

Documented Code For glossaries v4.38

Nicola L.C. Talbot

Dickimaw Books

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

2018-05-10

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 v4.38: 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	34
1.4	Xindy	44
1.5	Loops and conditionals	53
1.6	Defining new glossaries	59
1.7	Defining new entries	64
1.8	Resetting and unsetting entry flags	90
1.9	Keeping Track of How Many Times an Entry Has Been Unset	93
1.10	Loading files containing glossary entries	98
1.11	Using glossary entries in the text	98
1.12	Adding an entry to the glossary without generating text	157
1.13	Creating associated files	159
1.14	Writing information to associated files	178
1.15	Glossary Entry Cross-References	187
1.16	Displaying the glossary	189
1.17	Acronyms	217
1.18	Predefined acronym styles	222
1.19	Predefined Glossary Styles	253
1.20	Debugging Commands	254
1.21	Compatibility with version 2.07 and below	259
2	Prefix Support (glossaries-prefix Code)	261
3	Glossary Styles	268
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	268
3.2	In-line Style (glossary-inline.sty)	270
3.3	List Style (glossary-list.sty)	273
3.4	Glossary Styles using longtable (the glossary-long package)	276
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	282
3.6	Glossary Styles using longtable (the glossary-longragged package)	287
3.7	Glossary Styles using multicol (glossary-mcols.sty)	292
3.8	Glossary Styles using supertabular environment (glossary-super package)	298
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	305
3.10	Tree Styles (glossary-tree.sty)	311

4 Backwards Compatibility	321
4.1 glossaries-compatible-207	321
4.2 glossaries-compatible-307	327
5 Accessibility Support (glossaries-accsupp Code)	341
5.1 Defining Replacement Text	342
5.2 Accessing Replacement Text	345
5.3 Displaying the Glossary	361
5.4 Acronyms	362
5.5 Debugging Commands	377
6 Multi-Lingual Support	379
6.1 Polyglossia Captions	379
Glossary	381
Change History	382
Index	406

1 Main Package Code

1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2018/05/10 v4.38 (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}[\val\nr]{true,false,showtargets}[true]{%
34   \ifcase\nr\relax
35     \@gls@debugtrue
36     \renewcommand*\GlossariesWarning}[1]{%
37       \PackageWarning{glossaries}{##1}%
38     }%
39     \renewcommand*\GlossariesWarningNoLine}[1]{%
40       \PackageWarningNoLine{glossaries}{##1}%
41     }%
42     \let\@glsshowtarget\@gobble
43     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
44   \or
45     \@gls@debugfalse
46     \let\@glsshowtarget\@gobble
47     \PackageInfo{glossaries}{debug mode OFF}%
48   \or
49     \@gls@debugtrue
50     \renewcommand*\GlossariesWarning}[1]{%
51       \PackageWarning{glossaries}{##1}%
```

```

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

```

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

```

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

```

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

```

71 \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`

```

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

```

`seenoinindex`

```

88 \define@choicekey{glossaries.sty}{seenoinindex}[\val\nr]{error,warn,ignore}{%
89   \ifcase\nr

```

```

90 \renewcommand*{\@gls@see@noindex}{%
91 \PackageError{glossaries}%
92 {\@gls@xr@key' key may only be used after \string\makeglossaries\space
93 or \string\makenoidxglossaries}%
94 {You must use \string\makeglossaries\space
95 or \string\makenoidxglossaries\space before defining
96 any entries that have a \@gls@xr@key' key}%
97 }%
98 \or
99 \renewcommand*{\@gls@see@noindex}{%
100 \GlossariesWarning{\@gls@xr@key' key ignored}%
101 }%
102 \or
103 \renewcommand*{\@gls@see@noindex}{}%
104 \fi
105 }

```

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.

```
106 \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.

```
107 \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.

```

108 \ifcsundef{chapter}%
109 {\newcommand*{\@glossarysec}{section}}%
110 {\newcommand*{\@glossarysec}{chapter}}

```

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

```

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

```

Determine whether or not to use numbered sections.

glossarysecstar

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

lossaryseclabel

```
115 \newcommand*{\@glossaryseclabel}{}%
```

\glsautoprefix Prefix to add before label if automatically generated:

```
116 \newcommand*{\glsautoprefix}{}%
```

numberedsection

```

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

```

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

```

136 \ifpackageloaded{classicthesis}
137 {\newcommand*{\@glossary@default@style}{index}}
138 {\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`.

