` An indexing application. [9](#), [13](#), [29](#), [30](#), [181](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [9](#), [13](#), [29](#), [30](#), [181](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added .. 7
General: Added range facility in format key 116	1.12 (2008-03-08)
\writeist: Added spaces after \delimN and \delimR in ist file 163	\@GLSpl: now uses
1.04 (2007-08-03)	\glentrydescplural and
General: Added \glstextformat 100	\glentrysymbolplural instead of
1.05 (2007-08-10)	\glentrydesc and
\glossarysection: added \@mkboth to \glossarysection 43	\glentrysymbol 130
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 84	\@Glspl@: now uses
1.07 (2007-09-13)	\glentrydescplural and
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 114	\glentrysymbolplural instead of
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 160	\glentrydesc and
1.08 (2007-10-13)	\glentrysymbol 128
General: Added babel support 37	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 124
listgroup: changed listgroup style to use \glsgetgrouptitle 275	descriptionplural: new 66
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 276	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 84
1.1 (2008-02-22)	descriptionplural support added 84
\@glossarysection: numbered sections and auto label added 44	symbolplural support added 84
\@gls@tmpb: changed \toksdef to \newtoks 118	\Glsentrydescplural: New 153
\@gls@toc: numberline added 46	\glentrydescplural: New 153
\@p@glossarysection: numbered sections and auto label added 44	\Glsentrysymbolplural: New 154
General: amsgen now loaded (\new@ifnextchar needed) 4	\glentrysymbolplural: New 154
translate: translate option added 26	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 242
\setglossarysection: new 44	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 248
numberedsection: numberedsection package option added 8	symbolplural: new 67

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	131–139
\ACRfullpl: new	223
\Acrfullpl: new	222
\acrfullpl: new	222
\acrpluralsuffix: New	220
\gls@defglossaryentry: Changed default first value	84
Changed default firstplural value	84
Removed restriction on only using \newglossaryentry in the preamble	89
\newacronym: Removed restriction on only using \newacronym in the preamble	220
1.14 (2008-06-17)	
\@gls@hypergroup: new	270
General: added nonumberlist key to \printglossary	207
added numberedsection key to \printglossary	206
\firstacronymfont: new	223
\glsautoprefix: new	8
\glsnavhyperlink: changed \edef to \protected@edef	269
\glsnavhypertarget: added write to aux file	269
\glsnavigation: changed to only use labels for groups that are present ..	271
1.15 (2008-08-15)	
\@gls@link: added \glslabel	114
\gls@defglossaryentry: check for \@glo@first in description	88
check for \@glo@text in symbol	88
\gls@hypergroup: new	270
\glsnavhypertarget: added check if rerun required	269
\glssettoctitle: new	36
\printglossary: changed the way the TOC title is set	191
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	127
\@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
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	129
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	126
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	131
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	128
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page	124
\gls@defglossaryentry: Changed def to let	84
1.17 (2008-12-26)	
\@do@esc@wrglossary: new	185
\@do@seeglossary: new	189
\@glo@storeentry: new	90
\@gls@glossary: changed definition to use \index instead of \@index	182
\@glsdefaultplural: new	71
\@glsdefaultsort: new	71
\@glshypernumber: new	217
\@glsnoname: new	70
\@glsnonextpages: new	207
General: added xindy support	29
parent: new	68
see: new	68
\gls@defglossaryentry: added nonumberlist key	84
added parent key	84
added see key	84
Stored main part of entry format when entry is defined	89
\gls@suffixF: new	41
\gls@suffixFF: new	41
\gls@wrglossary: modified to allow for xindy support	182

\glshyperlink: new	159	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	242
\glshypernumber: modified to allow material to be attached to location	217	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	248
\glshnavhyperlink: replaced \hyperlink to \@glslink	269	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	251
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	269	2.01 (2009 May 30) \@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	115
\glsssee: new	190	\forall glossaries: replaced \ifthenelse with \ifx	55
\glssseeformat: new	190	\forall glossentries: replaced \ifthenelse with \ifx	55
\glssSetSuffixF: new	41	\glssdefmain: new	16
\glssSetSuffixFF: new	41	\glssdescwidth: changed \linewidth to \hsize	277, 299
\ifglssxindy: new	29	\glsslistdottedwidth: changed \linewidth to \hsize	277
\istfilename: added xindy support ...	40	\glsspagelistwidth: changed \linewidth to \hsize	277, 299
\newglossarystyle: made \newglossarystyle long	216	nomain: added nomain package option	16
\nopostdesc: new	39	\writeist: removed item_02 - no such makeindex key	167
nonumberlist: new	68	2.02 (2007-07-13) \@printglossary: suppressed warning globally rather than locally	194
\printglossary: added check to determine if \printglossary is already defined	191	2.02 (2009-07-13) \glossarysection: changed \@mkboth to \glossarymark	43
added print language to aux file	191	\glssglossarymark: New	43
order: order package option added	29	2.03 (2009-09-23) \@GLS@: Added check for hyperfirst	127
\writeist: added xindy support	163	\@GLSpl: Added check for hyperfirst	130
1.18 (2009-01-14) \@glsl@loadlist: new	10	\@GLs@: Added check for hyperfirst	127
\@glsl@loadlong: new	9	\@GLspl@: Added check for hyperfirst	129
\@glsl@loadsuper: new	10	\@glsl@: Added check for hyperfirst	126
\@glsl@loadtree: new	10	\@glsl@@link: new	113
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort	84	\@glsl@link: added \leavevmode	114
moved sort sanitization to \newglossaryentry	88	Moved entry existence check to avoid duplicate code	114
\glstarget: new	210	\@glsl@disp: Added check for hyperfirst	131
\oldacronym: new	219	\@glspl@: Added check for hyperfirst	128
nolist: new	10	\glssglossarymark: Added check to see if it's already defined	43
nolong: new	10	hyperfirst: new	27
sort: moved sanitization to \newglossaryentry	66		
nostyles: new	10		
nosuper: new	10		
notree: new	10		
1.19 (2009-03-02) \glsclearpage: new	45		
\glssdisp: new	130		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	246		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	127
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms	130
\@GLs@: Changed test to check if glossary type has been identified as a list of acronyms	127
\@GLspl@: Changed test to check if glossary type has been identified as a list of acronyms	129
\@glossaryentryfield: new	90
\@glossarysubentryfield: new	90
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	126
\@glsacronymlists: new	18
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	131
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	128
\@newglossaryentryposthook: new	89
\@newglossaryentryprehook: new	89
acronymlists: new	19
\DeclareAcronymList: new	18
\DefineAcronymSynonyms: new	236
\gls@defglossaryentry: added user1-6 keys	84
\glsadd: fixed bug that ignored counter	160
\Glsentryuseri: new	155
\Glsentryuseri: new	155
\Glsentryuserii: new	156
\Glsentryuserii: new	156
\Glsentryuseriii: new	156
\Glsentryuseriii: new	156
\Glsentryuseriv: new	156
\Glsentryuseriv: new	156
\Glsentryuseriv: new	156
\Glsentryuseriv: new	156
\Glsentryuseri: new	157
\Glsentryuseri: new	157
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	63
\SetAcronymLists: new	19
\SetDefaultAcronymDisplayStyle: new	238
\SetDefaultAcronymStyle: new	239
\SetDescriptionAcronymDisplayStyle: new	244
\SetDescriptionDUAAcronymDisplayStyle: new	242
\SetDescriptionFootnoteAcronymDisplayStyle: new	240
\SetDUADisplayStyle: new	251
\SetFootnoteAcronymDisplayStyle: new	246
\SetSmallAcronymDisplayStyle: new	249
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	130
Removed spurious brace. Patch provided by Sergiu Dotenco	131
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	168
2.06 (2010-06-14)	
\altnewglossary: new	64
\CustomAcronymFields: new	254
\CustomNewAcronymDef: new	254
\SetCustomDisplayStyle: new	254
\SetCustomStyle: new	254
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	160
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites	172
\gls@wrglossary: modified to take into account savewrites	182
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument	116
3.0 (2011-04-02)	
\@do@esc@wrglossary: added check for hyper location prefix	187
modified to use new format	185
\@@glossarysec: replaced \@ifundefined with \ifcsundef	7
\@do@seeglossary: Sanitize and escape cross-referencing information	189
\@gls@counterwithin: new	12

\@gls@ifinlist: new	46	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter	160
\@gls@saveentrycounter	115	\GlsAddXdyCounters: new	47
added \@gls@setsort	115	\glseentrycounterlabel: new	209
\@gls@saveentrycounter: new	115	\glseentryitem: new	209
\@gls@setupsort@def: new	14	\glseentrylong: new	157
\@gls@setupsort@standard: new	13	\glseentrylong: new	157
\@gls@setupsort@use: new	14	\glseentrylongpl: new	157
\@gls@xdy@locationlist: new	49	\glseentrylongpl: new	157
\@glslink: replaced \@ifundefined		\glseentryshort: new	157
with \ifcsundef	124	\glseentryshort: new	157
\@glsnextpages: new	207	\glseentryshortpl: new	157
\@print@glossary: replaced		\glseentryshortpl: new	157
\@ifundefined with \ifcsundef ..	195	\glsgetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	213
\currentglossary	193	\gls glossarymark: replaced	
added \glsnextpages	193	\@ifundefined with \ifcsundef ..	43
make toctitle default to title	193	\glshyperlink: changed default from	
\@xdyattributelist: new	46	\glseentryname to \glseentrytext ..	159
General: added prefix to hyperlink	218	\glshypernumber: replaced	
etoolbox now loaded	4	\@ifundefined with \ifcsundef ..	217
replaced \@ifundefined with		\glsnumberformat: replaced	
\ifcsundef	35, 38, 111, 205	\@ifundefined with \ifcsundef ..	41
\acrfootnote: new	240	\glsrefentry: new	209
\ACRfull: added starred version	222	\glsresetsubentrycounter: new ...	208
\Acrfull: added starred version	221	\glsseeitem: hyperlink uses	
\acrfull: added starred version	220	\glsseeitemformat instead of	
\ACRfullpl: added starred version ...	223	\glseentryname	191
\Acrfullpl: added starred version ...	222	\glsseeitemformat: new	191
\acrfullpl: added starred version ...	222	\glssortnumberfmt: new	14
\acrlinkfootnote: new	240	\glsstepentry: new	208
\acrno linkfootnote: new	240	\glsstepsubentry: new	208
savewrites: new	30	\glssubentrycounterlabel: new ...	209
see: added \@glo@seeautonumberlist ..	68	\glssubentryitem: new	209
seeautonumberlist: new	9	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef	209
\@ifundefined with \ifcsundef ..	43	short: new	70
\glossarystyle: replaced		shortplural: new	70
\@ifundefined with \ifcsundef ..	215	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	56
\@ifundefined with \ifcsundef ..	29	\ifglseentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	56
\@gls@defsort	88	\istfile: deprecated	180
added short and long keys	85	glossaryentry: new	11
replaced \@ifundefined with		glossarysubentry: new	12
\ifcsundef	85	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	45	\newrobustcmd	73

<code>\newglossarystyle</code> : replaced		<code>\showglouseriii</code> : new	257
<code>\@ifundefined</code> with <code>\ifcsundef</code>	216	<code>\showglouseriv</code> : new	257
<code>\ns@newglossary</code> : added		<code>\showglouserv</code> : new	258
<code>\@gls@defsortcount</code>	64	<code>\showglouservi</code> : new	258
replaced <code>\@ifundefined</code> with		<code>subentrycounter</code> : new	12
<code>\ifcsundef</code>	63	<code>\writeist</code> : added xindy-only macro	
<code>entrycounter</code> : new	12	definitions to glossary open tag	165
<code>\oldacronym</code> : replaced <code>\@ifundefined</code>		modified to support new format	163
with <code>\ifcsundef</code>	219	3.01 (2011-04-12)	
<code>compatible-2.07</code> : <code>compatible-2.07</code>		<code>\@glswritefiles</code> : added check for	
option added	31	empty glossaries	180
<code>long</code> : new	70	General: made robust	127
<code>longplural</code> : new	70	<code>\ACRfull</code> : made robust	222
<code>nonumberlist</code> : now boolean	68	<code>\Acrfull</code> : made robust	221
<code>sort</code> : new	13	<code>\acrfull</code> : made robust	220
<code>counter</code> : replaced <code>\@ifundefined</code> with		<code>\acrfullformat</code> : removed	
<code>\ifcsundef</code>	67	<code>\acronymfont</code> as it should already be	
<code>counterwithin</code> : new	12	set in the second argument.	221
<code>\printglossary</code> : replaced		<code>\ACRfullpl</code> : made robust	223
<code>\@ifundefined</code> with <code>\ifcsundef</code>	191	<code>\Acrfullpl</code> : made robust	222
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>		<code>\acrfullpl</code> : made robust	222
expanded options link options	240	<code>\ACRlong</code> : made robust	148
<code>\setentrycounter</code> : added optional		<code>\Acrlong</code> : made robust	148
argument	214	<code>\acrlong</code> : made robust	147
<code>\showacronymlists</code> : new	260	<code>\ACRlongpl</code> : made robust	150
<code>\showglocounter</code> : new	257	<code>\Acrlongpl</code> : made robust	150
<code>\showgloDESC</code> : new	258	<code>\acrlongpl</code> : made robust	149
<code>\showgloDESCplural</code> : new	258	<code>\ACRshort</code> : made robust	145
<code>\showglofirst</code> : new	256	<code>\Acrshort</code> : made robust	144
<code>\showglofirstpl</code> : new	256	<code>\acrshort</code> : made robust	143
<code>\showgloflag</code> : new	259	<code>\ACRshortpl</code> : made robust	146
<code>\showgloindex</code> : new	259	<code>\Acrshortpl</code> : made robust	146
<code>\showglolevel</code> : new	256	<code>\acrshortpl</code> : made robust	145
<code>\showglongame</code> : new	258	<code>\Gls</code> : made robust	126
<code>\showgloparent</code> : new	256	<code>\glsadd</code> : made robust	160
<code>\showgloplural</code> : new	256	<code>\glsaddall</code> : made robust	160
<code>\showglosort</code> : new	258	<code>\GLSdesc</code> : made robust	136
<code>\showglossaries</code> : new	260	<code>\Glsdesc</code> : made robust	136
<code>\showglossarycounter</code> : new	261	<code>\glsdesc</code> : made robust	135
<code>\showglossaryentries</code> : new	261	<code>\GLSdescplural</code> : made robust	137
<code>\showglossaryin</code> : new	260	<code>\Glsdescplural</code> : made robust	136
<code>\showglossaryout</code> : new	260	<code>\glsdescplural</code> : made robust	136
<code>\showglossarytitle</code> : new	261	<code>\glsfirst</code> : made robust	132
<code>\showglosymbol</code> : new	259	<code>\GLSfirstplural</code> : made robust	134
<code>\showglosymbolplural</code> : new	259	<code>\Glsfirstplural</code> : made robust	134
<code>\showglotext</code> : new	256	<code>\glsfirstplural</code> : made robust	134
<code>\showglotype</code> : new	257	<code>\glslink</code> : made robust	113
<code>\showglouserI</code> : new	257	<code>\GLSname</code> : made robust	135
<code>\showglouserII</code> : new	257	<code>\Glsname</code> : made robust	135

\glsname: made robust	135	\@printglossary: add a way to fetch	
\GLSpl: made robust	129	current entry label	194
\Glspl: made robust	129	savenumberlist: new	9
\glspl: made robust	128	ucmark: new	11
\GLSplural: made robust	133	\gls@defglossaryentry: added	
\GLSsymbol: made robust	138	numberlist element	88
\Glsymbol: made robust	137	\gls@save@numberlist: new	191
\glssymbol: made robust	137	\gls@wrglossary: added check for	
\GLSsymbolplural: made robust	138	glossary file defined	182
\Glsymbolplural: made robust	138	\glsdisplaynumberlist: new	158
\glssymbolplural: made robust	138	\glsentrycounter: set default value ..	115
\Glstext: made robust	132	\Glsentryfull: fixed bug (replaced	
\glstext: made robust	131	\glsentryshortpl with	
\GLSuseri: made robust	139	\glsentryshort)	158
\Glsuseri: made robust	139	\glsentryfullpl: fixed bug (replaced	
\glsuseri: made robust	139	\glsentryshort with	
\GLSuserii: made robust	140	\glsentryshortpl)	158
\Glsuserii: made robust	140	\glsentrynumberlist: new	158
\glsuserii: made robust	139	\glsmoveentry: new	90
\GLSuseriii: made robust	141	\glsresetsubentrycounter: new ...	208
\Glsuseriii: made robust	141	\ifglshaschildren: new	58
\glsuseriii: made robust	140	\ifglshasparent: new	58
\GLSuseriv: made robust	142	\makeglossaries: added list parser ..	176
\Glsuseriv: made robust	141	indexonlyfirst: new	27
\glsuseriv: made robust	141	\renewglossarystyle: new	216
\GLSuserv: made robust	142	\showglossaryentries: fixed misspelt	
\Glsuserv: made robust	142	command	261
\glsuserv: made robust	142	\SmallNewAcronymDef: fixed broken	
\GLSuservi: made robust	143	short and long plural	249
\Glsuservi: made robust	143	3.03 (2012/09/21)	
\glsuservi: made robust	143	\@gls@sanitizesort: new	22
3.02 (2012-05-19)		\@gls@setupsort@standard: used	
\glsnumlistlastsep: new	159	\@gls@sanitizesort	13
\glsnumlistsep: new	159	\@printglossary: allow title to override	
3.02 (2012-05-21)		default toctitle	192
\@do@wrglossary: changed		General: allow title to set toctitle	205
\@glslocref to		\glsinlinedescformat: new	273
\theglsentrycounter	188	\glsinlineemptydescformat: new ..	273
\@do@wrglossary: changed		\glsinlinenameformat: new	273
\@do@wr@glossary to test for		\glsinlinepostchild: new	273
indexonlyfirst option; put old		\glsinlinesubdescformat: new	273
\@do@wr@glossary code into		\glsinlinesubnameformat: new	273
\@do@wrglossary	183	\glspostinline: replaced "." with	
\@gls@missingnumberlist: new	71	\glspostdescription	273
\@glswritefiles: added check for		list: added check for glsnogroupskip ..	275
existence of token in case		altlongragged4col: added check for	
\makeglossaries has been		glsnogroupskip	292
omitted	180	altsuperragged4col: added check for	
		glsnogroupskip	311

alttree: added check for		\gls@disablepagerefexpansion: new	183
glsnogroupskip	320	\gls@numberpage: new	183
index: added check for glsnogroupskip	314	\gls@protected@pagefmts: new	183
nogroupskip: new	11	\gls@romanpage: new	184
long: added check for glsnogroupskip	278	\glsdefmain: added check for doc	
long3col: added check for		package	16
glsnogroupskip	280	\glsorg@endtheglossary: new	5
long4col: added check for		\glsorg@theglossary: new	5
glsnogroupskip	281	\PrintChanges: new	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip	289	\@do@esc@wrglossary: add Roman	
longragged3col: added check for		case. Fixed bugs in the else	
glsnogroupskip	290	statements	186
nopostdot: new	11	\@gls@link: added check for	
tree: added check for glsnogroupskip	315	“nohypertypes”	114
treenoname: added check for		mcolalttree: replaced ‘2’ with	
glsnogroupskip	317	\glsmcols	298
super: added check for glsnogroupskip	300	mcolindex: replaced ‘2’ with \glsmcols	294
super3col: added check for		mcolindexspannav: replaced ‘2’ with	
glsnogroupskip	302	\glsmcols	295
super4col: added check for		mcoltree: replaced ‘2’ with \glsmcols	295
glsnogroupskip	304	mcoltreenoname: replaced ‘2’ with	
superragged: added check for		\glsmcols	297
glsnogroupskip	307	mcoltreespannav: replaced ‘2’ with	
superragged3col: added check for		\glsmcols	296
glsnogroupskip	309	\gls@protected@pagefmts: added	
3.04 (2012-11-11)		Roman to list	183
altlist: replaced \newline with		\gls@Romanpage: new	184
paragraph break	275	\glsgetgrouplabel: fixed bug (typo in	
3.04 (2012-11-18)		\equal)	214
\@do@wrglossary: changed		\nopostdesc: made robust	39
\theglsentrycounter back to		3.05 (2013/04/21)	
\@glslocref	188	\@gls@nohyperlist: new	19
\@do@esc@wrglossary: modified to		\GlsDeclareNoHyperList: new	19
compensate for possible incorrect		nohypertypes: new	19
page number	186	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	53
\gls@romanpage	117	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	114
unwanted whatsit occurring here. . .	195	\glossarypreamble: modified to work	
General: Added check for doc package	4	with \setglossarypreamble	42
added datatool-base as a required		\gls@docclearpage: added check for	
package	4	openright	45
added local key	111	\glspostdescription: Added	
\gls@Alphpage: new	183	spacefactor code	11
\gls@alphpage: new	183		