```

139 \define@key{glossaries.sty}{style}{%
140   \def\@glossary@default@style{#1}%
141 }

```

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

```

142 \newcommand*{\@gls@declareoption}[2]{%
143   \DeclareOptionX{#1}{#2}%
144   \DeclareOption{#1}{#2}%
145 }

```


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

```
146 \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).

```
147 \@gls@declareoption{nonumberlist}{%
148   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
149 }
```

savenumberlist

Provide means to store the number list for entries.

```
150 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{}
151 \gls@savenumberlistfalse
```

eautionumberlist

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

eautionumberlist

Automatically activates number list for entries containing the see key.

```
153 \@gls@declareoption{seeautonumberlist}{%
154   \renewcommand*{\@glo@seeautonumberlist}{%
155     \def\@glo@prefix{\glsnextpages}%
156   }%
157 }
```

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.

```
158 \define@boolkey{glossaries.sty}[gls]{esclocations}[true]{}
159 \gls@esclocationstrue
```

\@gls@loadlong

```
160 \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.

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

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

```

162 \IfFileExists{supertabular.sty}{%
163   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
164   \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.

```

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

```

`\@gls@loadlist`

```

166 \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`.

```

167 \@gls@declareoption{nolist}{%
168   \renewcommand*{\@gls@loadlist}{%
169     \ifdefstring{\@glossary@default@style}{list}%
170     {\let\@glossary@default@style\relax}%
171     }%
172   }%
173 }

```

`\@gls@loadtree`

```

174 \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.

```

175 \@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).

```

176 \@gls@declareoption{nostyles}{%
177   \renewcommand*{\@gls@loadlong}{}%
178   \renewcommand*{\@gls@loadsuper}{}%
179   \renewcommand*{\@gls@loadlist}{}%
180   \renewcommand*{\@gls@loadtree}{}%
181   \let\@glossary@default@style\relax
182 }

```

`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.)

```

183 \newcommand*{\glspostdescription}{%
184   \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
185 }

```

nopostdot Boolean option to suppress post description dot

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

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

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

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

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

191 \@ifclassloaded{memoir}
192 {%
193   \glsucmarktrue
194 }%
195 {%
196   \glsucmarkfalse
197 }
```

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

```
198 \newcommand*{\@gls@define@glossaryentrycounter}{%
199   \ifgl Sentrycounter
200     Define the glossaryentry counter if it doesn't already exist.
201     \ifundef\c@glossaryentry
202       {%
203         \ifx\@gls@counterwithin\@empty
204           \newcounter{glossaryentry}%
205         \else
206           \newcounter{glossaryentry}[\@gls@counterwithin]%
207         \fi
208       }%
209     }%
210   \fi
211 }
```

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`.

```
212 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
213 \gl Sentrycounterfalse
```

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

```

214 \define@key{glossaries.sty}{counterwithin}{%
215   \renewcommand*{\@gls@counterwithin}{#1}%
216   \glssentrycountertrue
217   \@gls@define@glossaryentrycounter
218 }

s@counterwithin The default value is no parent counter:
219 \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
220 \newcommand{\@gls@define@glossarysubentrycounter}{%
    Check if counter already defined.
221   \ifundef\c@glossarysubentry
222   {%
223     \ifglssubentrycounter
224     \ifglssentrycounter
225     \newcounter{glossarysubentry}[glossaryentry]%
226     \else
227     \newcounter{glossarysubentry}%
228     \fi
    As with \theHglossaryentry, this starts with \currentglossary. to help avoid duplicate
    hyper targets.
229     \def\theHglossarysubentry{\currentglossary.\currentglssubentry.\theglossarysubentry}%
230     \fi
231   }%
232   {}%
233 }

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.
234 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
235 \glssubentrycounterfalse

efault@sorttype Initialise default sort for \printnoidxglossary
236 \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.
237 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
238   \renewcommand*{\@glo@default@sorttype}{#1}%
239   \csname @gls@setupsort@#1\endcsname
240 }

```

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.

```
241 \newcommand*{\glsprestandardsort}[3]{%
242   \glsdosanitizesort
243 }
```

glscheck@sortallowed

```
244 \newcommand*{\@gls@check@sortallowed}[1]{}
```

glssetupsort@standard

Set up the macros for default sorting.

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

Store entry information when it's defined.

```
246   \def\do@gls@storeentry{\@gls@storeentry}%
```

No count register required for standard sort.