\GlsSetXdyCodePage: Added check for fontspec	54	\glseelist: made robust	190
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	244	\ifglshasdesc: new	59
\setglossarypreamble: new	42	\ifglshassymbol: new	59
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	292
list: updated list style to use \glossentry and \subglossentry	274	alttree: updated to use \glossentry and \subglossentry	319
listdotted: updated listdotted style to use \glossentry and \subglossentry	276	index: added paragraph break at end of environment	313
altlist: updated altlist style to use \glossentry and \subglossentry	275	updated to use \glossentry and \subglossentry	313
inline: updated inline style to use \glossentry and \subglossentry	272	long: updated to use \glossentry and \subglossentry	278
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	289
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	91	longragged3col: updated to use \glossentry and \subglossentry	290
updated for \glossentry	91	tree: updated to use \glossentry and \subglossentry	315
\@glossaryentryfield: switched to \glossentry	90	\setglossarystyle: new	215
\@glossarysubentryfield: switched to \subglossentry	90	\setglossentrycompatibility: new	212
General: added nogroupskip key to \printglossary	206	superragged: updated to use \glossentry and \subglossentry	307
removed definition of \@glossaryentryfield	362	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield	362	\@gls@assign@symbolplural@field: new	22
\compatibleglossentry: new	210	\@gls@default@value: new	67
\compatiblesubglossentry: new ...	211	\Glsentrydesc: made robust	153
\glossaryentryfield: deprecated ...	212	\Glsentrydescplural: made robust ..	153
\Glossentrydesc: new	211	\Glsentryfirst: made robust	154
\glossentrydesc: new	211	\Glsentryfirstplural: made robust .	155
\Glossentryname: new	211	\Glsentryfull: made robust	158
\glossentryname: new	210	\Glsentryfullpl: made robust	158
\Glossentrysymbol: new	211	\Glsentrylong: made robust	157
\glossentrysymbol: new	211	\Glsentrylongpl: made robust	157
\gls@assign@desc@field: new	21	\Glsentryname: made robust	152
\gls@assign@descplural@field: new	21	\Glsentryplural: made robust	154
\gls@assign@field: new	73	\Glsentryshort: made robust	157
\gls@ifnotmeasuring: new	92	\Glsentryshortpl: made robust	157
\glsaddallunused: new	161	\Glsentrysymbol: made robust	154
\glsexpandfields: new	73	\Glsentrysymbolplural: made robust	154
\glsnoexpandfields: new	73	\Glsentrytext: made robust	153
\glssee: made robust	190	\Glsentryuseri: made robust	155
\glsseeformat: made robust	190	\Glsentryuserii: made robust	156
\glsseeitem: made robust	191	\Glsentryuseriii: made robust	156
		\Glsentryuseriv: made robust	156
		\Glsentryuserv: made robust	156
		\Glsentryuservi: made robust	157

<code>\glstextup: new</code>	220	<code>\@Gls@: add \glsifplural,</code>	
<code>\ifglshassymbol: changed test to check</code>		<code>\glscapscase, \glscustomtext and</code>	
<code>for \@gls@default@symbol</code>	59	<code>\glsinsert</code>	127
3.10a (2013-09-28)		change to using <code>\glentryfmt</code> style	
<code>\gls@assign@type@field: new</code>	21	commands	127
3.10a (2013-10-13)		removed <code>\makefirstuc</code> (now dealt	
<code>\@gls@keymap: new</code>	75	with in <code>\glentryfmt</code>)	127
<code>\@gls@provide@newglossary: new</code> ...	62	<code>\@Glspl@: add \glsifplural,</code>	
<code>\@gls@writedef: new</code>	75	<code>\glscapscase, \glscustomtext and</code>	
<code>\@glsdefaultplural: Obsolete</code>	71	<code>\glsinsert</code>	129
<code>\@glsnodesc: new</code>	71	change to using <code>\glentryfmt</code> style	
<code>\@print@glossary: Added</code>		commands	129
<code>providecommand code to aux file</code> ..	195	removed <code>\makefirstuc</code> (now dealt	
<code>\gls@defglossaryentry: Changed to</code>		with in <code>\glentryfmt</code>)	129
<code>using \@gls@default@value</code>	84	<code>\@acrlong: added \glslabel,</code>	
<code>new</code>	83	<code>\glsifplural, \glscapscase,</code>	
<code>\glswritedefhook: new</code>	83	<code>\glsinsert and \glscustomtext</code>	361
<code>\makeglossaries: Added</code>		<code>\@acrshort: added \glslabel,</code>	
<code>providecommand code to aux file</code> ..	174	<code>\glsifplural, \glscapscase,</code>	
<code>\new@glossaryentry: new</code>	74	<code>\glsinsert and \glscustomtext</code>	360
<code>\ns@newglossary: added</code>		<code>\@gls@: add \glslabel, \glsifplural,</code>	
<code>\@gls@provide@newglossary</code>	63	<code>\glscapscase, \glscustomtext and</code>	
3.11a (2013-10-15)		<code>\glsinsert</code>	126
<code>\@ACRlong: added \glslabel,</code>		change to using <code>\glentryfmt</code> style	
<code>\glsifplural, \glscapscase,</code>		commands	126
<code>\glsinsert and \glscustomtext</code>	362	<code>\@gls@noexpand@fields: Fixed bug</code>	
<code>\@ACRshort: added \glslabel,</code>		<code>expand replaced with noexpand</code>	72
<code>\glsifplural, \glscapscase,</code>		<code>\@glsdisp: add \glslabel,</code>	
<code>\glsinsert and \glscustomtext</code>	360	<code>\glsifplural, \glscapscase,</code>	
<code>\@Acrlong: added \glslabel,</code>		<code>\glscustomtext and \glsinsert</code>	130
<code>\glsifplural, \glscapscase,</code>		change to using <code>\glentryfmt</code> style	
<code>\glsinsert and \glscustomtext</code>	361	commands	131
<code>\@Acrshort: added \glslabel,</code>		<code>\@glspl@: add \glslabel,</code>	
<code>\glsifplural, \glscapscase,</code>		<code>\glsifplural, \glscapscase,</code>	
<code>\glsinsert and \glscustomtext</code>	360	<code>\glscustomtext and \glsinsert</code>	128
<code>\@GLS@: add \glslabel, \glsifplural,</code>		change to using <code>\glentryfmt</code> style	
<code>\glscapscase, \glscustomtext and</code>		commands	128
<code>\glsinsert</code>	127	General: added <code>\glslabel,</code>	
change to using <code>\glentryfmt</code> style		<code>\glsifplural, \glscapscase,</code>	
commands	127	<code>\glsinsert and</code>	
removed <code>\MakeUppercase</code> (now		<code>\glscustomtext</code>	144–150
moved to <code>\glentryfmt</code>)	127	changed to just use	
<code>\@GLSpl: add \glslabel,</code>		<code>\Glsentrydescplural</code>	137
<code>\glsifplural, \glscapscase,</code>		changed to just use	
<code>\glscustomtext and \glsinsert</code>	130	<code>\glentrydescplural</code>	136, 137
change to using <code>\glentryfmt</code> style		changed to just use <code>\Glsentrydesc</code> .	136
commands	130	changed to just use <code>\glentrydesc</code> .	136
removed <code>\MakeUppercase</code> as now		changed to just use	
dealt with in <code>\glentryfmt</code>	130	<code>\Glsentryfirstplural</code>	134

changed to just use		
\glentryfirstplural	134	
changed to just use \Glentryfirst	133	
changed to just use		
\glentryfirst	132, 133	
changed to just use \Glentryname .	135	
changed to just use \glentryname .	135	
changed to just use \Glentryplural	133	
changed to just use		
\glentryplural	133, 134	
changed to just use		
\Glentrysymbolplural	138	
changed to just use		
\glentrysymbolplural	138, 139	
changed to just use \Glentrysymbol	137	
changed to just use		
\glentrysymbol	137, 138	
Changed to just use \Glentrytext .	132	
changed to just use \glentrytext .	131	
changed to just use		
\Glentryuseriii	141	
changed to just use		
\glentryuseriii	140, 141	
changed to just use \Glentryuserii	140	
changed to just use \glentryuserii	140	
changed to just use \Glentryuseriv	141	
changed to just use		
\glentryuseriv	141, 142	
changed to just use \Glentryuseri	139	
changed to just use \glentryuseri	139	
changed to just use \Glentryuservi	143	
changed to just use \glentryuservi	143	
changed to just use \Glentryuserv	142	
changed to just use		
\glentryuserv	142, 143	
Now requires textcase	4	
acronymlists: replaced		
\@addtoacronymlists with		
\DeclareAcronymList	19	
\defgldisplay: obsoleted	110	
\defgldisplayfirst: obsoleted	110	
\defglentryfmt: new	62	
\forglentries: replaced \ifx with		
\ifdefempty	55	
\gls@assign@desc: new	83	
\gls@defglossaryentry: Fixed default		
counter if none supplied	87	
\gls@doentryfmt: new	62	
\gldisplay: obsoleted	109	
\gldisplayfirst: obsoleted	109	
\glsgenentryfmt: new	104	
\glsgetgrouptitle: Added check in		
case non-Latin alphabet in use	213	
\glsglossarymark: replaced		
\MakeUppercase with		
\mfirstucMakeUppercase	43	
\glsnavigation: switched to using		
\@gls@getgrouptitle	271	
\ifglshasdesc: replaced \ifdefempty		
with \ifcempty	59	
\ifglshaslong: new	59	
\ifglshasshort: new	59	
\ifglshassymbol: replaced		
\ifdefempty with \ifcempty	59	
\ifglused: replaced \ifthenelse with		
\ifbool	56	
\longnewglossaryentry: new	83	
\ns@newglossary: replaced		
\gldisplay and		
\gldisplayfirst with		
\glentryfmt	63	
compatible-3.07: cnew	31	
\SetCustomDisplayStyle: updated to		
use \defglentryfmt	254	
\SetDefaultAcronymDisplayStyle:		
changed to use \defglentryfmt .	238	
\SetDescriptionAcronymDisplayStyle:		
updated to use \defglentryfmt .	244	
\SetDescriptionDUAAcronymDisplayStyle:		
updated to use \defglentryfmt .	242	
\SetDescriptionFootnoteAcronymDisplayStyle:		
updated to use \defglentryfmt .	240	
\SetDUADisplayStyle: updated to use		
\defglentryfmt	251	
\SetFootnoteAcronymDisplayStyle:		
updated to use \defglentryfmt .	246	
\SetSmallAcronymDisplayStyle:		
updated to use \defglentryfmt .	249	
\setupglossaries: new	34	
\showglolong: new	259	
\showgloshort: new	259	
numbers: new	33	
symbols: new	32	
3.12a (2013-10-16)		
\gls@defglossaryentry: added		
\glslabel	84	
\glsaddkey: new	77	

3.13a (2013-11-05)		long: switched to \tabularnewline ..	278
\@gls@assign@symbol@field: changed		long3col: switched to	
to use \glsetnoexpandfield	22	\tabularnewline	279
\@gls@assign@symbolplural@field:		long3colheader: switched to	
changed to use		\tabularnewline	280
\glsetnoexpandfield	22	long3colheaderborder: switched to	
\@gls@link: removed \relax	115	\tabularnewline	280
\@gls@notranslatorhook: new	25	long4col: switched to	
\@gls@setupsort@standard: moved		\tabularnewline	281
\@gls@santizesort to		long4colheader: switched to	
\glsprestandardsort	13	\tabularnewline	281
ucmark: added check for memoir	11	longheader: switched to	
see: added \gls@checkseeallowed ...	68	\tabularnewline	279
\glossarysection: changed		longheaderborder: switched to	
\glossarymark to		\tabularnewline	279
\glsglossarymark	43	\SetFootnoteAcronymDisplayStyle:	
\glossarystyle: fixed bug caused by		fixed missing argument bug	247
using \ifdef instead of \ifcsdef .	215	super: switched to \tabularnewline .	300
\gls@assign@desc@field: changed to		super3col: switched to	
use \glsetnoexpandfield	21	\tabularnewline	302
\gls@assign@descplural@field:		super3colheader: switched to	
changed to use		\tabularnewline	302
\glsetnoexpandfield	21	super4col: switched to	
\gls@assign@name@field: changed to		\tabularnewline	303
use \glsetnoexpandfield	22	super4colheader: switched to	
\gls@assign@type@field: changed to		\tabularnewline	304
use \glsetexpandfield	21	super4colheaderborder: switched to	
\gls@checkseeallowed: new	68	\tabularnewline	304
\glsaddallunused: set default to		superheader: switched to	
\@glo@types	161	\tabularnewline	301
\Glsentryfull: changed to use		superheaderborder: switched to	
\acrfullformat	158	\tabularnewline	301
\glentryfull: changed to use			
\acrfullformat	158	3.14a (2013-11-12)	
\Glsentryfullpl: changed to use		\@glswritefiles: renamed	
\acrfullformat	158	\glswritefiles to	
\glentryfullpl: changed to use		\@glswritefiles and used	
\acrfullformat	158	“savewrites” option to set	
\glsglossarymark: renamed		\glswritefiles	180
\glsglossarymark to		General: new	262
\glsglossarymark to avoid conflict		acronyms: new	17
with memoir	43	\gls@defglossaryentry: added check	
\glsprestandardsort: new	13	for existence of default glossary	85
\glsetexpandfield: new	21	set the default for firstplural to be the	
\glsetnoexpandfield: new	21	value of plural	87
altsuper4colheader: switched to		xindygloss: new	30
\tabularnewline	305	\longprovideglossaryentry: new ...	83
altsuper4colheaderborder: switched		compatible-2.07: added check for 2.07	
to \tabularnewline	306	before setting 3.07 compatibility	31
		notranslate: new	26

\provideglossaryentry:new	74	index:new	33
4.0 (2013-11-14)		\newacronymstyle:new	226
\gls@defglossaryentry: added check		long-sc-short:new	229
for first key	87	long-sc-short-desc:new	230
super: fixed typo in \subglossentry		long-short:new	227
(\glossentrydesc)	300	long-short-desc:new	230
4.01 (2013-11-16)		long-sm-short:new	229
General: fixed non-value options so that		long-sm-short-desc:new	230
they can be passed to document class .	8	long-sp-short-desc:new	230
\CustomAcronymFields: inserted		footnote:new	234
missing comma	254	footnote-desc:new	236
4.02 (2013-12-05)		footnote-sc:new	235
\@acrfull: now using \acrfullfmt ..	221	footnote-sc-desc:new	236
\@gls@indexdef:new	33	footnote-sm:new	235
\@gls@numbersdef:new	33	footnote-sm-desc:new	236
\@gls@symbolsdef:new	33	\setacronymstyle:new	225
General: Removed \acronymfont .	147–151	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new	222	Moved check for empty custom text to	
\Acrfullfmt:new	221	prevent unwanted parenthetical	
\acrfullfmt:new	221	material	244
\ACRfullplfmt:new	223	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new	223	Moved check for empty custom text to	
\acrfullplfmt:new	222	prevent unwanted parenthetical	
\acronymentry:new	225	material	240
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here	25	Moved check for empty custom text to	
sc-short-long:new	229	prevent unwanted parenthetical	
sc-short-long-desc:new	231	material	246
\Genacrfullformat:new	109	\SetGenericNewAcronym:new	224
\genacrfullformat:new	109	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new	225	Moved check for empty custom text to	
\Genplacrfullformat:new	109	prevent unwanted parenthetical	
\genplacrfullformat:new	109	material	249
\Glsentryfull: bug fix: added missing		dua:new	232
\acronymfont	158	dua-desc:new	233
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont	158	option	8
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont	158	\makeglossaries: made preamble only	176
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont	158	General: changed default to \@empty	
\glsngenacfmt:new	107	instead of \relax	30
\GlsUseAcrEntryDisplayStyle:new ...	226	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new	226	\@do@esc@wrglossary: added	
short-long:new	228	\glsdetoklabel	187
short-long-desc:new	231	\@do@noesc@wrglossary: added	
xindynoglsnumbers:new	30	\glsdetoklabel	185
sm-short-long:new	229	\@ACRlong: removed \glslabel	
sm-short-long-desc:new	231	(defined in \@gls@link)	362

\@ACRshort: removed \glslabel (defined in \@gls@link)	360	\Genplacrfullformat: redefined to use accessibility information	359
\@Acrlong: removed \glslabel (defined in \@gls@link)	361	\genplacrfullformat: redefined to use accessibility information	359
\@Acrshort: removed \glslabel (defined in \@gls@link)	360	\glossentryname: added \glsdetoklabel	210
\@GLS@: removed \glslabel (defined in \@gls@link)	127	\gls@defglossaryentry: added \glsdetoklabel	83
\@GLSpl: removed \glslabel (defined in \@gls@link)	130	replaced #1 with \@glo@label	85
\@Gls@: removed \glslabel (defined in \@gls@link)	127	replaced \ifthenelse with \ifdefequal	86
\@Gls@entry@field: new	151	\glsadd: added \glsdetoklabel	160
\@Glspl@: removed \glslabel (defined in \@gls@link)	129	\glsaddkey: switched to using \@gls@field@link	78
\@acrlong: removed \glslabel (defined in \@gls@link)	361	\glsdetoklabel: new	56
\@acrshort: removed \glslabel (defined in \@gls@link)	360	\glsdisplaynumberlist: added \glsdetoklabel	159
\@gls@: removed \glslabel (defined in \@gls@link)	126	\glsdoifexistsorwarn: new	57
\@gls@access@display: new	348	\glsentryaccess: switched to using \@gls@entry@field	346
\@gls@entry@field: new	151	\glsentrydescaccess: switched to using \@gls@entry@field	347
\@gls@fetchfield: new	76	\glsentrydescpluralaccess: switched to using \@gls@entry@field	347
\@gls@field@link: new	131	\glsentryfirstaccess: switched to using \@gls@entry@field	347
\@gls@link: added \glsdetoklabel .	114	\glsentryfirstplural: added \glsdetoklabel	155
moved \@gls@link@opts and \@gls@link@label to \@gls@link	114	\glsentrylongaccess: switched to using \@gls@entry@field	348
\@gls@writedef: added \glsdetoklabel	75	\glsentrylongpluralaccess: switched to using \@gls@entry@field	348
\@glsdisp: removed \glslabel (defined in \@gls@link)	130	\glsentrypluralaccess: switched to using \@gls@entry@field	347
\@Glspl@: removed \glslabel (defined in \@gls@link)	128	\glsentryshortaccess: switched to using \@gls@entry@field	347
\@printglossary: added \glsdetoklabel	194	\glsentryshortpluralaccess: switched to using \@gls@entry@field	347
General: removed \glslabel (defined in \@gls@link)	144	\glsentrysymbolaccess: switched to using \@gls@entry@field	347
sc-short-long-desc: redefined to use accessibility information	366	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field	347
\compatibleglossentry: added \glsdetoklabel	342	\glsentrytextaccess: switched to using \@gls@entry@field	347
\compatiblesubglossentry: added \glsdetoklabel	343	\glsngenacfmt: redefined to use accessibility information	357
\Genacrfullformat: redefined to use accessibility information	359		
\genacrfullformat: redefined to use accessibility information	359		