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

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

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

```
249     \ifx\@gls@sort\@gls@defaultsort
```

```
250       \let\@gls@sort\@gls@name
```

```
251     \fi
```

```
252     \let\glsdosanitizesort\@gls@sanitizesort
```

```
253     \glsprestandardsort{\@gls@sort}{##1}{##2}%
```

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

```
255   }%
```

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

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

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

```
257   \let\@gls@check@sortallowed\@gobble
```

```
258 }
```

Set standard sort as the default:

```
259 \@gls@setupsort@standard
```

glsnumberfmt

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

```
260 \newcommand*{\glsnumberfmt}[1]{%
```

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

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

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

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

```

265 \ifnum#1<10 0\fi
266 \number#1%
267 }

s@setupsort@def Set up the macros for order of definition sorting.
268 \newcommand*{\@gls@setupsort@def}{%
  Store entry information when it's defined.
269 \def\do@glo@storeentry{\@glo@storeentry}%
  Defined count register associated with the glossary.
270 \def\@gls@defsortcount##1{%
271 \expandafter\global
272 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
273 }%
  Increment count register associated with the glossary and use as the sort key.
274 \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.
275 \ifcsundef{glossary@##1@sortcount}%
276 {\@gls@defsortcount{##1}}%
277 {}%
278 \expandafter\global\expandafter
279 \advance\csname glossary@##1@sortcount\endcsname by 1\relax
280 \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
281 \expandafter\glssortnumberfmt
282 {\csname glossary@##1@sortcount\endcsname}}%
283 }%
  Don't need to do anything when the entry is used.
284 \def\@gls@setsort##1{%
  This sort option is allowed with \makeglossaries and \makenoidxglossaries.
285 \let\@glo@check@sortallowed\@gobble
286 }

s@setupsort@use Set up the macros for order of use sorting.
287 \newcommand*{\@gls@setupsort@use}{%
  Don't store entry information when it's defined.
288 \let\do@glo@storeentry\@gobble
  Defined count register associated with the glossary.
289 \def\@gls@defsortcount##1{%
290 \expandafter\global
291 \expandafter\newcount\csname glossary@##1@sortcount\endcsname
292 }%
  Initialise the sort key to empty.
293 \def\@gls@defsort##1##2{%
294 \expandafter\gdef\csname glo@##2@sort\endcsname{%
295 }%

```

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

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

Get the parent, if one exists

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

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

```
298 \ifx\@glo@parent\@empty
```

```
299 \else
```

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

```
301 \fi
```

Set index information for this entry

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

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

```
304 \ifx\@gls@tmp\@empty
```

```
305 \expandafter\global\expandafter
```

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

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

```
308 \expandafter\glssortnumberfmt
```

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

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

```
311 \fi
```

```
312 }%
```

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

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

```
314 }
```

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

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

Don't store entry index information.

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

No count register required for standard sort.

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

Don't modify sort value.

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

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

```
320 }%
```

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

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

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

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

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

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

```
325 }
```

`\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.)

```

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

```

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

```

332 \newcommand*{\gls@tr@set@main@toctitle}{%
333   \translatelet{\glossarytoctitle}{Glossary}%
334 }%
335 }

```

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`

```

336 \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`

```

337 \newcommand*{\acronymtype}{\glsdefaulttype}

```

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

```

338 \@gls@declareoption{nomain}{%
339   \let\glsdefaulttype\relax
340   \renewcommand*{\glsdefmain}{}%
341 }

```

`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.

```

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

```


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

```
348     \newcommand*{\gls@tr@set@acronym@toctitle}{%
349         \translatelet{\glossarytoctitle}{Acronyms}%
350     }%
351 }%
352 \else
353     \let\@gls@do@acronymsdef\relax
354 \fi
355 }
```

`\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.

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

`@do@acronymsdef` Set default value

```
366 \newcommand*{\@gls@do@acronymsdef}{}%
```

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

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

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

```
373     \newcommand*{\gls@tr@set@acronym@toctitle}{%
374         \translatelet{\glossarytoctitle}{Acronyms}%
375     }%
376 }%
377 }
```

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

```
378 \newcommand*{\@glsacronymlists}{}%
```

`dtoacronymlists`

```
379 \newcommand*{\@addtoacronymlists}[1]{%
380     \ifx\@glsacronymlists\@empty
```

```

381 \protected@xdef\@glsacronymlists{#1}%
382 \else
383 \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
384 \fi
385 }

```

`\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.)