\glsgenentryfmt: redefined to use accessibility information	354	sm-short-long-desc: redefined to use accessibility information	366
\glshyperlink: added \glsdetoklabel	159	long-sc-short-desc: redefined to use accessibility information	365
\glslocalreset: added \glsdetoklabel	92	long-short: redefined to use accessibility information	363
\glslocalunset: added \glsdetoklabel	93	long-short-desc: redefined to use accessibility information	365
\glsmoveentry: added \glsdetoklabel	90	long-sm-short-desc: redefined to use accessibility information	365
replaced \ifthenelse with \ifdefequal	90	footnote: redefined to use accessibility information	369
\glsrefentry: added \glsdetoklabel	209	footnote-desc: redefined to use accessibility information	371
\glsreset: added \glsdetoklabel ...	92	footnote-sc: redefined to use accessibility information	371
\glsseelist: added \expandafter commands	190	footnote-sc-desc: redefined to use accessibility information	372
\glsstepentry: added \glsdetoklabel	208	footnote-sm: redefined to use accessibility information	371
\glsstepsubentry: added \glsdetoklabel	208	footnote-sm-desc: redefined to use accessibility information	372
\glsunset: added \glsdetoklabel ...	93	\renewacronymstyle: new	226
short-long: commented spurious EOL	228	\showglocounter: added \glsdetoklabel	257
redefined to use accessibility information	364	\showglodesc: added \glsdetoklabel	258
short-long-desc: redefined to use accessibility information	366	\showglodescaccess: added \glsdetoklabel	378
\ifglshdescsuppressed: added \glsdetoklabel	59	\showglodescplural: added \glsdetoklabel	258
fixed typo	59	\showglodescpluralaccess: added \glsdetoklabel	378
\ifglshentryexists: added \glsdetoklabel	56	\showglofirst: added \glsdetoklabel	256
\ifglshaschildren: added \glsdetoklabel	58	\showglofirstaccess: added \glsdetoklabel	378
\ifglshasdesc: added \glsdetoklabel	59	\showglofirstpl: added \glsdetoklabel	256
\ifglshasfield: new	60	\showglofirstpluralaccess: added \glsdetoklabel	378
\ifglshaslong: added \glsdetoklabel	59	\showgloflag: added \glsdetoklabel	259
\ifglshasparent: added \glsdetoklabel	58	\showgloindex: added \glsdetoklabel	259
\ifglshasshort: added \glsdetoklabel	59	\showglolevel: added \glsdetoklabel	256
\ifglshassymbol: added \glsdetoklabel	59	\showglolong: added \glsdetoklabel	259
replaced \ifcempty with \ifdefempty and replaced \ifx with \ifdefequal	59	\showglolongaccess: added \glsdetoklabel	379
\ifglshused: added \glsdetoklabel ..	56		

\showglolongpluralaccess: added		redefined to use accessibility	
\glsdetoklabel	379	information	369
\showglongname: added \glsdetoklabel	258	4.04 (2014-03-04)	
\showglongnameaccess: added		\@gls@getcounterprefix: added	
\glsdetoklabel	378	warning if no prefix can be formed .	189
\showgloparent: added		4.04 (2014-03-06)	
\glsdetoklabel	256	\@@gls@noidx@nosanitizesort: new .	23
\showgloplural: added		\@@gls@noidx@sanitizesort: new ...	22
\glsdetoklabel	256	\@@gls@nosanitizesort: new	22
\showglopluralaccess: added		\@@gls@sanitizesort: new	22
\glsdetoklabel	378	\@glo@addchildren: new	196
\showgloshort: added		\@glo@do@sortentries: new	197
\glsdetoklabel	259	\@glo@grabfirst: new	202
\showgloshortaccess: added		\@glo@sortedinsert: new	197
\glsdetoklabel	379	\@glo@sortentries: new	196
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new	198
\glsdetoklabel	379	\@glo@sorthandler@letter: new ...	198
\showglosort: added \glsdetoklabel	258	\@glo@sorthandler@nocase: new ...	198
\showglosymbol: added		\@glo@sorthandler@word: new	198
\glsdetoklabel	259	\@glo@sortmacro@case: new	199
\showglosymbolaccess: added		\@glo@sortmacro@def: new	200
\glsdetoklabel	378	\@glo@sortmacro@def@do: new	200
\showglosymbolplural: added		\@glo@sortmacro@letter: new	199
\glsdetoklabel	259	\@glo@sortmacro@nocase: new	200
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ...	199
\glsdetoklabel	378	\@glo@sortmacro@use: new	201
\showglotext: added \glsdetoklabel	256	\@glo@sortmacro@word: new	199
\showglotextaccess: added		\@gls@noidx@do: new	202
\glsdetoklabel	378	\@gls@noidx@getgrouptitle: new ..	214
\showglotype: added \glsdetoklabel	257	\@gls@noref@warn: new	180
\showglouseri: added		\@gls@reference: new	205
\glsdetoklabel	257	\@gls@warnonglossdefined: new	20
\showglouserii: added		\@gls@warnontheGLOSSdefined: new .	21
\glsdetoklabel	257	\@no@makeglossaries: new	180
\showglouseriii: added		\@print@glossary: new	194
\glsdetoklabel	257	\@print@noidx@glossary: new	201
\showglouseriv: added		\@printgloss@setsort: new	192
\glsdetoklabel	257	\@printglossary: new	192
\showglouserv: added		General: added sort key to printgloss	
\glsdetoklabel	258	group	207
\showglouservi: added		\compatibleglossentry: changed	
\glsdetoklabel	258	\newcommand to \def as is may or	
dua: fixed bug in \acrfullfmt	233	may not be defined	342
fixed bug in \Acrfullplfmt	233	\compatiblesubglossentry: changed	
fixed bug in \acrfullplfmt	233	\newcommand to \def as is may or	
redefined to use accessibility		may not be defined	343
information	367	\defglsdisplayfirst: fixed unwanted	
dua-desc: commented spurious EOL ..	234	space	110
		\glo@grabfirst: new	202

\gls@defglossaryentry: replaced \ifx with \ifdefvoid	89	4.08 (2014-07-30)	\@ACRlong: added \do@gls@link@checkfirsthyper	361
\glsnoidxdisplayloc: new	204		\@ACRshort: added \do@gls@link@checkfirsthyper	360
\glsnoidxdisplayloclisthandler: new	204		\@Acrlong: added \do@gls@link@checkfirsthyper	361
\glsnoidxloclist: new	204		\@Acrshort: added \do@gls@link@checkfirsthyper	360
\glsnoidxloclisthandler: new	204		\@GLS@: moved \glsifhyper	127
\glsnoidxstripaccents: new	23		moved check for first use to \@gls@link	127
alttree: moved hangindent and parindent assignments outside level test	319		\@GLSpl: moved \glsifhyper	130
\makeglossaries: Moved definition of \glswrite to \makeglossaries ..	174		moved check for first use to \@gls@link	130
\makenoidxglossaries: new	176		\@Gls@: moved \glsifhyper	127
\printglossary: changed to use new \@printglossary	192		moved check for first use to \@gls@link	127
\printnoidxglossaries: new	192		\@Glspl@: moved \glsifhyper	129
\printnoidxglossary: new	192		moved check for first use to \@gls@link	129
\showgloclist: new	260		\@acrlong: added \do@gls@link@checkfirsthyper	361
\warn@noprintglossary: Activate warning in \makeglossaries	191		\@acrshort: added \do@gls@link@checkfirsthyper	359
\writeist: checked for definition of \glswrite	163, 167		\@closegls: new	173
4.06 (2014-03-12)			\@gls@: moved \glsifhyper	126
\@GLS@: added \glsifhyper	127		moved check for first use to \@gls@link	126
\@GLSpl@: added \glsifhyper	130		\@gls@automake: new	173
\@Gls@: added \glsifhyper	127		\@gls@doautomake: new	30
\@Glspl@: added \glsifhyper	129		\@gls@field@link: added assignment of \do@gls@link@checkfirsthyper	131
\@gls@: added \glsifhyper	126		\@gls@forbidtexext: new	62
\@gls@numbersdef: added hook to set toc title	33		\@gls@hyp@opt: new	112
\@gls@symbolsdef: added hook to set toc title	33		\@gls@link: removed redundancy	114
\@glsdisp: added \glsifhyper	131		renamed \gls@type to \glstype ...	114
\@glspl@: added \glsifhyper	128		\@gls@link@checkfirsthyper: new .	113
General: added \glsifhyper	144–151		\@glsdisp: moved \glsifhyper	131
acronym: added hook to set toc title	17		moved check for first use to \@gls@link	131
acronyms: added hook to set toc title ...	17		\@glspl@: moved \glsifhyper	128
\glsdefmain: added hook to set toc title	16		moved check for first use to \@gls@link	128
4.07 (2014-04-04)			\@ignored@glossaries: new	65
\@glossarysection: added optional argument when using unstarred version	44		General: added entrycounter option to printgloss family	206
\@gls@noidx@do: added \global in case it's used in a tabular-like style	202			
\Acrfullplfmt: fixed no case change bug	223			
\glsletentryfield: new	151			

added nopostdot option to printgloss family	206	removed \@sglstext	131
added subentrycounter option to printgloss family	206	removed \@sGLSuseriii	141
explicitly initialise hyper key	111	removed \@sGlsuseriii	141
moved \glsifhyper	144–151	removed \@sglsuseriii	140
removed \@sACRlongpl	150	removed \@sGLSuserii	140
removed \@sAcrlongpl	150	removed \@sGlsuserii	140
removed \@sacrlongpl	149	removed \@sGLSuseriv	142
removed \@sACRlong	148	removed \@sGlsuseriv	141
removed \@sAcrlong	148	removed \@sglsuseriv	141
removed \@sacrlong	147	removed \@sGLSuseri	139
removed \@sACRshortpl	146	removed \@sGlsuseri	139
removed \@sAcrshortpl	146	removed \@sglsuseri	139
removed \@sacrshortpl	145	removed \@sGLSuservi	143
removed \@sACRshort	145	removed \@sGlsuservi	143
removed \@sAcrshort	144	removed \@sglsuservi	143
removed \@sacrshort	144	removed \@sGLSuser	142
removed \@sgls@link	113	removed \@sGlsuser	142
removed \@sGLSdescplural	137	removed \@sglsuser	142
removed \@sGlsdescplural	137	removed \@sGLS	127
removed \@sglsdescplural	136	removed \@sGls	126
removed \@sGLSdesc	136	removed \@sgls	125
removed \@sGlsdesc	136	removed \@thirdofthree (defined in kernel)	125
removed \@sglsdesc	135	removed sPGLS	267
removed \@sglsdisp	130	removed sPgls	265
removed \@sGLSfirstplural	134	removed spgls	264
removed \@sGlsfirstplural	134	removed sPGLSpl	267
removed \@sglsfirstplural	134	removed sPglspl	266
removed \@sGLSfirst	133	removed spglspl	265
removed \@sGlsfirst	132	\ACRfull: removed \s@ACRfull	222
removed \@sglsfirst	132	switched to using \@gls@hyp@opt ..	222
removed \@sGLSname	135	\Acrfull: removed \s@Acrfull	221
removed \@sGlsname	135	switched to using \@gls@hyp@opt ..	221
removed \@sglsname	135	\acrfull: removed \@sacrfull	220
removed \@sGLSplural	134	switched to using \@gls@hyp@opt ..	220
removed \@sGlsplural	133	\ACRfullpl: removed \s@ACRfullpl ..	223
removed \@sglsplural	133	switched to using \@gls@hyp@opt ..	223
removed \@sGLSpl	129	\Acrfullpl: removed \s@Acrfullpl ..	222
removed \@sGlspl	129	switched to using \@gls@hyp@opt ..	222
removed \@sglspl	128	\acrfullpl: removed \s@acrfullpl ..	222
removed \@sGLSsymbolplural	138	switched to using \@gls@hyp@opt ..	222
removed \@sGlsymbolplural	138	\ACRlong: switched to using \@gls@hyp@opt	148
removed \@sglsymbolplural	138	\Acrlong: switched to using \@gls@hyp@opt	148
removed \@sGLSsymbol	137	\acrlong: switched to using \@gls@hyp@opt	147
removed \@sglsymbol	137		
removed \@sGLStext	132		
removed \@sGlstext	132		

\ACRlongpl: switched to using \@gls@hyp@opt	150	\glsenablehyper: added \KV@glslink@hypertrue to definition	125
\Acrlongpl: switched to using \@gls@hyp@opt	150	\GLSfirst: switched to using \@gls@hyp@opt	133
\acrlongpl: switched to using \@gls@hyp@opt	149	\Glsfirst: switched to using \@gls@hyp@opt	132
\ACRshort: switched to using \@gls@hyp@opt	145	\glsfirst: switched to using \@gls@hyp@opt	132
\Acrshort: switched to using \@gls@hyp@opt	144	\GLSfirstplural: switched to using \@gls@hyp@opt	134
\acrshort: switched to using \@gls@hyp@opt	143	\Glsfirstplural: switched to using \@gls@hyp@opt	134
\ACRshorttpl: switched to using \@gls@hyp@opt	146	\glsfirstplural: switched to using \@gls@hyp@opt	134
\Acrshorttpl: switched to using \@gls@hyp@opt	146	\glsifhyper: deprecated	112
\acrshorttpl: switched to using \@gls@hyp@opt	145	\glslink: switched to using \@gls@hyp@opt	113
\forallacronyms: new	55	\glslinkcheckfirsthyperhook: new	114
\GLS: switched to using \@gls@hyp@opt	127	\glslinkvar: new	112
\Gls: switched to using \@gls@hyp@opt	126	\GLSname: switched to using \@gls@hyp@opt	135
\gls: switched to using \@gls@hyp@opt	125	\Glsname: switched to using \@gls@hyp@opt	135
\gls@defglossaryentry: added check for ignored glossary	85	\glsname: switched to using \@gls@hyp@opt	135
\gls@istfilebase: new	40	\GLSpl: switched to using \@gls@hyp@opt	129
\glsaddkey: removed \@sGLS@user@<key>	79	\Glspl: switched to using \@gls@hyp@opt	129
removed \@sGls@user@<key>	79	\glspl: switched to using \@gls@hyp@opt	128
removed \@sgls@user@<key>	78	\GLSplural: switched to using \@gls@hyp@opt	133
switched to using \@gls@hyp@opt	78, 79	\Glsplural: switched to using \@gls@hyp@opt	133
\GLSdesc: switched to using \@gls@hyp@opt	136	\glsplural: switched to using \@gls@hyp@opt	133
\Glsdesc: switched to using \@gls@hyp@opt	136	\glsspace: new	221
\glsdesc: switched to using \@gls@hyp@opt	135	\GLSsymbol: switched to using \@gls@hyp@opt	138
\GLSdescplural: switched to using \@gls@hyp@opt	137	\Glsymbol: switched to using \@gls@hyp@opt	137
\Glsdescplural: switched to using \@gls@hyp@opt	136	\glsymbol: switched to using \@gls@hyp@opt	137
\glsdescplural: switched to using \@gls@hyp@opt	136	\GLSsymbolplural: switched to using \@gls@hyp@opt	138
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	125	\Glsymbolplural: switched to using \@gls@hyp@opt	138
\glsdisp: switched to using \@gls@hyp@opt	130		
\glsdohyperlink: new	124		
\glsdohypertarget: new	124		

\glssymbolplural: switched to using \@gls@hyp@opt	138	\newglossary: added starred version ..	63
\GLStext: switched to using \@gls@hyp@opt	132	\newignoredglossary: new	65
\Glstext: switched to using \@gls@hyp@opt	132	\ns@newglossary: added \@glotype@<name>@log	63
\glstext: switched to using \@gls@hyp@opt	131	new	63
\glstreenamefmt: new	312	\p@gls@hyp@opt: new	112
\GLSuseri: switched to using \@gls@hyp@opt	139	\PGLS: changed to use \@gls@hyp@opt	267
\Glsuseri: switched to using \@gls@hyp@opt	139	\Pgls: changed to use \@gls@hyp@opt	265
\glsuseri: switched to using \@gls@hyp@opt	139	\pgls: changed to use \@gls@hyp@opt	264
\GLSuserii: switched to using \@gls@hyp@opt	140	\PGLSpl: changed to use \@gls@hyp@opt	267
\Glsuserii: switched to using \@gls@hyp@opt	140	\Pglspl: changed to use \@gls@hyp@opt	266
\glsuserii: switched to using \@gls@hyp@opt	139	\pglspl: changed to use \@gls@hyp@opt	265
\GLSuseriii: switched to using \@gls@hyp@opt	141	\s@gls@hyp@opt: new	112
\Glsuseriii: switched to using \@gls@hyp@opt	141	\s@newglossary: new	63
\glsuseriii: switched to using \@gls@hyp@opt	140	automake: new	30
\GLSuseriv: switched to using \@gls@hyp@opt	142	4.09 (2014-08-12)	
\Glsuseriv: switched to using \@gls@hyp@opt	141	\glsaddkey: fixed bug in user commands	78
\glsuseriv: switched to using \@gls@hyp@opt	141	4.10 (2014-08-27)	
\GLSuserv: switched to using \@gls@hyp@opt	142	\@Gls@acrentryname: new	152
\Glsuserv: switched to using \@gls@hyp@opt	142	\@Gls@entryname: new	152
\glsuserv: switched to using \@gls@hyp@opt	143	\@gls@glossary: Renamed \@glossary to \@gls@glossary	182
\ifignoredglossary: new	65	\glspercentchar: new	162
altlongragged4col: fixed bug that displayed description instead of symbol	292	\glstildechar: new	162
		alttree: moved space after symbol	319, 320
		4.11 (2014-09-01)	
		\@do@esc@wrglossary: added hook ..	187
		sanitize: none option	25
		\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	182
		\glsaddprotectedpagefmt: new	184
		\glsbackslash: new	162
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute	49
		\@gls@adjustmode: new	160
		\@gls@notranslatorhook: removed ...	25
		\@gls@toc: added \protect to \numberline	46
		\@gls@usetranslator: new	26
		\glsacrpluralsuffix: new	37
		\glsadd: added check for vertical mode	160
		\glsaddallunused: replaced @gobble with glsignore	161
		\glsifusedtranslatordict: new	26
		\glsignore: new	161