```

386 \newcommand*\@DeclareAcronymList[1]{%
387 \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
388 }

```

`\IfListOfAcronyms`

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

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

```

389 \newcommand{\glsIfListOfAcronyms}[1]{%
390 \edef\@do@gls@islistofacronyms{%
391 \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
392 \@do@gls@islistofacronyms
393 }

```

Internal command requires label and list to be expanded:

```

394 \newcommand{\@gls@islistofacronyms}[4]{%
395 \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
396 \def\@before{##1}\def\@after{##2}}%
397 \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
398 \ifx\@after\@nnil

```

Not found

```

399 #4%
400 \else

```

Found

```

401 #3%
402 \fi
403 }

```

`\glsisacronymlist` Convenient boolean.

```
404 \newif\if@glsisacronymlist
```

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

```

405 \newcommand*\@chkisacronymlist[1]{%
406 \glsIfListOfAcronyms{#1}%
407 {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
408 }

```

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.)

```
409 \newcommand*{\SetAcronymLists}[1]{%
410   \renewcommand*{\@glsacronymlists}{#1}%
411 }
```

acronymlists

```
412 \define@key{glossaries.sty}{acronymlists}{%
413   \DeclareAcronymList{#1}%
414 }
```

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

```
415 \newcommand{\glscounter}{page}
```

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

```
416 \define@key{glossaries.sty}{counter}{%
417   \renewcommand*{\glscounter}{#1}%
418 }
```

gls@nohyperlist

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

lareNoHyperList

```
420 \newcommand*{\GlsDeclareNoHyperList}[1]{%
421   \ifdefempty\@gls@nohyperlist
422   {%
423     \renewcommand*{\@gls@nohyperlist}{#1}%
424   }%
425   {%
426     \appto\@gls@nohyperlist{,#1}%
427   }%
428 }
```

nohypertypes

```
429 \define@key{glossaries.sty}{nohypertypes}{%
430   \GlsDeclareNoHyperList{#1}%
431 }
```

ossariesWarning Prints a warning message.

```
432 \newcommand*{\GlossariesWarning}[1]{%
433   \PackageWarning{glossaries}{#1}%
434 }
```

esWarningNoLine Prints a warning message without the line number.

```

435 \newcommand*\GlossariesWarningNoLine}[1]{%
436   \PackageWarningNoLine{glossaries}{#1}%
437 }

```

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

```

438 \newcommand*\glosortentrieswarning{%
439   \typeout{Using TeX to sort glossary entries---this may
440     take a while}%
441 }

```

nowarn Define package option to suppress warnings

```

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

```

issinglang@warn Missing language warning.

```

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

```

nolangwarn Suppress warning if language support not found.

```

459 \@gls@declareoption{nolangwarn}{%
460   \renewcommand*\@gls@missinglang@warn}[2]{}%
461 }

```

nonglossdefined Issue a warning if overriding \printglossary

```

462 \newcommand*\@gls@warnonglossdefined{%
463   \GlossariesWarning{Overriding \string\printglossary}%
464 }

```

theglossdefined Issue a warning if overriding theglossary

```

465 \newcommand*\@gls@warnontheglossdefined{%
466   \GlossariesWarning{Overriding 'theglossary' environment}%
467 }

```

noredefwarn Suppress warning on redefinition of \printglossary

```

468 \@gls@declareoption{noredefwarn}{%
469   \renewcommand*{\@gls@warnonglossdefined}{}%
470   \renewcommand*{\@gls@warnontheGLOSSdefined}{}%
471 }
```

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

```

472 \newcommand*{\@gls@sanitizedesc}{%
473 }
```

lssetexpandfield `\glssetexpandfield{<field>}`

Sets field to always expand.

```

474 \newcommand*{\glssetexpandfield}[1]{%
475   \csdef{gls@assign@#1@field}##1##2{%
476     \@gls@expand@field{##1}{#1}{##2}%
477   }%
478 }
```

setnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```

479 \newcommand*{\glssetnoexpandfield}[1]{%
480   \csdef{gls@assign@#1@field}##1##2{%
481     \@gls@noexpand@field{##1}{#1}{##2}%
482   }%
483 }
```

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

```

484 \glssetexpandfield{type}
```

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

```

485 \glssetnoexpandfield{desc}
```

escplural@field

```

486 \glssetnoexpandfield{descplural}
```

ls@sanitizename

```

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

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

```

488 \glssetnoexpandfield{name}
```

@sanitizesymbol

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

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

```
490 \glssetnoexpandfield{symbol}
```

bolplural@field

```
491 \glssetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```
492 \newcommand*{\@gls@sanitizesort}{%  
493   \ifglssanitizesort  
494     \@gls@sanitizesort  
495   \else  
496     \@gls@nosanitizesort  
497   \fi  
498 }
```

ls@sanitizesort

```
499 \newcommand*{\@gls@sanitizesort{%  
500   \@onelevel@sanitize\@glo@sort  
501 }
```

@nosanitizesort

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

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

```
503 \newcommand*{\@gls@noidx@sanitizesort{%  
504   \ifdefvoid\@glo@sort  
505   }{%  
506   {%  
507     \expandafter\@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort  
508   }%  
509 }  
510 \def\@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%  
511   \def\@glo@sort{#1#2}%  
512   \@onelevel@sanitize\@glo@sort  
513 }
```

@nosanitizesort

```
514 \newcommand*{\@gls@noidx@nosanitizesort}{%  
515   \ifdefvoid\@glo@sort  
516   }{%  
517   {%  
518     \expandafter\@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort  
519   }%
```

```

520 }
521 \def\@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
522   \bgroup
523     \glsnoidxstripaccents
524     \protected@xdef\@glo@sort{#1#2}%
525   \egroup
526   \let\@glo@sort\@glo@sort
527 }

```

`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`.

```

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

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

559   \def\TH{TH}%
560   \def\dh{dh}%
561   \def\DH{DH}%
562 }

```

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.

```

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

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

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

```

sanitizesort

```

591 \define@boolkey{glossaries.sty}[glS]{sanitizesort}[true]{%
592   \ifglSsanitizesort
593     \glSsetnoexpandfield{sortvalue}%
594     \renewcommand*{\@glS@noidx@setsanitizesort}{%
595       \glSsanitizesorttrue
596       \glSsetnoexpandfield{sortvalue}%
597     }%
598   \else
599     \glSsetexpandfield{sortvalue}%
600     \renewcommand*{\@glS@noidx@setsanitizesort}{%
601       \glSsanitizesortfalse
602       \glSsetexpandfield{sortvalue}%
603     }%
604   \fi
605 }

```


Default setting:

```
606 \glssanitizesorttrue
607 \glsssetnoexpandfield{sortvalue}%
```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```
608 \newcommand*{\@gls@noidx@setsanitizesort}{%
609   \glssanitizesortfalse
610   \glsssetexpandfield{sortvalue}%
611 }

612 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
613   \setbool{glssanitizesort}{#1}%
614   \ifglssanitizesort
615     \glsssetnoexpandfield{sortvalue}%
616   \else
617     \glsssetexpandfield{sortvalue}%
618   \fi
619   \GlossariesWarning{sanitize={sort} package option
620     deprecated. Use sanitizesort instead}%
621 }
```

sanitize

```
622 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
623   \ifthenelse{\equal{#1}{none}}{%
624     {%
625       \GlossariesWarning{sanitize package option deprecated}%
626       \glsssetexpandfield{name}%
627       \glsssetexpandfield{symbol}%
628       \glsssetexpandfield{symbolplural}%
629       \glsssetexpandfield{desc}%
630       \glsssetexpandfield{descplural}%
631     }%
632     {%
633       \setkeys[gls]{sanitize}{#1}%
634     }%
635 }
```

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

```
636 \newif\ifglstranslate
```

otranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator

```
637 \newcommand*\@gls@usetranslator{%
  polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
  polyglossia as well.
638   \@ifpackageloaded{polyglossia}%
```

```

639  {%
640    \let\glsifusetranslator\@secondoftwo
641  }%
642  {%
643    \@ifpackageloaded{babel}%
644    {%
645      \IfFileExists{translator.sty}%
646      {%
647        \RequirePackage{translator}%
648        \let\glsifusetranslator\@firstoftwo
649      }%
650    }%
651  }%
652  {}%
653 }%
654 }

```

dtranslatordict Checks if given translator dictionary has been loaded.

```

655 \newcommand{\glsifusedtranslator}dict}[3]{%
656   \glsifusetranslator
657   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
658   {#3}%
659 }