\glsupacrpluralsuffix:new	37	4.16 (2015-06-18)	
\ProvidesGlossariesLang:new	37	\glsaddstoragekey:new	76
\RequireGlossariesLang:new	37	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	362
\indexspace:new	274, 294, 312	\@ACRshort: added \glspostlinkhook	361
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	361
\@glslocalreset:new	94	\@Acrshort: added \glspostlinkhook	360
\@glslocalunset:new	93	\@GLS@: added \glspostlinkhook ...	128
\@glsreset:new	94	\@GLSpl: added \glspostlinkhook ..	130
\@glsunset:new	93	\@Gls@: added \glspostlinkhook ...	127
\@newglossaryentry@defcounters:		\@Glspl@: added \glspostlinkhook ..	129
new	95	\@acrlong: added \glspostlinkhook	361
\cGls:new	98	\@acrshort: added \glspostlinkhook	360
\cGls@:new	98	\@gls@: added \glspostlinkhook ...	126
\cGlspl@:new	99	\@gls@@link: added	
\cgls:new	98	\glspostlinkhook	113
\cgls@:new	98	\@gls@field@link: added	
\cglspl@:new	99	\glspostlinkhook	131
\cglspl@:new	99	\@gls@link: moved definition of	
\gls@entry@count:new	98	\glsifhyperon outside of this	
\gls@increment@currcount:new ...	97	macro	115
\gls@local@increment@currcount:		\@glsdisp: added \glspostlinkhook	131
new	97	\@glspl@: added \glspostlinkhook ..	128
\gls@write@entrycounts:new	97	General: added \glspostlinkhook	144–151
\glslocalreset:new	94	\glsacspace:new	228
\glslocalunset:new	93	\glsadd: changed \@do@wrglossary to	
\glsreset:new	94	\@do@wrglossary	160
\glsunset:new	93	\glsfielddef:new	81
\@newglossaryentry@defcounters:		\glsfieldedef:new	80
new	89	\glsfieldfetch:new	81
\cGls:new	98	\glsfieldgdef:new	80
\cgls:new	98	\glsfieldxdef:new	80
\cGlsformat:new	99	\glsifhyperon: moved definition of	
\cglsformat:new	98	\glsifhyperon	114
\cGlspl:new	99	\glslinkpostsetkeys:new	114
\cglspl:new	99	\glspostlinkhook:new	113
\cGlsplformat:new	100	\glswriteentry:new	183
\cglsplformat:new	99	\ifglsfieldcseq:new	82
\gls@defdocnewglossaryentry:new ..	73	\ifglsfielddefeq:new	82
\glsenableentrycount:new	95	\ifglsfielddeq:new	81
\glslocalreset: switched to		long-sp-short:new	227
\@glslocalreset	92	\showglofield:new	260
\glslocalunset: switched to		4.18 (2015-09-09)	
\@glslocalunset	93	General: split mfirstuc into separate	
\glsreset: switched to \@glsreset ...	92	bundle	4
\glsunset: switched to \@glsunset ...	93	4.19 (2015-10-31)	
4.15 (2015-03-16)		\glstreenamebox:new	318
General: bug fix replaced \@glo@type		4.19 (2015-11-22)	
with \glstype	150	\@gls@link@nocheckfirsthyper:new	131

\@gls@preglossaryhook: new	192	\glsfindwidesttoplevelname: new .	318
\@printglossary: added		\glslistgroupheaderfmt: new	274
\@gls@preglossaryhook	194	\glslistnavigationitem: new	274
\do@glsglisablehyperinlist: new ..	114	\glstreegroupheaderfmt: new	312
\doifglossarynoexistsordo: new ...	58	\glstreenavigationfmt: new	312
\gls@gobbleopt: new	62	\ifglswrallowprimitivemods: new .	185
\glsglsoifexistsordo: new	57	list: fixed missing space before	
4.20 (2015-11-30)		description	274
\@gls@link: added		long: fixed typo in \glossentrydesc .	278
\@gls@setdefault@glslink@opts	114	super4col: fixed bug in \glossentry .	303
added \glsglsonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed	115	\glsglscurrentfieldvalue: new	61
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new	114	\glsglscurrentfieldvalue	60, 61
\glsgl@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	68	nogroupskip changed	292
\glsglsonohyperlink: new	124	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	311
\@printglossary: warn if no style has		long: check for nogroupskip changed ..	278
been set	193	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	284
assignment	144–150	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed	280
already set	215	long3col-booktabs: check for	
\glsglLTpenaltycheck: new	287	nogroupskip changed	285
\glsglpatchLToutput: new	287	long4col: check for nogroupskip	
\glsglspenaltygroupskip: new	287	changed	281
altlong4col-booktabs: new	285	long4col-booktabs: check for	
altlongragged4col-booktabs: new .	286	nogroupskip changed	285
long-booktabs: new	284	longragged: check for nogroupskip	
long3col-booktabs: new	284	changed	289
long4col-booktabs: new	285	longragged3col: check for nogroupskip	
longragged-booktabs: new	286	changed	290
longragged3col-booktabs: new	286	super: check for nogroupskip changed .	300
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set	215	changed	302
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@esc@wrglossary: added check		changed	304
for \@arabic	186	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	307
modifications and added arabic case	186	superragged3col: check for	
mcolalttreespannav: new	299	nogroupskip changed	309
mcolindexspannav: new	295	4.24 (2016-05-27)	
mcoltreenonamespannav: new	297	\@gls@extramakeindexopts: new ...	172
mcoltreespannav: new	296	\@gls@glossary: added check for debug	
\glsgl@arabicpage: new	183	mode	182
\glsgl@protected@pagefmts: added		\@gls@see@noindex: new	6
arabic to list	183	debug: new	5
\glsglentrytitlecase: new	155	seenoinindex: new	6

\glsnomakeindexwarning: new	46	added \TH, \dh and \DH	24
\GlsSetQuote: new	169	4.31 (2017-08-10)	
\GlsSetWriteIstHook: new	169	nolist: added check for “list” style	10
4.25 (2016-06-09)		4.31 (2017-09-10)	
\@gls@enablesavenonumberlist: new	69	style: changed \renewcommand to \def .	8
\@gls@initnonumberlist: new	69	4.32 (2017-08-24)	
\@gls@savenonumberlist: new	69	\@glsnavhypertarget: new	269
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	294	\gls@the and \gls@number	187
mcolindexspannav: replaced \@idxitem		renamed from	
with \glstreeitem	295	\@do@esc@wrglossary	185
\glstreechildpredesc: new	313	\@do@noesc@wrglossary: new	184
\glstreeitem: new	312	\@do@wrglossary: changed to check	
\glstreepredesc: new	313	for esclocations	184
\glstreesubitem: new	313	\@gls@missinglang@warn: new	20
\glstreesubsubitem: new	313	\GlsSetXdyFirstLetterAfterDigits:	
4.28 (2017-01-07)		added starred version	162
\glspatchtabularx: new	92	\GlsSetXdyNumberGroupOrder: new .	162
4.29 (2017-01-19)		esclocations: new	9
\@gls@noidx@do: current letter group		4.34 (2017-11-03)	
assignment made global	203	mcolalttreespannav: removed spurious	
\@print@noidx@glossary: moved		space	299
definition of		\glsshowtarget: modified to check for	
\@gls@currentlettergroup outside		math mode and inner	6
of the glossary environment	201	4.35 (2017-11-14)	
General: added check for		\glsadd: added \@gls@setsort (in case	
\@glsxtr@doaccsupp	342	of sort=use)	160
\glsnavhyperlinkname: new	269	4.36 (2018-03-07)	
4.30 (2017-06-11)		\@gls@glossary: removed \index ...	182
\@glo@autosee: new	89	4.37 (2018-04-07)	
\@glo@autoseehook: new	89	\gls@begindocdefs: new	74
\@glo@check@sortallowed: new	13	4.38 (2018-05-10)	
\@gls@noidx@do: letter group		\@gls@define@glossaryentrycounter:	
assignment made global	203	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	51	new	12
\@xdylocationclassorder: bug fix:		prepended \currentglossary. to	
changed \edef to \def	52	\theHglossarysubentry and	
\glosortentrieswarning: new	20	removed spurious eol space	12
\gls@set@xr@key: new	68	\glsaccsupp: added braces around	
\gls@xr@key: new	68	actual text argument	348
\GlsAddXdyLocation: bug fix: changed		\glsentrycounterlabel: bug fix: move	
#1 to #2	52	conditional inside command	209
\glsnoidxstripaccents: added \a ...	23	\GlsEntryCounterLabelPrefix: new	206

\glsentryitem: bug fix: move conditional inside command	209	\gls@numberedsection@nr	206
\glsrefentry: bug fix: move conditional inside command	209	debug: changed \val and \nr to \gls@debug@val and \gls@debug@nr	5
\glsresetsubentrycounter: bug fix: move conditional inside command .	208	seenoindex: changed \val and \nr to \gls@seenoindex@val and \gls@seenoindex@nr	6
\glsstepentry: bug fix: move conditional inside command	208	kernelglossredefs: new	32
\glsstepsubentry: bug fix: move conditional inside command	208	\glossary: added warning	32
\glssubentrycounterlabel: bug fix: move conditional inside command .	209	\gls@original@glossary: new	31
\glssubentryitem: bug fix: move conditional inside command	209	\gls@original@makeglossary: new ..	31
\showglongnameaccess: bug fix: corrected field (was showing text access field)	378	\makeglossaries: removed redefinition of \makeglossary	175
4.40 (2018-06-01)		\makeglossary: added warning	31
\istfile: changed \def to \providecommand	180	nonumberlist: changed \val and \nr to \gls@nonumberlist@val and \gls@nonumberlist@nr	68
\makenoidxglossaries:false	177	translate: changed \val and \nr to \gls@translate@val and \gls@translate@nr	26
4.41 (2018-07-23)		numberedsection: changed \val and \nr to \gls@numberedsection@val and \gls@numberedsection@nr	8
\@gls@override@glossary: new	31		
General: changed \val and \nr to \gls@numberedsection@val and			

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	
\!	120, 121
\"	23, 118–121, 123
\#	165, 166
\%	162, 168, 326, 327
\&	37, 159
\'	23
\.	11, 23
\=	23
\?	118–120, 170
\@	74
\@@delimN	217
\@@do@@wrglossary	177, 185, 188
\@@do@esc@wrglossary	184
\@@do@noesc@wrglossary	184
\@@do@wrglossary	160, 183
\@@glo@assign@sortkey	177
\@@glo@list	55
\@@glo@sort	23
\@@glo@type	192
\@@glossarysec	7, 44, 45
\@@glossaryseclabel	8, 44, 45, 206
\@@glossarysecstar	8, 44, 45, 206
\@@gls@checkactual	122
\@@gls@checkbar	121
\@@gls@checkescactual	120
\@@gls@checkescbar	120
\@@gls@checkesclevel	121
\@@gls@checkescquote	119, 171, 172
\@@gls@checklevel	122
\@@gls@checkquote	118, 119, 169, 170
\@@gls@default@entryfmt	101, 110
\@@gls@expand@field	21, 72, 73, 77, 78, 239, 241, 243, 245, 247, 250, 252, 373–377
\@@gls@extramakeindexopts	169, 174, 175
\@@gls@fixbraces	190
\@@gls@noexpand@field	21, 71, 72
\@@gls@noidx@no@sanitizesort	23
\@@gls@noidx@nosanitizesort	179
\@@gls@nosanitizesort	22, 179
\@@gls@sanitizesort	22, 179
\@@gls@xdycheckbackslash	123, 124
\@@gls@xdycheckquote	123
\@@gls@localreset	94, 96
\@@gls@localunset	93, 96
\@@glsreset	94, 96
\@@glsunset	93, 96
\@@newglossaryentry@defcounters	95
\@@this@glo@	56
\@ACRfull	222
\@ACRfullpl	223
\@ACRlong	148, 222
\@ACRlongpl	150, 223
\@ACRshort	145, 222
\@ACRshortpl	146, 147, 223
\@Acrfull	221
\@Acrfullpl	222, 223
\@Acrlong	148, 221
\@Acrlongpl	150, 223
\@Acrshort	144
\@Acrshortpl	146
\@Alph	183, 186, 187
\@GLS	127
\@GLS@	127, 267
\@GLSdesc	136
\@GLSdesc@	136
\@GLSdescplural	137
\@GLSdescplural@	137
\@GLSfirst	133
\@GLSfirst@	133
\@GLSfirstplural	134
\@GLSfirstplural@	134
\@GLSname	135
\@GLSname@	135

\@GLSpl	129	\@Glsuseri	139
\@GLSpl@	129, 130, 268	\@Glsuseri@	139
\@GLSplural	133, 134	\@Glsuserii	140
\@GLSplural@	134	\@Glsuserii@	140
\@GLSsymbol	138	\@Glsuseriii	141
\@GLSsymbol@	138	\@Glsuseriii@	141
\@GLSsymbolplural	138	\@Glsuseriv	141
\@GLSsymbolplural@	138, 139	\@Glsuseriv@	141
\@GLStext	132	\@Glsuserv	142
\@GLStext@	132	\@Glsuserv@	142
\@GLSuseri	139	\@Glsuservi	143
\@GLSuseri@	139	\@Glsuservi@	143
\@GLSuserii	140	\@Mi	287
\@GLSuserii@	140	\@PGLS	267
\@GLSuseriii	141	\@PGLS@	267
\@GLSuseriii@	141	\@PGLSpl	267
\@GLSuseriv	142	\@PGLSpl@	267
\@GLSuseriv@	142	\@Pgls	265
\@GLSuserv	142	\@Pgls@	265
\@GLSuserv@	142, 143	\@Pglspl	266
\@GLSuservi	143	\@Pglspl@	266
\@GLSuservi@	143	\@Roman	184, 186, 187
\@Gls	126	\@acrfull	220
\@Gls@	96, 98, 126, 266	\@acrfullpl	222
\@Gls@acrentryname	224	\@acrlong	147, 221
\@Gls@entry@field	78, 152–157	\@acrlongpl	149, 222
\@Gls@entryname	152, 224	\@acrshort	144, 221
\@GlsSetXdyFirstLetterAfterDigits	162	\@acrshortpl	145, 222, 223
\@GlsSetXdyNumberGroupOrder	162, 163	\@addtoacronymlists	18
\@Glsdesc	136	\@after	18
\@Glsdesc@	136	\@afterheading	275, 330
\@Glsdescplural	136, 137	\@alph	183, 186, 187
\@Glsdescplural@	137	\@arabic	183, 186, 187
\@Glsfirst	132	\@auxout	62,
\@Glsfirst@	132, 133		63, 97, 174, 175, 177, 180, 191, 195, 270
\@Glsfirstplural	134	\@backslashchar	117, 123, 124
\@Glsfirstplural@	134	\@before	18
\@Glsname	135	\@bsphack	182
\@Glsname@	135	\@cGls	98
\@Glspl	129	\@cGls@	96, 98
\@Glspl@	97, 99, 129, 266, 267	\@cGlspl	99
\@Glsplural	133	\@cGlspl@	97, 99
\@Glsplural@	133	\@cclv	287, 288
\@Glssymbol	137	\@cgls	98
\@Glssymbol@	137	\@cgls@	96, 98
\@Glssymbolplural	138	\@cglspl	99
\@Glssymbolplural@	138	\@cglspl@	96, 99
\@Glstext	132	\@chapter	35
\@Glstext@	132	\@classoptionslist	34

<code>\@closegls</code>	173, 174	<code>\@glo@autoseehook</code>	89
<code>\@colht</code>	287	<code>\@glo@check@mkidxrangechar</code>	116, 117, 188, 323, 324
<code>\@colroom</code>	287, 288	<code>\@glo@check@sortallowed</code> ..	13–16, 176, 179
<code>\@currentlabelname</code>	8, 206	<code>\@glo@childlist</code>	196
<code>\@curroptions</code>	34	<code>\@glo@counter</code>	68, 84, 87
<code>\@declaredoptions</code>	34	<code>\@glo@counterprefix</code>	180, 185, 187–189, 214, 215, 218
<code>\@delimN</code>	217	<code>\@glo@default@sorttype</code> ..	13, 178, 199, 200
<code>\@delimR</code>	217	<code>\@glo@defaultcounter</code>	87
<code>\@disable@onlypremakeg</code>	175	<code>\@glo@desc</code>	66, 83–85, 88
<code>\@disable@premakecs</code>	36	<code>\@glo@descaccess</code>	344–346
<code>\@disabled@gl saddxdycounters</code>	48	<code>\@glo@descplural</code>	66, 83, 84
<code>\@do@addcounter</code>	47	<code>\@glo@descpluralaccess</code>	344–346
<code>\@do@auxoutstuff</code>	195	<code>\@glo@do@sortentries</code>	196
<code>\@do@glossentry</code>	210, 342, 343	<code>\@glo@entry</code>	161
<code>\@do@gls@getcounterprefix</code>	185, 187	<code>\@glo@entryprefix</code>	262
<code>\@do@gls@islistofacronyms</code>	18	<code>\@glo@entryprefixfirst</code>	262
<code>\@do@glssee</code>	89	<code>\@glo@entryprefixfirstplural</code> ..	262, 263
<code>\@do@ifinlist</code>	46, 47	<code>\@glo@entryprefixplural</code>	262
<code>\@do@newglossaryentry</code> ..	224, 238, 239, 241–243, 245, 247, 250, 252, 254, 373–377	<code>\@glo@esclabel</code>	90, 91
<code>\@do@seeglossary</code>	177, 190	<code>\@glo@etext</code>	101–103
<code>\@do@subglossentry</code>	211, 212, 343	<code>\@glo@first</code>	67, 84, 87, 88, 250, 377
<code>\@do@wrglossary</code>	115	<code>\@glo@firstaccess</code>	343, 345, 346
<code>\@do@writeaux@info</code>	191	<code>\@glo@firstplural</code>	67, 84, 87, 377
<code>\@ehc</code>	287	<code>\@glo@firstpluralaccess</code>	344–346
<code>\@empty</code> 11, 15, 18, 30, 34, 35, 46, 48, 51, 54, 55, 85, 86, 91, 115, 116, 126–130, 144–150, 163, 166, 168, 169, 173, 174, 181, 182, 189, 214, 239, 241, 243–246, 248, 249, 251, 252, 254, 324, 326, 328, 360–362		<code>\@glo@grabfirst</code>	202
<code>\@end@fixbraces</code>	190	<code>\@glo@label</code>	70, 71, 77, 78, 80–82, 84–89, 95, 159, 262, 263, 318, 346
<code>\@endfortrue</code>	27, 58, 76, 270	<code>\@glo@list</code>	89
<code>\@esphack</code>	183	<code>\@glo@long</code>	59, 70, 85, 87
<code>\@expandtwoargs</code>	34	<code>\@glo@longaccess</code>	344–346
<code>\@firstofone</code>	23	<code>\@glo@longpl</code>	70, 85, 87, 239, 241, 243, 245, 247, 250, 252, 373
<code>\@firstofthree</code>	112, 125, 126, 128, 130, 144, 146, 147, 149, 360–362	<code>\@glo@longpluralaccess</code>	344–346
<code>\@firstoftwo</code>	26, 27, 75, 76, 112, 128–130, 146, 147, 149, 150	<code>\@glo@name</code>	13, 66, 71, 84, 86–88
<code>\@for</code>	27, 34, 36, 47, 48, 55, 75, 76, 117, 163–165, 175, 176, 183, 190, 196, 226, 239, 242, 244, 246, 248, 251, 253, 255, 270, 271, 324	<code>\@glo@no@assign@sortkey</code>	176
<code>\@glo@@desc</code>	88	<code>\@glo@nonumberlist</code>	69
<code>\@glo@@symbol</code>	88	<code>\@glo@numfmt</code>	188, 324
<code>\@glo@access</code>	343, 345, 346, 348	<code>\@glo@parent</code>	15, 68, 84, 86, 87, 91, 197
<code>\@glo@addchildren</code>	196, 201	<code>\@glo@plural</code>	66, 84, 86, 87, 375
<code>\@glo@assign@sortkey</code>	176, 177, 207	<code>\@glo@pluralaccess</code>	344–346
<code>\@glo@autosee</code>	89	<code>\@glo@prefix</code>	9, 68, 69, 84, 91, 116, 117, 188, 323, 324
		<code>\@glo@range</code>	188, 323, 324
		<code>\@glo@see</code>	68, 84, 89
		<code>\@glo@seeautonumberlist</code>	9, 68
		<code>\@glo@short</code>	59, 60, 70, 85, 87, 376
		<code>\@glo@shortaccess</code>	344–346, 373–376

<code>\@glo@shortpl</code>	70, 85, 87, 238,	<code>\@gls@Hcounter</code>	115, 116
	239, 241, 243, 245, 247, 250, 252, 373, 376	<code>\@gls@ReturnAfterFi</code>	218
<code>\@glo@shortpluralaccess</code>	344–346	<code>\@gls@access@display</code>	348–350
<code>\@glo@sort</code>	13, 16, 22, 23, 66, 84, 87, 91	<code>\@gls@actualchar</code> .	91, 120, 122, 167, 168, 327
<code>\@glo@sortedinsert</code>	197	<code>\@gls@addpredefinedattributes</code> .	163, 172
<code>\@glo@sortentries</code>	199, 200	<code>\@gls@adjustmode</code>	160
<code>\@glo@sorthandler@case</code>	200	<code>\@gls@automake</code>	176
<code>\@glo@sorthandler@letter</code>	199	<code>\@gls@between</code>	271
<code>\@glo@sorthandler@nocase</code>	200	<code>\@gls@body</code>	152
<code>\@glo@sorthandler@word</code>	199	<code>\@gls@checkactual</code>	118, 170
<code>\@glo@sortinghandler</code>	196, 197	<code>\@gls@checkbar</code>	118, 170
<code>\@glo@sortinglist</code>	196, 197, 200	<code>\@gls@checkedmkidx</code>	117–123, 169–171
<code>\@glo@sorttype</code>	178, 201, 202, 207	<code>\@gls@checkescactual</code>	118, 170
<code>\@glo@storeentry</code>	13–15	<code>\@gls@checkescbar</code>	118, 171
<code>\@glo@suffix</code>	116, 117, 188, 324	<code>\@gls@checkescquote</code>	118, 170, 171
<code>\@glo@symbol</code>	59, 67, 84, 88, 244, 249, 345	<code>\@gls@checklevel</code>	118, 171
<code>\@glo@symbolaccess</code>	344–346, 376	<code>\@gls@checkmkidxchars</code>	
<code>\@glo@symbolplural</code>	67, 84, 88		.. 90, 91, 116, 170, 177, 187, 189, 323, 324
<code>\@glo@symbolpluralaccess</code>	344–346	<code>\@gls@checkquote</code>	118, 169, 170
<code>\@glo@text</code>	66,	<code>\@gls@classI</code>	164
	84, 87, 88, 126–131, 151–153, 245, 263, 375	<code>\@gls@classII</code>	164
<code>\@glo@textaccess</code> ...	343, 345, 346, 373–376	<code>\@gls@codepage</code>	195
<code>\@glo@thislabel</code>	90	<code>\@gls@counter</code>	
<code>\@glo@thislettergrp</code>	202, 203	 111, 114–116, 160, 180, 188, 189, 324
<code>\@glo@thisvalue</code>	60, 61	<code>\@gls@counterwithin</code>	11, 12
<code>\@glo@tmp</code>	77, 78, 189	<code>\@gls@ctr</code>	47
<code>\@glo@type</code>	8, 15, 67,	<code>\@gls@currentlettergroup</code>	201, 203
	84, 85, 87–89, 160, 161, 175, 180, 181,	<code>\@gls@debugfalse</code>	5
	192–197, 200–202, 205, 206, 224, 239,	<code>\@gls@debugtrue</code>	5
	241–243, 245, 246, 248, 251–255, 269, 271	<code>\@gls@declareoption</code>	
<code>\@glo@types</code>	55,	 9, 10, 16, 17, 20, 21, 26, 29, 30, 32, 33
	56, 63, 94, 95, 160, 161, 175, 176, 260, 318	<code>\@gls@default</code>	100
<code>\@glo@useri</code>	69, 84, 87	<code>\@gls@default@value</code>	
<code>\@glo@userii</code>	69, 84, 87	 59–61, 71, 72, 84, 86–88, 249, 262
<code>\@glo@useriii</code>	70, 84, 87	<code>\@gls@deffile</code>	74, 75
<code>\@glo@useriv</code>	70, 84, 87	<code>\@gls@define@glossaryentrycounter</code> .	
<code>\@glo@userv</code>	70, 84, 87	 12, 34, 206, 208
<code>\@glo@uservi</code>	70, 84, 87	<code>\@gls@define@glossarysubentrycounter</code>	35, 207, 208
<code>\@glodesc</code>	88	<code>\@gls@defsort</code>	13–15, 88
<code>\@glolist@</code>	85	<code>\@gls@defsortcount</code>	13–15, 64
<code>\@gloname</code>	88	<code>\@gls@do@acronymsdef</code>	17, 34, 35, 65
<code>\@glossary@default@style</code>	8, 10, 193, 215, 255	<code>\@gls@do@indexdef</code>	33–35, 65
<code>\@glossaryentryfield</code>	91	<code>\@gls@do@numbersdef</code>	33–35, 65
<code>\@glossarysection</code>	43	<code>\@gls@do@symbolsdef</code>	33, 65
<code>\@glossarystyle</code>	193, 194, 205	<code>\@gls@do@symbolssdef</code>	34, 35
<code>\@glossarysubentryfield</code>	91	<code>\@gls@doautomake</code>	30, 176
<code>\@gls</code>	125	<code>\@gls@docheckquotedef</code>	169–172
<code>\@gls@</code>	96, 98, 125, 265, 266	<code>\@gls@docloadedfalse</code>	4
<code>\@gls@@link</code>	113		

<code>\@gls@docloadedtrue</code>	4	<code>\@gls@link@checkfirsthyper</code>	126–130
<code>\@gls@dodedeflistparser</code>	176	<code>\@gls@link@label</code>	114, 240, 246
<code>\@gls@doentrycounterdef</code>	34, 35	<code>\@gls@link@nocheckfirsthyper</code>	131, 144–150
<code>\@gls@doentrydef</code>	110	<code>\@gls@link@opts</code>	114, 240, 246
<code>\@gls@dolast</code>	190	<code>\@gls@list</code>	270, 271
<code>\@gls@donext</code>	190	<code>\@gls@listsuffix</code>	46
<code>\@gls@donext@def</code>	159	<code>\@gls@loadlist</code>	10, 255
<code>\@gls@dosubentrycounterdef</code>	34, 35	<code>\@gls@loadlong</code>	10, 255
<code>\@gls@dotothiswrite</code>	173, 174	<code>\@gls@loadsuper</code>	10, 255
<code>\@gls@elem</code>	270	<code>\@gls@loadtree</code>	10, 255
<code>\@gls@enablesavenonumberlist</code>	74	<code>\@gls@local@increment@currcount</code>	96
<code>\@gls@encapchar</code>		<code>\@gls@loclist</code>	178, 179, 202, 203
..... 120, 121, 167, 168, 188, 189, 324, 327		<code>\@gls@map</code>	75, 76
<code>\@gls@entry@count</code>	97	<code>\@gls@missinglang@warn</code>	20, 38
<code>\@gls@entry@field</code>		<code>\@gls@missingnumberlist</code>	88
..... 77, 78, 95, 96, 152–158, 346–348		<code>\@gls@noaccess</code>	348
<code>\@gls@escbsdq</code>	118, 168, 328	<code>\@gls@noexpand@fields</code>	73
<code>\@gls@expand@fields</code>	72, 73	<code>\@gls@nohyperlist</code>	19, 65, 114
<code>\@gls@expandonce</code>	73	<code>\@gls@noidx@do</code>	201
<code>\@gls@extramakeindexopts</code>	175	<code>\@gls@noidx@getgrouptitle</code>	177
<code>\@gls@fetchfield</code>	60	<code>\@gls@noidx@sanitizesort</code>	22, 179
<code>\@gls@field@link</code>	78, 79, 131–143	<code>\@gls@noidx@setsanitizesort</code> ..	24, 25, 179
<code>\@gls@firsttok</code>	202	<code>\@gls@noidx@loclist@finalsep</code>	178
<code>\@gls@fixbraces</code>	89	<code>\@gls@noidx@loclist@prev</code>	178, 204
<code>\@gls@forbidtexext</code>	63	<code>\@gls@noidx@loclist@sep</code>	178, 204
<code>\@gls@get@counterprefix</code>	189	<code>\@gls@noref@warn</code>	176, 202
<code>\@gls@getbody</code>	152	<code>\@gls@numberlink</code>	217, 218
<code>\@gls@getcounterprefix</code>	185, 187	<code>\@gls@numbersdef</code>	33
<code>\@gls@getgrouptitle</code>	177, 213, 271	<code>\@gls@numlist@lastsep</code>	159
<code>\@gls@glossary</code>	181, 182	<code>\@gls@numlist@nextsep</code>	159
<code>\@gls@gobbleopt</code>	62	<code>\@gls@numlist@sep</code>	159
<code>\@gls@grptitle</code>	213, 269, 271	<code>\@gls@old@chapter</code>	35
<code>\@gls@hyp@opt</code>		<code>\@gls@oldnewglossaryentryposthook</code> .	345
79, 98, 99, 113, 125–150, 220–223, 264–267		<code>\@gls@oldnewglossaryentryprehook</code> ..	345
<code>\@gls@hyp@opt@cs</code>	112	<code>\@gls@onlypremakeg</code>	35, 36
<code>\@gls@hypergroup</code>	270	<code>\@gls@order</code>	173, 174
<code>\@gls@ifinlist</code>	47	<code>\@gls@org@LT@output</code>	287
<code>\@gls@ifnotmeasuring</code>	92	<code>\@gls@org@glsnoidxdisplayloc</code>	179
<code>\@gls@igtype</code>	65	<code>\@gls@org@glssseeformat</code>	179
<code>\@gls@increment@currcount</code>	96	<code>\@gls@override@glossary</code>	32
<code>\@gls@indexdef</code>	33	<code>\@gls@patchtabularx</code>	92
<code>\@gls@initnonumberlist</code>	69, 84	<code>\@gls@preglossaryhook</code>	194
<code>\@gls@islistofacronyms</code>	18	<code>\@gls@prevlevel</code> .	298, 299, 318–321, 336, 337
<code>\@gls@keylist</code>	372	<code>\@gls@provide@newglossary</code>	63
<code>\@gls@keymap</code>	69, 75–78, 262, 345	<code>\@gls@quotechar</code>	119–122, 167–171, 327
<code>\@gls@label</code>	177, 180, 185, 187, 188	<code>\@gls@reference</code>	177, 180
<code>\@gls@langmod</code>	173	<code>\@gls@removespaces</code>	218
<code>\@gls@levelchar</code> ..	91, 121, 122, 167, 168, 327	<code>\@gls@renewglossary</code>	173
<code>\@gls@link</code> ..	113, 126–131, 144–151, 360–362	<code>\@gls@replacementtext</code>	348

\@gls@rest	152	\@glsacronymlists	18, 19, 55, 224, 226, 239, 241–246, 248, 251–255, 260
\@gls@restoreat	74, 75	\@glsaddkey	77, 78
\@gls@roman	50, 324, 325	\@glsaddstoragekey	76, 77
\@gls@sanitized@tmp	117	\@glsaddxdyattribute	47, 48
\@gls@sanitizedesc	28	\@glsdefaultsort	13
\@gls@sanitizesort	13	\@glsdesc	135
\@gls@sanitizesymbol	28	\@glsdesc@	135, 136
\@gls@saveentrycounter	115, 160	\@glsdescplural	136
\@gls@savenonumberlist	69	\@glsdescplural@	136
\@gls@see@noindex	7, 68	\@glsdisp	130
\@gls@setacrstyle	28, 34, 35	\@glsentry	94, 95, 97
\@gls@setcounter	64	\@glsentrytitlecase	155
\@gls@setdefault@glslink@opts	114	\@glsfirst	132
\@gls@setsort	13–16, 115, 160	\@glsfirst@	132
\@gls@setupshortcuts	34, 35	\@glsfirstletter	162
\@gls@sort	203	\@glsfirstplural	134
\@gls@sort@A	198	\@glsfirstplural@	134
\@gls@sort@B	198	\@glshypernumber	217
\@gls@startswithexpandonce	72	\@glsisacronymlistfalse	19
\@gls@storenonumberlist	69, 88	\@glsisacronymlisttrue	19
\@gls@symbolsdef	33	\@glslink	115, 125, 159, 269
\@gls@this	183	\@glslocalreset	93, 96
\@gls@thisHloc	188	\@glslocalunset	93, 96
\@gls@thisfield	60, 61	\@glslocref	180, 185, 187, 188, 323, 324
\@gls@thislabel	58, 190, 200	\@glsminrange	163, 164, 325
\@gls@thislist	159	\@glsname	135
\@gls@thisloc	188	\@glsname@	135
\@gls@thisval	76	\@glsnavhypertarget	269
\@gls@title	43	\@glsnextpages	193
\@gls@tmp ...	15, 38, 51, 73, 117, 182, 270, 271	\@glsnodesc	84, 85, 88
\@gls@tmpb	118–123, 169–171	\@glsnoname	84, 86, 88
\@gls@toc	44, 45	\@glsnonextpages	193
\@gls@type	176, 226, 239, 242, 244, 246, 248, 251, 253, 255, 318	\@glsnumberformat	111, 114, 160, 180, 188, 323, 324
\@gls@updatechecked	117, 118, 170, 171	\@glsopenfile	173, 181
\@gls@usetranslator	26, 27, 37	\@glsorder	174, 175
\@gls@value	72, 155	\@glspl	128
\@gls@warnonglossdefined	21, 191	\@glspl@	96, 99, 128, 265–267
\@gls@warnonthe glossdefined	21, 210	\@glsplural	133
\@gls@write@entrycounts	97	\@glsplural@	133
\@gls@writedef	74	\@glsreset	92, 96
\@gls@writeisthook	167, 169	\@glssee	89, 190
\@gls@xdy@locationlist	164	\@glsshowtarget	5, 6, 124
\@gls@xdycheckbackslash	117	\@glssymbol	137
\@gls@xdycheckquote	117	\@glssymbol@	137
\@gls@xref	189	\@glssymbolplural	138
\@gls@Alphacompositor	41, 51, 325	\@glssymbolplural@	138
\@gls@Hlocref	185, 187	\@glstarget	125, 210, 270

\@glstext	131	\@onlypreamble ..	64, 74, 83, 97, 100, 176, 179
\@glstext@	131	\@onlypremakeg	40, 41, 47, 48, 52, 64, 169
\@glunset	93, 96	\@org@glossaryentrynumbers	193, 194
\@gluseri	139	\@org@gl@assign@descplural	
\@gluseri@	139	239, 247, 248, 250, 252, 373, 376, 377
\@gluserii	139, 140	\@org@gl@assign@firstpl	
\@gluserii@	140	239, 241, 243, 245, 247, 250, 252, 373–377
\@gluseriii	140	\@org@gl@assign@plural	
\@gluseriii@	140	239, 241, 243, 245, 247, 250, 252, 373–377
\@gluseriv	141	\@org@gl@assign@symbolplural	
\@gluseriv@	141	239, 241, 243, 245, 250, 252, 374, 375, 377
\@gluserv	142	\@org@gl@numberformat	159
\@gluserv@	142	\@org@newglossaryentryprehook	83
\@gluservi	143	\@outputpage	287, 288
\@gluservi@	143	\@p@glossarysection	43
\@glswidestname	318–320, 336	\@pgls	264
\@glswritefiles	30	\@pgls@	264
\@glxtr@doaccsupp	342	\@pglspl	265
\@gobble	5, 13–15, 75, 92, 117, 161, 162, 165, 166, 177, 322, 326, 327	\@pglspl@	265
\@idxitem	312	\@plus	274, 294, 312
\@ifclassloaded	4, 11, 43	\@print@glossary	192
\@ifnextchar	64, 112	\@print@noidx@glossary	192
\@ifpackageloaded		\@printgloss@setsort	176, 177, 193
.....	4, 8, 26, 27, 38, 54, 92, 158, 169, 342	\@printglossary	192
\@ifstar	63, 76, 77, 112, 162, 219	\@roman	50, 324
\@ifundefined	37, 270, 277, 288, 299, 306, 319, 320, 336, 350	\@secondofthree	
\@ignored@glossaries	65	112, 125, 127, 129, 144, 146, 148, 150, 360
\@input@	194	\@secondoftwo .	23, 26, 27, 38, 75, 76, 124– 127, 130, 144, 145, 147–149, 360–362, 380
\@istfilename	174, 175	\@set@glo@numformat	188, 324
\@makecol	287, 288	\@sglsaddkey	77
\@makeglossary	175	\@sglsaddstoragekey	76, 77
\@minus	274, 294, 312	\@thirdofthree	
\@mkboth	43, 44	112, 127, 130, 145, 147, 149, 150, 360
\@newglossary	62, 63	\@this@attr	165, 166
\@newglossaryentry@defcounters ..	89, 95	\@this@childlabel	196, 197
\@newglossaryentryposthook		\@this@counter	48
.....	77, 78, 89, 262, 345	\@this@ctr	165
\@newglossaryentryprehook		\@this@key	76
.....	77, 78, 83, 85, 262, 345	\@this@label	196
\@nil	18, 89, 116–118, 152, 170, 171, 188, 189, 202, 203, 217, 218, 323, 324	\@thiscs	36
\@nnil	18, 190	\@tmp	50, 325
\@no@makeglossaries	176, 177	\@use@option	34
\@no@post@desc	329	\@warn@nomakeglossaries	174, 195
\@nopostdesc	193	\@wrglossary@pageformat	184
\@onelevel@sanitize	22, 50, 75, 91, 117, 162, 163, 166, 189, 191, 202, 325, 326	\@wrglossarynumberhook	184, 187
		\@xdy@main@language	29, 173, 195
		\@xdy@attributelist	48, 165
		\@xdy@attributes	48, 164, 322, 324