```

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

```

660 \@gls@declareoption{notranslate}{%
661   \glstranslatefalse
662   \let\@gls@usetranslator\relax
663   \let\glsifusetranslator\@secondoftwo
664 }

```

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

```

665 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
666 {true,false,babel}[true]%
667 {%
668   \ifcase\nr\relax
669     \glstranslatetrue
670     \renewcommand*\@gls@usetranslator{%
671       \@ifpackageloaded{polyglossia}%
672       {%
673         \let\glsifusetranslator\@secondoftwo
674       }%
675     }%
676     \@ifpackageloaded{babel}%
677     {%
678       \IfFileExists{translator.sty}%
679       {%
680         \RequirePackage{translator}%
681         \let\glsifusetranslator\@firstoftwo

```

```

682         }%
683     {}%
684     }%
685     {}%
686     }%
687 }%
688 \or
689     \glstranslatefalse
690     \let\@gls@usetranslator\relax
691     \let\glsifusetranslator\@secondoftwo
692 \or
693     \glstranslatetrue
694     \let\@gls@usetranslator\relax
695     \let\glsifusetranslator\@secondoftwo
696 \fi
697 }

```

Set the default value:

```

698 \glstranslatefalse
699 \let\glsifusetranslator\@secondoftwo
700 \@ifpackageloaded{translator}%
701 {%
702     \glstranslatetrue
703     \let\glsifusetranslator\@firstoftwo
704 }%
705 {%
706     \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
707     {
708         \@ifpackageloaded{\gls@thissty}%
709         {%
710             \glstranslatetrue
711             \@endfortrue
712         }%
713     }%
714 }
715 }

```

indexonlyfirst Set whether to only index on first use.

```

716 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
717 \glsindexonlyfirstfalse

```

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

```

718 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
719 \glshyperfirsttrue

```

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

```

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

```

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

```

721 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{%
722   \ifbool{glsacrdescription}%
723   {}%
724   {%
725     \renewcommand*{\@gls@sanitizedesc}{}%
726   }%
727   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
728 }

```

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

```

729 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
730   \renewcommand*{\@gls@sanitizesymbol}{}%
731   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
732 }

```

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

```

733 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
734   \renewcommand*{\@gls@sanitizesymbol}{}%
735   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
736 }

```

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

```

737 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
738   \renewcommand*{\@gls@sanitizesymbol}{}%
739   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
740 }

```

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

```

741 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
742   \renewcommand*{\@gls@sanitizesymbol}{}%
743   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
744 }

```

shortcuts Define acronym shortcuts.

```

745 \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.

```

746 \newcommand*{\glsorder}{word}

```

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

```

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

```

order

```

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

```

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

```
750 \newif\ifglxindy
```

The default is `makeindex`:

```
751 \glxindyfalse
```

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

```
752 \@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.

```
753 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
754 \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.)

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

Define key to set the language

```
756 \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.

```
757 \ifcsundef{inputencodingname}{%
758   \def\gls@codepage{}}{%
759   \def\gls@codepage{\inputencodingname}
760 }
```

Define a key to set the code page.

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

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

```
762 \define@key{glossaries.sty}{xindy}[]{%
763   \glxindytrue
764   \setkeys[gls]{xindy}{#1}%
765 }
```

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

```
766 \@gls@declareoption{xindygloss}{%
767   \glxindytrue
768 }
```

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

```
769 \@gls@declareoption{xindynoglsnumbers}{%
770   \glxindytrue
771   \gls@xindy@glsnumbersfalse
772 }
```

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

```

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

```

@gls@doautomake

```

788 \newcommand*{\@gls@doautomake}{}
789 \AtEndDocument{\@gls@doautomake}

```

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

```

790 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
791   \ifglssavewrites
792     \renewcommand*{\glswritefiles}{\@glswritefiles}%
793   \else
794     \let\glswritefiles\@empty
795   \fi
796 }

```

Set default:

```

797 \glssavewritesfalse
798 \let\glswritefiles\@empty

```

compatible-3.07

```

799 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{}
800 \boolfalse{glscpatible-3.07}

```

compatible-2.07

```

801 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
  Also set 3.07 compatibility if this option is set.
802   \ifbool{glscpatible-2.07}{%
803     {%
804       \booltrue{glscpatible-3.07}%
805     }%
806   }%
807 }
808 \boolfalse{glscpatible-2.07}