<code>\@xdycounters</code>	47, 48, 165	<code>\acronymentry</code>	224, 227, 228, 230, 231, 233–236, 364–366, 369–372
<code>\@xdycrossrefhook</code>	165	<code>\acronymfont</code>	107, 144–147, 152, 158, 223, 225, 227–231, 233–236, 240–242, 244, 246–251, 357, 358, 360–362, 364–366, 368–372, 374–376
<code>\@xdylanguage</code>	195	<code>\acronymname</code>	17, 39
<code>\@xdylettergroups</code>	55, 167, 327	<code>\acronymsort</code>	224, 227, 228, 230, 231, 233, 234, 236, 364–366, 369, 370, 372
<code>\@xdylocationclassorder</code>	52, 165, 326	<code>\acronymtype</code>	17, 224, 226, 238, 239, 241– 243, 245, 247, 248, 250–252, 254, 373–376
<code>\@xdylocref</code>	48, 167, 322, 326	<code>\acrpluralsuffix</code>	224, 227–229, 233–236, 239–243, 245–248, 250–252, 254, 364, 365, 369–371, 373–377
<code>\@xdynumbergrouporder</code>	54, 162, 163	<code>\Acrshort</code>	237
<code>\@xdyrequiredstyles</code>	53, 163, 324	<code>\acrshort</code>	237
<code>\@xdysortrules</code>	53, 167, 327	<code>\Acrshorttpl</code>	237
<code>\@xdystyle</code>	163, 164, 324	<code>\acrshorttpl</code>	237
<code>\@xdyuseralphabets</code>	49, 164, 324	<code>\addcontentsline</code>	46
<code>\@xdyuserlocationdefs</code>	51, 52, 165, 323, 325	<code>\addglossarytocaptions</code>	38
<code>\@xdyuserlocationnames</code>	52, 323	<code>\addtolength</code>	320, 336
<code>\@xfor@nextelement</code>	190	<code>\advance</code>	14, 15, 86, 115, 287
<code>\</code>	90, 117, 162, 168, 217, 218, 327, 328, 330–332, 340, 341	<code>\AE</code>	23
<code>\{</code>	75, 161, 168, 322, 327, 328	<code>\ae</code>	23
<code>\}</code>	75, 161, 168, 322, 328	<code>amsgen package</code>	4, 111
<code>^</code>	23	<code>amsmath package</code>	92
<code>'</code>	23	<code>\andname</code>	191
<code> </code>	118, 120, 171	<code>\AnyTrackedLanguages</code>	38, 380
<code>~</code>	23	<code>\appto</code>	19, 69, 77, 78, 262, 345
A			
<code>\a</code>	23	<code>array package</code>	284, 288, 306
<code>\AA</code>	23	<code>article class</code>	188
<code>\aa</code>	23	<code>\AtBeginDocument</code>	17, 54, 74, 92, 160, 177
<code>accsupp package</code>	342	<code>\AtEndDocument</code>	30, 74, 97, 176, 180, 195, 270
<code>\accsuppglossaryentryfield</code>	342	B	
<code>\accsuppglossarysubentryfield</code>	343	<code>\b</code>	23
<code>\acrfootnote</code>	240, 246	<code>babel package</code>	26, 36, 38, 53
<code>\Acrfull</code>	237	<code>\begin</code>	166, 201, 274, 278–283, 286–312, 326
<code>\acrfull</code>	237	<code>\BeginAccSupp</code>	348
<code>\ACRfullfmt</code>	222, 225, 233, 235, 368, 370	<code>\begingroup</code>	5, 182, 186, 218
<code>\Acrfullfmt</code>	221, 225, 233, 235, 368, 370	<code>\bfseries</code>	279–282, 284, 285, 289–291, 293, 301–306, 308–312
<code>\acrfullfmt</code>	221, 225, 233, 234, 368, 370	<code>\bgroup</code>	23, 83, 158, 193, 196
<code>\acrfullformat</code>	158, 221, 238, 239, 254	<code>bib2gls</code>	186
<code>\Acrfullpl</code>	238	<code>booktabs package</code>	283–286
<code>\acrfullpl</code>	237	<code>\boolean</code>	253
<code>\ACRfullplfmt</code>	223, 225, 233, 235, 368, 370	<code>\boolfalse</code>	31
<code>\Acrfullplfmt</code>	223, 225, 233, 235, 368, 370	<code>\booltrue</code>	31
<code>\acrfullplfmt</code>	222, 225, 233, 235, 368, 370	<code>\bottomrule</code>	284, 285
<code>\acrlinkfootnote</code>	240		
<code>\acrlinkfullformat</code>	221–223		
<code>\Acrlong</code>	237		
<code>\acrlong</code>	237		
<code>\Acrlongpl</code>	237		
<code>\acrlongpl</code>	237		
<code>\acrnameformat</code>	245, 375		

<code>\box</code>	287	<code>\CurrentTrackedTag</code>	38, 380, 381
C		<code>\CustomAcronymFields</code>	254
<code>\c</code>	23	<code>\CustomNewAcronymDef</code>	255
<code>\c@equation</code>	115	D	
<code>\c@glossaryentry</code>	11	<code>\d</code>	23
<code>\c@glossarysubentry</code>	12	<code>datatool</code> package	198
<code>\c@page</code>	183, 184, 187	<code>\day</code>	163, 167, 324, 327
<code>\catcode</code>	74	<code>\DeclareAcronymList</code>	17, 19, 224, 226, 239, 241, 243, 246, 248, 251, 253, 255
<code>\cGls</code>	98	<code>\DeclareListParser</code>	176
<code>\cgl</code> s	98	<code>\DeclareOption</code>	9, 262, 342
<code>\cGlsformat</code>	96	<code>\DeclareOptionX</code>	9
<code>\cgl</code> sformat	96	<code>\DeclareRobustCommand</code>	39, 190, 191, 248, 348–350
<code>\cGlspl</code>	99	<code>\def</code>	8, 9, 11–16, 18, 22–24, 29, 31, 32, 34–36, 39, 40, 43, 46, 49–54, 58, 62–64, 66–70, 73, 81, 83– 88, 90, 92, 96–100, 110, 111, 114–124, 126–151, 159, 160, 163, 167, 169–171, 173–175, 177, 178, 181, 184, 185, 187– 190, 192, 193, 196, 200–202, 204, 205, 207, 214–219, 221–224, 239, 241, 243– 245, 247–252, 254, 262, 264–268, 271– 273, 298, 299, 318–321, 323, 324, 327, 329, 336, 337, 342–345, 359–362, 373–377
<code>\cgl</code> spl	99	<code>\def@gl</code> s@xdycheckbackslash	123, 124
<code>\cGlsplformat</code>	97	<code>\DefaultNewAcronymDef</code>	240
<code>\cgl</code> splformat	96	<code>\defgl</code> sentryfmt	63, 65, 110, 226, 238, 240, 242, 244, 246, 249, 251, 254
<code>\char</code>	214	<code>\define@bool</code> key	7, 9, 11, 12, 17, 24, 27–31, 111, 207
<code>classicthesis</code> package	8	<code>\define@choice</code> key	5–8, 13, 25, 26, 29, 32, 68, 206, 207
<code>\cleardoublepage</code>	45	<code>\define@key</code>	8, 12, 19, 25, 29, 66– 70, 77, 78, 111, 160, 205, 207, 262, 343, 344
<code>\clearpage</code>	45	<code>\DefineAcronymSynonyms</code>	34, 238
<code>\closeout</code>	74, 167, 169, 173, 181	<code>\delim</code> N	166, 176, 204, 217, 326
<code>\compatglossarystyle</code>	329–341	<code>\delim</code> R	166, 217, 326
<code>\compatibleglossentry</code>	212	<code>\DescriptionDUANewAcronymDef</code>	244
<code>\compatiblesubglossentry</code>	212	<code>\DescriptionFootnoteNewAcronymDef</code> .	242
<code>\copy</code>	287, 288	<code>\description</code> name	39, 279–282, 284, 285, 289–291, 293, 301–306, 308–312
<code>\count@</code>	202	<code>\DescriptionNewAcronymDef</code>	246
<code>\csdef</code>	21, 77–79, 89, 90, 95, 96, 196, 197, 216, 226, 328	<code>\DH</code>	24
<code>\csedef</code>	97, 184	<code>\dh</code>	24
<code>\csgdef</code>	42, 62, 65, 96, 97, 191, 205	<code>\dimen@</code>	228, 287, 318
<code>\cslet</code>	69, 83, 90, 200	<code>\disable@keys</code>	34
<code>\csname</code> ...	13–16, 34, 36, 38, 39, 44, 45, 48, 50, 51, 54, 55, 58, 63, 64, 71, 72, 77–82, 85–91, 93, 94, 110, 114–116, 126–131, 144–151, 159, 160, 164–166, 170–173, 177, 180–182, 184, 188, 189, 192–195, 197, 205, 210, 212, 215, 216, 219, 256– 263, 270, 271, 318–320, 322–324, 336, 342, 343, 346, 347, 350, 360–362, 378, 379	<code>\do</code>	27, 34, 36, 47, 48, 55, 75, 76, 117, 159, 163–165,
<code>\csshow</code>	260		
<code>\csuse</code>	39, 42, 62, 72, 78, 79, 110, 173, 174, 197, 199, 201, 203, 204, 206, 207, 215, 226, 263, 329–341		
<code>\csxdef</code>	88, 97		
<code>\currentglossary</code>	11, 12, 42, 193		
<code>\currentgl</code> ssubentry	12, 208		
<code>\CurrentOption</code>	34, 262, 342		
<code>\CurrentTrackedLanguage</code>	38, 380, 381		

175, 176, 183, 190, 196, 226, 239, 242, 244, 246, 248, 251, 253, 255, 270, 271, 324	\endcsname 13–16, 34, 36, 38, 39, 44, 45, 48, 50, 51, 54, 55, 58, 63, 64, 71, 72, 77–82, 85–91, 93, 94, 110, 114–116, 126–131, 144–151, 159, 160, 164–166, 170–173, 177, 180–182, 184, 188, 189, 192–195, 197, 205, 210, 212, 215, 216, 219, 256– 263, 270, 271, 318–320, 322–324, 336, 342, 343, 346, 347, 350, 360–362, 378, 379
\do@glo@storeentry 13–15, 89	\endfoot . 278–280, 282, 284, 285, 289–291, 293
\do@gl@link@checkfirsthyper 113, 114, 126–131, 144–150, 360, 361	\endgroup 5, 183, 187, 218
\do@gl@xdycheckbackslash 117	\endhead . 278–280, 282, 284, 285, 289–291, 293
\do@gl@disablehyperinlist 114	\endtheglossary 5
\do@gl@haschildren 58	\entryname 39, 279–282, 284, 285, 289–291, 293, 301–306, 308–312
doc package 4, 5, 16	\equal 25, 35, 45, 115, 175, 214, 270
\doifglossarynoexistsordo 63	equation (counter) 115, 116
\dtl@ifsingle 214	etoolbox package 4
\dtl@insertinto 197	\expandafter 13–16, 22, 23, 34– 36, 38, 48, 50, 51, 53–56, 58, 63–65, 71, 72, 75–82, 85–89, 91–94, 110, 114–123, 152, 159, 161, 162, 165, 166, 169–173, 181–183, 185, 187, 188, 190, 193, 197, 202, 203, 210, 212, 216, 218, 219, 240, 246, 256–263, 270, 271, 318, 322–324, 326, 327, 342, 343, 346, 348, 372, 378, 379
\dtl@sortresult 198	\expandonce 72, 73, 117, 170, 171, 184, 198, 210, 212, 224, 238, 239, 241, 243, 245, 247, 250, 252, 342, 343
\dtl@compare 198	
\dtl@lcompare 198	
\DTL@lfinlist 65, 114	
\DTL@lfinint 214	
\dtl@letterindexcompare 198	
\DTL@substituteall 117	
\dtl@wordindexcompare 198	
\DUANewAcronymDef 253	
E	
\eappto 65, 90, 184	
\edef 15, 18, 35, 38, 46–53, 55, 58, 63, 65, 72, 74, 76, 80–82, 84, 85, 90, 91, 110, 114–123, 159–162, 167, 169–171, 173, 174, 176, 177, 181, 185, 187, 188, 191, 195–198, 202, 208, 214, 218, 219, 238, 241, 242, 245, 247, 250, 252, 269, 322, 323, 325, 327, 372–376	
\egroup 23, 83, 159, 194, 197	
\else 6, 11, 12, 15–18, 20, 22, 24, 25, 30, 32, 34, 35, 40, 41, 43–50, 52–55, 69, 71, 86, 87, 90–92, 96, 97, 113–115, 117–124, 126–131, 152, 162, 163, 166– 172, 174, 181–185, 187–190, 193, 202, 207, 209, 214, 217, 218, 228, 242, 243, 246, 248, 249, 251, 253, 255, 270, 275, 278, 280, 281, 284, 285, 287, 289, 291, 292, 300, 302, 304, 307, 309, 311, 314, 315, 317, 319, 320, 323–329, 334–337, 348	
\emph 190, 219	
\empty 218, 342	
\end 166, 201, 274, 278–283, 286–312, 326	
\end@doifinlist 46, 47	
\end@getprefix 189	
\end@gl@islistofacronyms 18	
\EndAccSupp 348	
	F
	\fi 5–8, 11– 18, 20, 22, 24, 25, 27, 30, 32, 34, 35, 39– 41, 43–55, 64, 69, 71, 85–88, 90–92, 96, 97, 113–124, 126–131, 153, 160, 162– 164, 166–174, 176, 181–185, 187–191, 193, 195, 202, 206–209, 215, 217, 218, 228, 238, 239, 242, 243, 246, 248, 249, 251, 253, 255, 261, 270, 275, 278, 280, 281, 284, 285, 287–289, 291, 292, 300, 302, 304, 307, 309, 311, 314, 315, 317– 320, 322–326, 328, 329, 334–337, 342, 348
	file types
	.aux 195
	.glo 90
	.ist 161, 172
	.toc 45
	.xdy 40
	glo 261
	\firstacronymfont ... 109, 227–229, 234, 240, 244, 246, 249, 359, 363–365, 369, 370

`\footnote` 234, 240, 370
`\FootnoteNewAcronymDef` 248
`\forallforallglossaries` 56, 180, 192, 318
`\forallforallglsentries` 94, 95, 97, 161
`\ForEachTrackedDialect` 38, 380, 381
`\forallforallglsentries` 56, 58, 90, 200, 318
`\forallforalllistcsloop` 196, 201
`\forallforalllistloop` 178, 179, 204

G

`garamondx` package 220
`\gdef` 15, 48, 63, 81, 86, 87, 182, 207, 270
`\Genacrfullformat`
 108, 225, 227, 228, 234, 359, 363, 364, 370
`\genacrfullformat` 108, 109,
 225, 227, 228, 234, 358, 359, 363, 364, 369
`\GenericAcronymFields` 224, 227, 228, 230,
 231, 233, 234, 236, 363–366, 368, 369, 372
`\Genplacrfullformat`
 108, 225, 227, 228, 234, 358, 364, 370
`\genplacrfullformat`
 108, 109, 225, 227, 228, 234, 358, 364, 370
`\glo@desc` 329
`\glo@do@compare` 198, 199
`\glo@grabfirst` 203
`\glo@label` 58, 90
`\glo@list` 90
`\glo@name` 210
`\glo@parent` 58
`\glo@type` 90
`\glo@value` 75
`\global` 14–16, 71, 74, 83,
 88, 93, 94, 182, 194, 202, 203, 208, 287, 288
`\glolinkprefix` 115, 159, 210
`\glosortentrieswarning` 20, 196
`glossaries` package 32,
 33, 53, 54, 163, 255, 262, 274, 322, 342
`glossaries-accsupp` package 90, 342
`glossaries-extra` package 74, 165, 342
`\GlossariesWarning` 5, 7,
 20, 21, 24, 25, 31, 32, 42, 46, 57, 61, 68,
 71, 98, 99, 110, 112, 158, 174, 176, 178–
 180, 183, 189, 193, 194, 212, 215, 322, 342
`\GlossariesWarningNoLine`
 5, 6, 20, 175, 177, 181, 195, 270
`\glossary` 31, 32, 323, 324
`glossary` package 1, 32, 219
`glossary` styles:
 `altlist` 275, 276, 330

`altlistgroup` 276, 330
`altlisthypergroup` 276, 330
`altlong4col` 282, 283, 291, 332
`altlong4col-booktabs` 285, 287
`altlong4colborder` 283, 332
`altlong4colheader` 283, 285, 332
`altlong4colheaderborder` 283, 332
`altlongragged4col` ... 286, 291–293, 333
`altlongragged4col-booktabs` 286
`altlongragged4colborder` 293, 333
`altlongragged4colheader` 292, 333
`altlongragged4colheaderborder` 293, 334
`altsuper4col` 305, 310, 341
`altsuper4colborder` 305, 341
`altsuper4colheader` 305, 341
`altsuper4colheaderborder` ... 305, 341
`altsuperragged4col` 310, 311, 339
`altsuperragged4colborder` ... 311, 339
`altsuperragged4colheader` ... 311, 339
`altsuperragged4colheaderborder` .
 311, 339
`alttree` 298, 313, 318, 320, 336
`alttreegroup` 321, 337
`alttreehypergroup` 321, 337
`index` 8, 294, 312–315, 334
`indexgroup` 314, 334
`indexhypergroup` 314, 334
`inline` 329
`list` 8, 10, 274–276, 329
`listdotted` 276, 277, 330
`listgroup` 275, 329
`listhypergroup` 275, 330
`long` 278, 279, 284, 288, 330, 332
`long-booktabs` 284, 286
`long3col` 279, 280, 284, 331
`long3col-booktabs` 284, 286
`long3colborder` 280, 331
`long3colheader` 280, 284, 331
`long3colheaderborder` 280, 331
`long4col` 281, 282, 285, 331
`long4col-booktabs` 285
`long4colborder` 282, 332
`long4colheader` 281, 285, 332
`long4colheaderborder` 282, 332
`longborder` 278, 331
`longheader` 278, 284, 331
`longheaderborder` 279, 331
`longragged` 286, 288–290
`longragged-booktabs` 286

longragged3col [286](#), [290](#), [291](#), [333](#)
longragged3col-booktabs [286](#)
longragged3colborder [291](#), [333](#)
longragged3colheader [291](#), [333](#)
longragged3colheaderborder . [291](#), [333](#)
longraggedborder [289](#), [332](#)
longraggedheader [289](#), [333](#)
longraggedheaderborder [290](#), [333](#)
mcolalmtree [298](#), [338](#)
mcolalmtreegroup [298](#), [338](#)
mcolalmtreehypergroup ... [298](#), [299](#), [338](#)
mcolindex [294](#), [337](#)
mcolindexgroup [294](#), [337](#)
mcolindexhypergroup [294](#), [295](#), [337](#)
mcoltree [295](#), [337](#)
mcoltreegroup [337](#)
mcoltreehypergroup [296](#), [337](#)
mcoltreenoname [297](#), [338](#)
mcoltreenonamegroup [297](#), [338](#)
mcoltreenonamehypergroup ... [297](#), [338](#)
sublistdotted [330](#)
super [300](#), [301](#), [308](#), [340](#)
super3col [301–303](#), [340](#)
super3colborder [302](#), [340](#)
super3colheader [302](#), [340](#)
super3colheaderborder [303](#), [340](#)
super4col [303–305](#), [341](#)
super4colborder [304](#), [341](#)
super4colheader [304](#), [341](#)
super4colheaderborder [304](#), [341](#)
superborder [300](#), [340](#)
superheader [301](#), [340](#)
superheaderborder [301](#), [340](#)
superragged [306](#), [308](#), [338](#)
superragged3col [308–310](#), [339](#)
superragged3colborder [309](#), [339](#)
superragged3colheader [309](#), [339](#)
superragged3colheaderborder [310](#), [339](#)
superraggedborder [307](#), [338](#)
superraggedheader [308](#), [338](#)
superraggedheaderborder [308](#), [339](#)
tree [295](#), [315](#), [316](#), [318](#), [334](#)
treegroup [296](#), [316](#), [335](#)
treehypergroup [316](#), [335](#)
treenoname [296](#), [313](#), [316](#), [317](#), [335](#)
treenonamegroup [317](#), [336](#)
treenonamehypergroup [317](#), [336](#)
glossary-hypernav package [161](#)
glossary-list package [8](#), [10](#), [274](#)
glossary-long package ... [10](#), [277](#), [291](#), [299](#), [300](#)
glossary-longragged package [288](#)
glossary-mcols package [293](#)
glossary-super package .. [10](#), [277](#), [299](#), [306](#), [310](#)
glossary-superragged package [306](#)
glossary-tree package [10](#), [312](#)
\glossaryentry [188](#), [189](#), [324](#)
glossaryentry (counter) [11](#), [12](#), [208](#), [209](#)
\glossaryentryfield
..... [210](#), [329–336](#), [338–341](#), [363](#)
\glossaryentrynumbers
..... [9](#), [166](#), [193](#), [194](#), [203](#), [207](#), [208](#), [326](#)
\glossaryheader
..... [166](#), [201](#), [272](#), [274–276](#), [278–](#)
[282](#), [284](#), [285](#), [288–294](#), [296–298](#), [300](#),
[301](#), [303](#), [307](#), [308](#), [310](#), [313–318](#), [321](#), [326](#)
\glossarymark [43](#)
\glossaryname [16](#), [38](#), [39](#)
\glossarypostamble [166](#), [201](#), [326](#)
\glossarypreamble [165](#), [201](#), [326](#)
\glossarysection [165](#), [201](#), [326](#)
glossarysubentry (counter) [12](#), [208](#), [209](#)
\glossarysubentryfield
..... [211](#), [329–336](#), [338–341](#), [363](#)
\glossarytitle .. [165](#), [192](#), [193](#), [201](#), [205](#), [326](#)
\glossarytoctitle [8](#), [16](#), [17](#),
[33](#), [36](#), [39](#), [43](#), [165](#), [192](#), [201](#), [205](#), [206](#), [326](#)
\glossentry [90](#), [194](#), [203](#), [212](#),
[272](#), [274–279](#), [281](#), [289](#), [290](#), [292](#), [300](#),
[302](#), [303](#), [307](#), [308](#), [310](#), [313](#), [315](#), [316](#), [319](#)
\Glossentrydesc [362](#)
\glossentrydesc
. [272–279](#), [281](#), [289](#), [290](#), [292](#), [300](#), [302](#),
[303](#), [307–310](#), [313–315](#), [317](#), [319](#), [320](#), [362](#)
\glossentryname
. [272–279](#), [281](#), [289](#), [290](#), [292](#), [300](#), [302](#),
[303](#), [307](#), [308](#), [310](#), [313–316](#), [319](#), [320](#), [362](#)
\Glossentrysymbol [363](#)
\glossentrysymbol [272](#), [273](#), [281](#),
[292](#), [303](#), [310](#), [313–315](#), [317](#), [319](#), [320](#), [363](#)
\Gls [98](#), [219](#), [238](#)
\gls [31](#), [98](#), [176](#), [209](#), [219](#), [238](#)
\gls@Alphpage [183](#), [187](#)
\gls@alphpage [183](#), [187](#)
\gls@arabicpage [183](#), [187](#)
\gls@assign@desc [83](#), [88](#)
\gls@assign@descplural
..... [239](#), [247](#), [248](#), [250](#), [252](#), [373](#), [376](#), [377](#)

<code>\gls@assign@field</code>	<code>\gls@orgarabic</code>	186, 187
..... 73, 77, 78, 83, 85, 87, 88, 262, 263	<code>\gls@orgnumber</code>	186, 187
<code>\gls@assign@firstpl</code>	<code>\gls@orgRoman</code>	186, 187
239, 241, 243, 245, 247, 250, 252, 373–377	<code>\gls@orgromannumeral</code>	186, 187
<code>\gls@assign@plural</code>	<code>\gls@orgthe</code>	186, 187
239, 241, 243, 245, 247, 250, 252, 373–377	<code>\gls@original@glossary</code>	32
<code>\gls@assign@symbolplural</code>	<code>\gls@original@makeglossary</code>	32
239, 241, 243, 245, 250, 252, 374, 375, 377	<code>\gls@protected@pagefmts</code>	117, 183, 184
<code>\gls@begindocdefs</code>	<code>\gls@Romanpage</code>	183, 187
74	<code>\gls@romanpage</code>	183, 187
<code>\gls@checkisacronymlist</code>	<code>\gls@save@numberlist</code>	9
113	<code>\gls@seen@index@nr</code>	7
<code>\gls@checkseeallowed</code>	<code>\gls@seen@index@val</code>	7
68, 74, 175, 177	<code>\gls@set@xr@key</code>	68
<code>\gls@checkseeallowed@preambleonly</code> ..	<code>\gls@suffixF</code>	41, 166, 168, 326, 328
74	<code>\gls@suffixFF</code>	41, 166–169, 326, 328
<code>\gls@codepage</code>	<code>\gls@text</code>	109
54, 173, 195	<code>\gls@the</code>	187
<code>\gls@debug@nr</code>	<code>\gls@thissty</code>	27
5, 32	<code>\gls@tmp</code>	181, 248, 249
<code>\gls@debug@val</code>	<code>\gls@tmplen</code>	124, 318–320, 336, 337
5, 32	<code>\gls@tr@set@acronym@toctitle</code>	17
<code>\gls@defdocnewglossaryentry</code>	<code>\gls@tr@set@main@toctitle</code>	16
75, 95	<code>\gls@tr@set@numbers@toctitle</code>	33
<code>\gls@defglossaryentry</code>	<code>\gls@tr@set@symbols@toctitle</code>	33
73–75, 83	<code>\gls@translate@nr</code>	26
<code>\gls@disablepagerefexpansion</code> ..	<code>\gls@translate@val</code>	26
182, 187	<code>\gls@wrglossary</code>	182
<code>\gls@do@addxdyattribute</code>	<code>\gls@xdystring</code>	117
48	<code>\gls@xindy@glsnumbersfalse</code>	30
<code>\gls@docclearpage</code>	<code>\gls@xindy@glsnumberstrue</code>	29
45	<code>\gls@xr@key</code>	6, 7, 68
<code>\gls@dosubst</code>	<code>\gls@sacccsupp</code>	348
117	<code>\gls@acronymtrue</code>	17
<code>\gls@dotocitle</code>	<code>\gls@acrpluralsuffix</code>	
193, 205 37, 220, 229, 233, 234, 236, 240	
<code>\gls@end@sanitizesort</code>	<code>\gls@acrshortcutsfalse</code>	34
22, 23	<code>\gls@acrshortcutstrue</code>	34
<code>\gls@endcheck</code>	<code>\gls@sacspace</code>	227, 228, 230
72, 73	<code>\gls@sadd</code>	31, 161
<code>\gls@glossary</code>	<code>\gls@sadd options</code>	
31, 32, 188, 189	counter	160
<code>\gls@gobbleopt</code>	format	160, 216
64	<code>\gls@saddall options</code>	
<code>\gls@grplabel</code>	types	160
269	<code>\GlsAddXdyAttribute</code>	47, 49, 322, 323
<code>\gls@hypergrouprrun</code>	<code>\GlsAddXdyCounters</code>	47, 48, 64
270	<code>\gls@automakefalse</code>	30
<code>\gls@ifnotmeasuring</code>	<code>\gls@autoprefix</code>	8, 206
92, 93		
<code>\gls@inlinepostchild</code>		
272, 273, 329		
<code>\gls@inlinessep</code>		
272, 329		
<code>\gls@inlinesubsep</code>		
272, 273, 329		
<code>\gls@islistofacronyms</code>		
18		
<code>\gls@istfilebase</code>		
39, 40, 173		
<code>\gls@label</code>		
219		
<code>\gls@level</code>		
86, 87, 203		
<code>\gls@noidxglossary</code>		
177		
<code>\gls@nonumberlist@nr</code>		
68		
<code>\gls@nonumberlist@val</code>		
68		
<code>\gls@nosetquote</code>		
84, 167, 169, 172		
<code>\gls@number</code>		
187		
<code>\gls@numberedsection@nr</code>		
8, 206		
<code>\gls@numberedsection@val</code>		
8, 206		
<code>\gls@numberpage</code>		
183, 187		
<code>\gls@org@glossaryentryfield</code>		
194		
<code>\gls@org@glossarysubentryfield</code>		
194		
<code>\gls@org@insert</code>		
244, 246, 249		
<code>\gls@orgAlpha</code>		
186, 187		
<code>\gls@orgalph</code>		
186, 187		

<code>\glscapscase</code>	101, 102, 105–108, 126–130, 144–150, 232, 244, 249, 350, 352, 354, 355, 357, 358, 360–362, 367
<code>\glsclearpage</code>	44
<code>\glsclosebrace</code>	51, 52, 166, 167, 326, 327
<code>\glscompositor</code>	40, 41, 51, 168, 325, 328
<code>\glscounter</code>	19, 35, 47, 64, 87, 115, 322
<code>\glscurrententrylabel</code>	191, 194
<code>\glscurrentfieldvalue</code>	60, 61
<code>\glscustomtext</code>	101, 104, 106, 107, 109, 126–130, 144–151, 232, 233, 240, 244, 246, 247, 249, 350, 354, 356, 357, 359–362, 367, 368
<code>\GlsDeclareNoHyperList</code>	19
<code>\glstypedef</code>	16, 42, 54, 55, 62, 63, 85, 100, 110, 181, 192
<code>\glstypedefmain</code>	16, 65
<code>\glstypedefdescriptionaccessdisplay</code>	352–354, 362, 363
<code>\glstypedefdescriptionpluralaccessdisplay</code>	350–352
<code>\glstypedefwidth</code>	278–280, 282, 283, 286–293, 300–303, 305–312
<code>\glstypedefdetoklabel</code> .	56–61, 69, 75, 80–84, 90, 93–97, 114, 151, 152, 159, 160, 177–179, 185, 187, 194, 197, 198, 202, 203, 205, 208–210, 256–260, 318, 342, 343, 378, 379
<code>\glstypedefdisplay</code>	101, 110
<code>\glstypedefdisplayfirst</code>	101, 110
<code>\glstypedefdisplaynumberlist</code>	178
<code>\glstypedefdohyperlink</code>	124, 125
<code>\glstypedefdohypertarget</code>	125
<code>\glstypedefdoifexists</code>	58, 60, 80–82, 92, 93, 126–131, 144–150, 158, 160, 178, 179, 264–268, 359–363
<code>\glstypedefdoifexistsordo</code>	113, 151
<code>\glstypedefdoifexistsorwarn</code>	205, 210, 211
<code>\glstypedefdoifnoexists</code>	73, 83
<code>\glstypedefdonohyperlink</code>	115, 124, 125
<code>\glstypedefdosanitizesort</code>	13
<code>\glstypedefentryaccess</code>	348
<code>\glstypedefentrycounter</code>	215, 218
<code>\glstypedefentrycounterfalse</code>	12
<code>\glstypedefentrycounterlabel</code>	209
<code>\GlsEntryCounterLabelPrefix</code> ...	208, 209
<code>\glstypedefentrycountertrue</code>	12
<code>\glstypedefentrycurrcount</code>	95, 97
<code>\Glsentrydesc</code>	136, 211, 362
<code>\glstypedefentrydesc</code> 103, 104, 136, 211, 352–354, 362	
<code>\glstypedefentrydescaccess</code>	349
<code>\Glsentrydescplural</code>	137
<code>\glstypedefentrydescplural</code>	101, 102, 136, 137, 350–352
<code>\glstypedefentrydescpluralaccess</code>	349
<code>\Glsentryfirst</code> ...	99, 103, 106, 133, 353, 356
<code>\glstypedefentryfirst</code>	98, 103, 104, 106, 132, 133, 352, 353, 356
<code>\glstypedefentryfirstaccess</code>	349
<code>\Glsentryfirstplural</code>	100, 102, 105, 134, 351, 355
<code>\glstypedefentryfirstplural</code>	99, 101, 102, 105, 134, 350, 352, 354, 355
<code>\glstypedefentryfirstpluralaccess</code>	349
<code>\glstypedefentryfmt</code>	63, 65
<code>\Glsentryfull</code>	225, 233, 235, 369, 371
<code>\glstypedefentryfull</code>	225, 233, 235, 368, 371
<code>\Glsentryfullpl</code>	225, 233, 235, 369, 371
<code>\glstypedefentryfullpl</code>	225, 233, 235, 369, 371
<code>\glstypedefentryitem</code>	272, 274–279, 281, 289, 290, 292, 300, 302, 303, 307, 308, 310, 313, 315, 316, 319, 329–336, 338–341
<code>\Glsentrylong</code>	99, 148, 152, 158, 227, 232, 233, 359, 361, 364, 367–369
<code>\glstypedefentrylong</code>	98, 109, 147, 149, 152, 158, 227, 228, 230–236, 247, 359, 361–372
<code>\glstypedefentrylongaccess</code>	349
<code>\Glsentrylongpl</code>	100, 150, 158, 227, 228, 232, 233, 359, 364, 367–369
<code>\glstypedefentrylongpl</code>	99, 109, 149, 151, 158, 227, 228, 232–235, 247, 254, 359, 364, 365, 367–371
<code>\glstypedefentrylongpluralaccess</code>	350
<code>\Glsentryname</code>	135, 211, 362
<code>\glstypedefentryname</code>	135, 318, 362
<code>\glstypedefentrynumberlist</code>	158, 178
<code>\Glsentryplural</code>	102, 105, 133, 351, 355
<code>\glstypedefentryplural</code>	101, 102, 105, 133, 134, 350, 351, 354, 355
<code>\glstypedefentrypluralaccess</code>	349
<code>\Glsentryprefix</code>	266
<code>\glstypedefentryprefix</code>	264, 267
<code>\Glsentryprefixfirst</code>	266
<code>\glstypedefentryprefixfirst</code>	265, 267
<code>\Glsentryprefixfirstplural</code>	267
<code>\glstypedefentryprefixfirstplural</code>	265, 268
<code>\Glsentryprefixplural</code>	266
<code>\glstypedefentryprefixplural</code>	265, 268
<code>\glstypedefentryprevcount</code>	95–97

<code>\Glsentryshort</code>	107, 144, 152, 228, 234, 235, 358–360, 364, 370, 371
<code>\glsentryshort</code>	107, 109, 144, 145, 152, 158, 225, 227, 228, 230, 231, 233–236, 357–360, 363–366, 368–372
<code>\glsentryshortaccess</code>	349
<code>\Glsentryshortpl</code>	107, 146, 228, 234, 235, 357, 364, 370, 371
<code>\glsentryshortpl</code>	107, 109, 146, 147, 158, 227, 228, 233–235, 254, 357, 359, 364, 368–371
<code>\glsentryshortpluralaccess</code>	349
<code>\Glsentrysymbol</code>	137, 211, 363
<code>\glsentrysymbol</code>	103, 104, 137, 138, 211, 240, 244, 249, 352–354, 363
<code>\glsentrysymbolaccess</code>	349
<code>\Glsentrysymbolplural</code>	138
<code>\glsentrysymbolplural</code>	101, 102, 138, 139, 240, 244, 249, 350–352
<code>\glsentrysymbolpluralaccess</code>	349
<code>\Glsentrytext</code>	103, 106, 132, 352, 356
<code>\glsentrytext</code>	103, 104, 106, 131, 132, 159, 191, 352, 353, 355, 356
<code>\glsentrytextaccess</code>	348
<code>\glsentrytype</code>	85
<code>\Glsentryuseri</code>	139
<code>\glsentryuseri</code>	139
<code>\Glsentryuserii</code>	140
<code>\glsentryuserii</code>	140
<code>\Glsentryuseriii</code>	141
<code>\glsentryuseriii</code>	140, 141
<code>\Glsentryuseriv</code>	141
<code>\glsentryuseriv</code>	141, 142
<code>\Glsentryuserv</code>	142
<code>\glsentryuserv</code>	142, 143
<code>\Glsentryuservi</code>	143
<code>\glsentryuservi</code>	143
<code>\glsesclocationfalse</code>	177
<code>\glsesclocationstrue</code>	9
<code>\glsfieldfetch</code>	155
<code>\glsfirstaccessdisplay</code>	352, 353, 356
<code>\glsfirstpluralaccessdisplay</code>	350, 351, 354, 355
<code>\glsfirstpluralacessdisplay</code>	355
<code>\glsgenacfmt</code>	227, 228, 234, 363, 364, 369
<code>\glsgenentryfmt</code>	227, 228, 233, 234, 238, 240, 242, 244, 246, 249, 251, 254, 363, 364, 368, 369
<code>\glsgetgrouptitle</code>	271, 275, 276, 294–299, 314–317, 321
<code>\gls glossarymark</code>	43
<code>\gls groupheading</code>	167, 203, 272, 274–276, 278, 279, 281, 289, 290, 292, 294–301, 303, 307, 308, 310, 313–318, 320, 321, 327
<code>\gls groupskip</code>	166, 167, 203, 273, 275, 278, 280, 281, 284, 285, 289–292, 300, 302, 304, 307, 309, 311, 314, 315, 317, 320, 326
<code>\gls hyperfirstfalse</code>	234, 369
<code>\gls hyperfirsttrue</code>	28
<code>\gls hyperlink</code>	191
<code>\gls hypernavsep</code>	271
<code>\gls hypernumber</code>	42, 218, 219
<code>\gls ifhyperon</code>	112
<code>\gls IfListOfAcronyms</code>	18, 19
<code>\gls ifplural</code>	101, 104, 107, 108, 126–130, 144–150, 232, 240, 244, 247, 249, 350, 354, 357, 358, 360–362, 367
<code>\gls ifusetranslator</code>	26, 27, 38, 39, 380
<code>\gls indexonlyfirstfalse</code>	27
<code>\gls inlinedescformat</code>	272, 329
<code>\gls inlinedopostchild</code>	272, 329
<code>\gls inlineemptydescformat</code>	272, 329
<code>\gls inlinenameformat</code>	272, 329
<code>\gls inlineparentchildseparator</code>	272, 329
<code>\gls inlinepostchild</code>	272, 329
<code>\gls inlineseparator</code>	272, 329
<code>\gls inlinesubdescformat</code>	273, 329
<code>\gls inlinesubnameformat</code>	272, 329
<code>\gls inlinesubseparator</code>	273, 329
<code>\gls insert</code>	101–108, 126–130, 144–150, 232, 233, 240, 244, 246, 247, 249, 350–362, 367, 368
<code>\gls keylisttok</code>	224, 239, 241–248, 250–255, 372–377
<code>\gls label</code>	84, 101–108, 113–115, 144–150, 227, 228, 232, 234, 240, 244, 246, 247, 249, 350–359, 363, 364, 367–369
<code>\gls labeltok</code>	224, 238, 240–242, 244–248, 250–255, 373–376
<code>\gls link</code>	225, 233–235, 240, 368, 370
<code>\gls link options</code>	111, 125, 261
counter	111, 125, 216
format	111, 113, 114, 125
hyper	111
local	111
<code>\gls linkcheckfirsthyperhook</code>	113
<code>\gls linkpostsetkeys</code>	114