```

symbols Create a “symbols” glossary type

```
809 \@gls@declareoption{symbols}{%  
810 \let\@gls@do@symbolsdef\@gls@symbolsdef  
811 }
```

Default is not to define the symbols glossary:

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

@gls@symbolsdef

```
813 \newcommand*{\@gls@symbolsdef}{%  
814 \newglossary[slg]{symbols}{sls}{slo}{\glsymbolsgroupname}%  
815 \newcommand*{\printsymbols}[1][]{\printglossary[type=symbols,##1]}%
```

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

```
816 \newcommand*{\gls@tr@set@symbols@toctitle}{%  
817 \translatelet{\glossarytoctitle}{Symbols (glossaries)}%  
818 }%  
819 }%
```

numbers Create a “numbers” glossary type

```
820 \@gls@declareoption{numbers}{%  
821 \let\@gls@do@numbersdef\@gls@numbersdef  
822 }
```

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

```
827 \newcommand*{\gls@tr@set@numbers@toctitle}{%  
828 \translatelet{\glossarytoctitle}{Numbers (glossaries)}%  
829 }%  
830 }%
```

index Create an “index” glossary type

```
831 \@gls@declareoption{index}{%  
832 \let\@gls@do@indexdef\@gls@indexdef  
833 }
```

Default is not to define index glossary:

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

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

```
835 \newcommand*{\@gls@indexdef}{%  
836 \newglossary[ilg]{index}{ind}{idx}{\indexname}%  
837 \newcommand*{\printindex}[1][]{\printglossary[type=index,##1]}%
```

```

838 \newcommand*{\newterm}[2] [] {%
839   \newglossaryentry{##2}%
840   {type={index},name={##2},description={\nopostdesc},##1}}
841 }%

```

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

```

842 \@for\CurrentOption := \@declaredoptions\do{%
843   \ifx\CurrentOption \@empty
844   \else
845     \@expandtwoargs
846     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
847     \ifin@
848     \@use@option
849     \expandafter \let\csname ds@\CurrentOption\endcsname \@empty
850   \fi
851 \fi
852 }

```

Now process options passed to the package:

```
853 \ProcessOptionsX
```

Load backward compatibility stuff:

```
854 \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:

```

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

```

Now define `\setupglossaries`:

```

858 \newcommand*{\setupglossaries}[1] {%
859   \renewcommand*{\@gls@setacrstyle}{}%
860   \ifglsacrshortcuts
861     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
862   \else
863     \def\@gls@setupshortcuts{%
864       \ifglsacrshortcuts
865         \DefineAcronymSynonyms
866       \fi
867     }%
868   \fi
869   \glsacrshortcutsfalse
870   \let\@gls@do@numbersdef\relax
871   \let\@gls@do@symbolssdef\relax
872   \let\@gls@do@indexdef\relax
873   \let\@gls@do@acronymsdef\relax
874   \ifglsentrycounter
875     \let\@gls@doentrycounterdef\relax

```



```

876 \else
877   \let\@gls@doentrycounterdef\@gls@define@glossaryentrycounter
878 \fi
879 \ifglssubentrycounter
880   \let\@gls@dosubentrycounterdef\relax
881 \else
882   \let\@gls@dosubentrycounterdef\@gls@define@glossarysubentrycounter
883 \fi
884 \setkeys{glossaries.sty}{#1}%
885 \@gls@setacrstyle
886 \@gls@setupshortcuts
887 \@gls@do@acronymsdef
888 \@gls@do@numbersdef
889 \@gls@do@symbolssdef
890 \@gls@do@indexdef
891 \@gls@doentrycounterdef
892 \@gls@dosubentrycounterdef
893 }

```

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

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

```

894 \ifthenelse{\equal{\glscounter}{section}}{%
895 {%
896   \ifcsundef{chapter}{}%
897   {%
898     \let\@gls@old@chapter\@chapter
899     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
900     \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}}}%
901   }%
902 }%
903 {}

```

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

```

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

```

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

```

905 \newcommand*{\@onlypremakeg}[1]{%
906   \ifx\@gls@onlypremakeg\@empty
907     \def\@gls@onlypremakeg{#1}%
908   \else
909     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%

```

```

910 \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
911 \fi
912 }

```

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

```

913 \newcommand*{\@disable@onlypremakeg}{%
914 \@for\@thiscs:=\@gls@onlypremakeg\do{%
915 \expandafter\@disable@premakecs\@thiscs%
916 }}

```

`\sable@premakecs` Disables the given command.