<code>\glslinkvar</code>	112	<code>\glspatchLToutput</code>	284–286
<code>\glslistdottedwidth</code>	276, 277, 330	<code>\glspenaltygroupskip</code>	284, 285
<code>\glslistgroupheaderfmt</code>	275, 276	<code>\glspencentchar</code>	75, 166, 167
<code>\glslistnavigationitem</code>	275, 276	<code>\Glspl</code>	99, 238
<code>\glslocalreset</code>	94	<code>\glspl</code>	99, 238
<code>\glslocalunset</code>	95, 126–131	<code>\glsppluralaccessdisplay</code>	350, 351, 354, 355
<code>\glslongaccessdisplay</code>	359, 361–373	<code>\glsppluralsuffix</code>	
<code>\glslongkey</code>	377	37, 87, 227–229, 364, 365, 369–371
<code>\glslongpluralaccessdisplay</code>		<code>\glspostdescription</code>	
.....	359, 364, 365, 367–371, 373	39, 273–276, 278, 289, 300, 307, 313–
<code>\glslongpluralkey</code>	377	315, 317, 319, 320, 329–332, 334–338, 340
<code>\glslongtok</code>		<code>\glspostinline</code>	272
.....	224, 225, 227, 228, 233, 234, 238–	<code>\glspostlinkhook</code>	
.....	248, 250–255, 363, 364, 368, 369, 372–377	113, 126–131, 144–151, 360–362
<code>\glsLTpenaltycheck</code>	287, 288	<code>\glsprestandardsort</code>	13
<code>\glsmcols</code>	294–299	<code>\glsreset</code>	94
<code>\glsnameaccessdisplay</code>	362, 363	<code>\glsresetentrycounter</code>	208
<code>\glsnamefont</code>	210, 211, 342, 343, 362	<code>\glsresetentrylist</code>	166, 201, 207, 326
<code>\glsnavhyperlink</code>	271	<code>\glsresetsubentrycounter</code> ...	209, 272, 329
<code>\glsnavhyperlinkname</code>	269, 270	<code>\glssanitizesortfalse</code>	25
<code>\glsnavhypertarget</code>		<code>\glssanitizesorttrue</code>	24, 25
.....	275, 276, 295–299, 315–317, 321	<code>\glssavenumberlistfalse</code>	9
<code>\glsnavigation</code>		<code>\glssavewritesfalse</code>	30
.....	275, 276, 294–299, 314, 316, 317, 321	<code>\glseeformat</code>	165, 177, 179, 326
<code>\glsnextpages</code>	9, 69, 193	<code>\glseeitem</code>	190
<code>\glsnogroupskipfalse</code>	11	<code>\glseeitemformat</code>	191
<code>\glsnoidxdisplayloc</code>	179, 180	<code>\glseeelastsep</code>	190
<code>\glsnoidxdisplayloclisthandler</code> ...	178	<code>\glseeelist</code>	190
<code>\glsnoidxloclist</code>	178, 203	<code>\glseeesep</code>	190
<code>\glsnoidxloclisthandler</code>	204	<code>\glssetexpandfield</code>	21, 24, 25
<code>\glsnoidxnumberlistloophandler</code> ...	179	<code>\glssetnoexpandfield</code>	21, 22, 24, 25
<code>\glsnoidxstripaccents</code>	23	<code>\GlsSetQuote</code>	84, 167
<code>\glsnomakeindexwarning</code>	169	<code>\glssettocitle</code>	39, 193
<code>\glsnonextpages</code>	68, 193	<code>\glsshortaccessdisplay</code>	
<code>\glsnopostdotfalse</code>	11	357–360, 363–366, 368–373
<code>\glsnoxindywarning</code> 41, 47–49, 52–54, 162, 163		<code>\glsshortkey</code>	377
<code>\glsnumberformat</code>	159	<code>\glsshortpluralaccessdisplay</code>	
<code>\glsnumberlistloop</code>	179	357, 359, 364, 368–371, 373
<code>\glsnumbersgroupname</code>	33, 39, 214	<code>\glsshortpluralkey</code>	377
<code>\glsnumlistlastsep</code>	159, 178	<code>\glsshorttok</code> 224, 238–248, 250–255, 373–377	
<code>\glsnumlistparser</code>	159, 176	<code>\glsshowtarget</code>	6
<code>\glsnumlistsep</code>	159, 178	<code>\glssortnumberfmt</code>	14, 15
<code>\glsopenbrace</code>	51, 52, 166, 167, 326, 327	<code>\glsspace</code>	221
<code>\glsorder</code>	29, 173–175, 199	<code>\glssstepentry</code>	209
<code>\glsorg@endtheglossary</code>	5	<code>\glssstepsubentry</code>	209
<code>\glsorg@PrintChanges</code>	5	<code>\glsssubentrycounterfalse</code>	12
<code>\glsorg@theglossary</code>	5	<code>\glsssubentrycounterlabel</code>	209
<code>\glspagelistwidth</code> 279, 280, 282, 283, 286,			
287, 290–293, 301–303, 305, 306, 308–312			

<code>\glssubentryitem</code>	273, 274, 276–279, 281, 289, 290, 292, 300, 302, 303, 307, 309, 310, 314, 315, 317, 319, 329–336, 338–341
<code>\glssymbolaccessdisplay</code>	352–354, 363
<code>\glssymbolpluralaccessdisplay</code>	350–352
<code>\glssymbolsgroupname</code>	33, 39, 214
<code>\glstarget</code>	212, 273–279, 281, 289, 290, 292, 300, 302, 303, 307–310, 313–317, 319, 320, 329–341
<code>\glstextaccessdisplay</code>	352, 353, 355, 356
<code>\glstextformat</code>	113, 115
<code>\glstextup</code>	37, 371
<code>\glstildechar</code>	48, 166, 167
<code>\glstranslatefalse</code>	26, 27
<code>\glstranslatetrue</code>	26, 27
<code>\glstreechildpredesc</code>	314, 315
<code>\glstreegroupheaderfmt</code>	294–299, 314, 316, 317, 321
<code>\glstreeindent</code>	315, 317, 319, 320, 335–337
<code>\glstreeitem</code>	294, 295, 313
<code>\glstreenamebox</code>	319, 320
<code>\glstreenamefmt</code>	312–316, 318–320
<code>\glstreenavigationfmt</code>	294–299, 314, 316, 317, 321
<code>\glstreepredesc</code>	313, 315, 317
<code>\glstreesubitem</code>	294, 313
<code>\glstreesubsubitem</code>	294, 313
<code>\glstype</code>	113, 114, 126–131, 144–151, 360–362
<code>\glsucmarkfalse</code>	11
<code>\glsucmarktrue</code>	11
<code>\glsunset</code>	92, 94, 96, 97, 126–131
<code>\glsupacrpluralsuffix</code>	229, 235, 242, 246, 248, 251
<code>\GlsUseAcrEntryDisplayStyle</code>	226, 229–231, 234–236, 365, 366, 369, 371, 372
<code>\GlsUseAcrStyleDefs</code>	226, 229–231, 234–236, 365, 366, 369, 371, 372
<code>\glswrallowprimitivemodstrue</code>	185
<code>\glswrite</code>	163–169, 174, 180, 181, 324–328
<code>\glswritedefhook</code>	75
<code>\glswriteentry</code>	183
<code>\glswritefiles</code>	30, 180
<code>\glsxindyfalse</code>	29
<code>\glsxindytrue</code>	29, 30
H	
<code>\H</code>	23
<code>\hangindent</code>	298, 299, 312, 315–317, 319–321, 334–337
<code>\hbox</code>	92, 276, 277, 330
<code>\hfill</code>	276, 277, 330
<code>\hline</code>	278–280, 282, 289–291, 293, 300–312
<code>\hsize</code>	277, 288, 299, 306
<code>\hspace</code>	313
<code>\hss</code>	276, 277, 330
<code>\ht</code>	287
<code>\hyperdef</code>	35
<code>\hyperlink</code>	111, 124, 218
<code>hyperref package</code>	188, 191, 216, 217, 261
<code>\hypertarget</code>	124
I	
<code>\IeC</code>	23
<code>\if</code>	117, 188, 323
<code>\if@endfor</code>	270
<code>\if@gl@debug</code>	5, 20, 182
<code>\if@gl@docloaded</code>	4, 16, 32
<code>\if@gl@isacronymlist</code>	113
<code>\if@openright</code>	45
<code>\ifbool</code>	17, 28, 31, 57, 101–103
<code>\ifboolexpr</code>	38, 62, 214
<code>\ifcase</code>	5, 7, 8, 26, 32, 68, 206, 314, 334
<code>\ifcsdef</code>	26, 39, 45, 71, 72, 78–82, 110, 182, 196, 197, 199, 201, 215, 226
<code>\ifcseempty</code>	58, 59, 264
<code>\ifcsequal</code>	59
<code>\ifcsstrequal</code>	82
<code>\ifcsstring</code>	82
<code>\ifcsundef</code>	7, 14, 29, 35, 38, 42, 43, 45, 56, 63, 65, 67, 85, 87, 88, 95, 96, 111, 115, 116, 124, 173, 181, 191, 195, 197, 205, 209, 214–217, 219, 225, 271, 328
<code>\ifdef</code>	60, 69, 74, 75, 92, 111, 151, 155, 178, 179, 220, 312, 313
<code>\ifdefempty</code>	19, 44, 45, 55, 59–61, 65, 101, 104, 107, 176, 202, 203, 224, 226, 232, 240, 244, 246, 248, 249, 350, 354, 357, 367
<code>\ifdefequal</code>	58–61, 71–73, 76, 86, 90, 203
<code>\ifdefstrequal</code>	82
<code>\ifdefstring</code>	10, 38, 62, 173–175, 199, 200, 202, 204
<code>\ifdefvoid</code>	22, 23, 89, 203
<code>\ifdim</code>	228, 287, 318
<code>\iffalse</code>	88, 94
<code>\IfFileExists</code>	10, 26, 27, 194
<code>\ifglossaryexists</code>	42, 54, 58, 172–174, 193
<code>\ifgl@sanitize@description</code>	24
<code>\ifgl@sanitize@name</code>	24

<code>\p@gl@s@hyp@opt</code>	112	<code>\PackageError</code>	6, 7, 16, 30, 36, 47, 54, 57, 58, 62, 67, 70, 71, 77–82, 85, 86, 95, 111, 151, 172, 173, 175, 178–180, 199–201, 205, 207, 215, 216, 225, 226, 242, 243, 248, 251, 328, 350
package options:		<code>\PackageInfo</code>	5, 6, 173, 182
acronym	16, 17, 36, 192, 220	<code>\PackageWarning</code>	5, 6, 20
true	17	<code>\PackageWarningNoLine</code>	5, 6, 20, 380, 381
counter	19	<code>\pagegoal</code>	287
debug		<code>\pagelistname</code>	39, 280, 282, 284, 285, 291, 293, 302–306, 309–312
showtargets	6	<code>\par</code>	39, 212, 274–276, 294, 296–299, 312, 313, 315–321, 330, 335–337
description	245, 246	<code>\parindent</code>	294–299, 313, 315–317, 319–321, 335–337
dua	244–246	<code>\parskip</code>	294–297, 313, 315, 316
entrycounter	11, 206, 208	<code>\PassOptionsToPackage</code>	262, 342
true	12	<code>\penalty</code>	287
esclocations	405	<code>\phantomsection</code>	45
false	9	polyglossia package	26, 38
footnote	126–131, 242, 244, 245, 248	<code>\printglossaries</code>	175
hyperfirst		<code>\printglossary</code>	17, 20, 33, 175, 192, 207
false	126–131	<code>\printglossary options</code>	
indexonlyfirst	389	entrycounter	206
kernelglossredefs		nogroupskip	206
nowarn	32	nonumberlist	207
makeindex	165, 261	nopostdot	206
nogroupskip	278, 280, 281, 284, 285, 289, 290, 292, 300, 302, 304, 307, 309, 311	numberedsection	206
nolist	255	style	205
nolong	255, 277	subentrycounter	206
nomain	16	title	205
nonumberlist	9	toctitle	205
nosuper	255	type	16, 191, 205
notree	255	<code>\printindex</code>	33
nowarn	5	<code>\printnoidxglossaries</code>	177
numberline	7	<code>\printnoidxglossary</code>	176, 177, 180, 192, 199, 200, 207
sanitize	24, 66, 152, 153	<code>\printnoidxglossary options</code>	
sanitizesort	21	sort	207
savewrites	30, 386	<code>\printnumbers</code>	33
false	172	<code>\printsymbols</code>	33
true	174, 180	<code>\ProcessOptions</code>	262, 342
section	7, 44	<code>\ProcessOptionsX</code>	34
sort		<code>\protect</code>	46, 109, 227, 228, 234, 240, 244, 246, 359, 363, 364, 369, 370
def	13, 14	<code>\protected@edef</code>	8, 48, 50, 53, 55, 85, 89, 91, 101–103, 109, 116, 159, 182, 185, 187, 206, 210, 211, 215, 224, 248, 254, 263, 269, 323, 324, 342, 343, 348
none	13		
standard	13		
use	13, 14, 405		
style	8, 255		
subentrycounter	12, 206, 208		
toc	7		
true	7		
translate	26		
false	26		
translator	25		
xindy	29, 30, 165, 261		

\protected@write	62, 63, 164, 165, 174, 175, 177, 180, 182, 191, 270, 324
\protected@xdef	13–15, 18, 23, 72, 91, 187, 346
\providecommand	17, 36, 37, 44, 62, 97, 125, 165, 174, 177, 180, 195, 210, 211, 274, 294, 312
\ProvidesFile	37
\ProvidesPackage	4, 262, 269, 271, 274, 277, 283, 288, 293, 299, 306, 312, 322, 328, 342, 380
R	
\r	23
\raggedright	286–293, 307–312
\raisebox	124
\ref	209
\refstepcounter	208
\relax	5, 8, 10, 14–17, 26, 27, 32, 34, 49, 62, 67, 68, 72, 74, 86, 88, 92, 96, 97, 112, 113, 115, 117–123, 152, 166, 167, 169, 171, 172, 175, 177, 179, 183, 187, 188, 190, 193, 194, 202, 203, 205, 206, 214, 215, 255, 270, 274, 287, 294, 298, 299, 312, 314–321, 323, 326– 328, 334–337, 345, 360, 361, 373–375, 377
\renewacronymstyle .	363–367, 369, 371, 372
\renewcommand	4– 10, 12, 13, 16, 17, 19–21, 24–26, 28, 30, 32, 34, 38–41, 54, 65, 68, 69, 83, 95– 97, 159, 160, 162, 163, 169, 170, 175– 179, 194, 195, 206, 224, 225, 227–231, 233–236, 239–243, 245, 246, 248, 251, 252, 254, 272–282, 284, 285, 287–304, 307–311, 313–323, 328–336, 338–341, 345, 350, 354, 357, 359, 362–366, 368–376
\renewenvironment	210, 271, 274, 278–283, 286–313, 315, 316, 318
\RequireGlossariesLang	38, 380, 381
\RequirePackage	4, 9, 10, 26, 27, 34, 37, 255, 261, 262, 277, 283, 284, 288, 294, 299, 306, 343, 380
\restorecounters@	116
\romannumeral ...	184, 186, 187, 318–320, 336
S	
\s@glshyp@opt	112
\s@GlsSetXdyFirstLetterAfterDigits	162
\s@GlsSetXdyNumberGroupOrder ..	162, 163
\s@newglossary	63
\savecounters@	115
\seename	190
\SetAcronymStyle	28
\setbool	25
\setbox	287, 288
\setcounter	208
\SetCustomDisplayStyle	255
\SetDefaultAcronymDisplayStyle	239
\SetDefaultAcronymStyle	253
\SetDescriptionAcronymDisplayStyle	246
\SetDescriptionAcronymStyle	253
\SetDescriptionDUAAcronymDisplayStyle	243, 244
\SetDescriptionDUAAcronymStyle	253
\SetDescriptionFootnoteAcronymDisplayStyle	242
\SetDescriptionFootnoteAcronymStyle	253
\SetDUADisplayStyle	253
\SetDUASStyle	253
\setentrycounter	48, 165, 204, 322
\SetFootnoteAcronymDisplayStyle ...	248
\SetFootnoteAcronymStyle	253
\SetGenericNewAcronym	226
\setglossarystyle	193, 215, 255, 275–287, 289–311, 314–317, 320, 321
\setglossentrycompatibility ...	205, 215
\setkeys	25, 29, 35, 44, 85, 114, 160, 161, 193, 224, 239, 241, 243, 246, 248, 251, 252, 255
\setlength	277, 288, 294– 297, 299, 306, 313, 315, 316, 320, 336, 337
\SetSmallAcronymDisplayStyle	251
\SetSmallAcronymStyle	253
\settoheight	124
\settowidth	228, 318–320, 336
\sfcode	11
\show	256–261, 378, 379
\small	6
\SmallNewAcronymDef	251
\space	6, 7, 30–32, 36, 47, 51, 52, 54, 55, 68, 70, 71, 95, 98, 99, 109, 110, 112, 158, 164–167, 172–175, 177, 179, 180, 191, 193, 195, 209, 215, 221, 227, 228, 230, 231, 233–236, 244, 249, 271, 273–276, 278, 289, 300, 307, 313–315, 317, 319, 320, 322, 323, 325–327, 329– 332, 334–338, 340, 359, 363–366, 368–373
\spacefactor	11
\SS	23
\ss	24

<code>\string</code>	6, 7, 16, 20, 30–32, 36, 46–48, 50–55, 62, 63, 68, 70, 71, 75, 78, 79, 90, 91, 95, 97–99, 110, 112, 116, 117, 119–121, 123, 158, 161, 162, 164–169, 171–180, 188, 189, 193, 195, 199, 200, 207, 212, 215, 270, 322–328
<code>\strut</code>	212, 275– 279, 281, 289, 290, 292, 300, 302, 303, 307, 309, 310, 317, 329–333, 335, 338–341
<code>\subglossentry</code> ...	90, 194, 203, 212, 272, 274, 276–279, 281, 289, 290, 292, 300, 302, 303, 307, 309, 310, 314, 315, 317, 319
<code>\subitem</code>	294, 313, 314, 334
<code>\subsubitem</code>	294, 313, 314, 334
<code>supertabular</code> package	10, 255, 299, 306
<code>\symbolname</code>	39, 281, 282, 285, 293, 304–306, 311, 312
T	
<code>\t</code>	23
<code>\tablehead</code>	300–312
<code>\tabletail</code>	300–312
<code>\tabularnewline</code>	278–282, 284, 285, 289–293, 300–312, 332, 333, 338, 339
<code>\texorpdfstring</code>	155
<code>\textbar</code>	271
<code>\textbf</code>	212, 218, 312, 334–337
<code>textcase</code> package	4
<code>\textit</code>	218
<code>\textmd</code>	218
<code>\textrm</code>	218
<code>\textsc</code> ..	219, 229, 235, 242, 246, 248, 251, 371
<code>\textsf</code>	218
<code>\textsl</code>	218
<code>\textsmaller</code> ..	229, 235, 242, 246, 248, 251, 371
<code>\texttt</code>	6, 218
<code>\textulc</code>	220
<code>\textup</code>	219, 220
<code>\TH</code>	24
<code>\th</code>	24
<code>\the</code>	35, 38, 48, 53, 55, 63, 118–123, 163, 167, 169–171, 181, 182, 186, 187, 191, 202, 210, 212, 218, 224, 225, 227, 228, 233, 234, 238, 239, 241–243, 245, 247, 250, 252, 254, 322, 324, 327, 343, 363, 364, 368, 369, 372–377
<code>\the@numberlist</code>	159
<code>\theglossary</code>	5
<code>\theglossaryentry</code>	11, 209
<code>\theglossarysubentry</code>	12, 209
<code>\theglentrycounter</code>	115, 116, 185, 187, 323, 324
<code>\theH</code>	189
<code>\theHglossaryentry</code>	11
<code>\theHglossarysubentry</code>	12
<code>\theHglentrycounter</code>	116, 185, 187
<code>\thesection</code>	35
<code>\this@dialect</code>	38, 380, 381
<code>\toks@</code>	35, 38, 48, 53, 55, 63, 118–123, 169–171, 191, 210–212, 218, 322, 342, 343
<code>\toprule</code>	284, 285
<code>tracklang</code> package	37, 380
<code>\trans@languages</code>	38
<code>\translate</code>	38, 39
<code>\translatelet</code>	16, 17, 33
<code>translator</code> package	16, 17, 26, 33, 37–39, 191
<code>\ttfamily</code>	6
<code>\TX@trial</code>	92
<code>\typeout</code>	20
U	
<code>\u</code>	23
<code>\uccode</code>	202
<code>\undef</code>	69, 75, 191
<code>\unskip</code>	83, 276, 277, 330
<code>\unvbox</code>	287, 288
<code>\usedictionary</code>	38
<code>\usepackage</code>	199, 200
V	
<code>\v</code>	23
<code>\vbox</code>	287, 288
<code>\vsize</code>	287, 288
<code>\vskip</code>	274, 287, 294, 312
<code>\vss</code>	287, 288
W	
<code>\warn@nomakeglossaries</code>	175–177
<code>\warn@noprintglossary</code>	175–177, 194
<code>\write</code>	75, 97, 163–169, 173, 174, 177, 181, 195, 324–328
<code>\writeist</code>	172, 173, 328
X	
<code>\x</code>	218
<code>\xatlevel@</code>	115
<code>\xcapitalisewords</code>	155
<code>\xdef</code>	80, 85–88, 194, 270
<code>\xglaccsupp</code>	348

\xifinlistcs	196, 197, 200	xspace package	4, 219
xindy	382		
xindy	9, 13, 29, 30, 40, 41,	Y	
	46, 49, 51, 53–55, 91, 122, 123, 162, 163,	\year	163, 167, 324, 327
	165, 181, 185, 186, 188, 195, 213, 261, 323		
\xmakefirstuc ...	101–103, 109, 151, 153, 263	Z	
\xspace	219	\z@	287