```

917 \newcommand*{\@disable@premakecs}[1]{%
918 \def#1{\PackageError{glossaries}{\string#1\space may only be
919 used before \string\makeglossaries}{You can't use
920 \string#1\space after \string\makeglossaries}}}%
921 }

```

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`

```

922 \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`

```

923 \providecommand*{\acronymname}{Acronyms}

```

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

```

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

```

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

`\entryname`

```

926 \providecommand*{\entryname}{Notation}

```

`\descriptionname`

```

927 \providecommand*{\descriptionname}{Description}

```

`\symbolname`

```

928 \providecommand*{\symbolname}{Symbol}

```

`\pagelistname`

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

Labels for makeindex's symbol and number groups:

`\symbolsgroupname`

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

`\numbersgroupname`

```
931 \providecommand*{\glsnumbersgroupname}{Numbers}
```

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

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

`\acrpluralsuffix` Default plural suffix for acronyms

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

`\acrpluralsuffix`

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

`\seename`

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

`\andname`

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

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

`\eGlossariesLang`

```
937 \newcommand*{\RequireGlossariesLang}[1]{%
938   \@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}%
939 }
```

`\sGlossariesLang`

```
940 \newcommand*{\ProvidesGlossariesLang}[1]{%
941   \ProvidesFile{glossaries-#1.ldf}%
942 }
```

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

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

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

```
944 \ifglstranslate
```

Load tracklang

```
945 \RequirePackage{tracklang}
```

Load translator if required.

```
946 \@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.)

```
947 \@ifpackageloaded{translator}
```

```
948 {%
```

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.

```
949 \ifboolexpr
```

```
950 {
```

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

```
952 and not
```

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

```
954 }
```

```
955 {%
```

```
956 \let\glsifusetranslator\@secondoftwo
```

```
957 }%
```

```
958 {%
```

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

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

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

```
962 {%
```

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

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

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

```
966 }%
```

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

```
968 }%
```

```
969 }%
```

```
970 }%
```

```
971 }%
```

```
972 {}%
```

Check for tracked languages

```
973 \AnyTrackedLanguages
```

```
974 {%
```

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

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

```
977 {glossaries-}% prefix
```

```
978 {.ldf}%
```

```
979 {%
```

```
980 \RequireGlossariesLang{\CurrentTrackedTag}%
```

```
981 }%
```

```
982 {%
```

```
983 \@gls@missinglang@warn\this@dialect\CurrentTrackedLanguage
```

```

984     }%
985   }%
986 }%
987 {}%

if using translator use translator interface.
988 \glsifusetranslator
989 {%
990   \renewcommand*{\glssettoctitle}[1]{%
991     \ifcsdef{gls@tr@set@#1@toctitle}%
992     {%
993       \csuse{gls@tr@set@#1@toctitle}%
994     }%
995     {%
996       \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
997     }%
998   }%
999   \renewcommand*{\glossaryname}{\translate{Glossary}}}%
1000   \renewcommand*{\acronymname}{\translate{Acronyms}}}%
1001   \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
1002   \renewcommand*{\descriptionname}{%
1003     \translate{Description (glossaries)}}}%
1004   \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
1005   \renewcommand*{\pagelistname}{%
1006     \translate{Page List (glossaries)}}}%
1007   \renewcommand*{\glssymbolsgroupname}{%
1008     \translate{Symbols (glossaries)}}}%
1009   \renewcommand*{\glsnumbersgroupname}{%
1010     \translate{Numbers (glossaries)}}}%
1011   }{}%
1012 \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.

```
1013 \DeclareRobustCommand*{\nopostdesc}{}
```

`\@nopostdesc` Suppress next description terminator.

```

1014 \newcommand*{\@nopostdesc}{%
1015   \let\org@glspostdescription\glspostdescription
1016   \def\glspostdescription{%
1017     \let\glspostdescription\org@glspostdescription}%
1018 }

```

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

```
1019 \newcommand*{\@no@post@desc}{\nopostdesc}
```

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

```
1020 \newcommand{\glspar}{\par}
```

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

```
1021 \newcommand{\setStyleFile}[1]{%
1022   \renewcommand*{\gls@istfilebase}{#1}%
      Just in case \istfilename has been modified.
1023   \ifglsxindy
1024     \def\istfilename{\gls@istfilebase.xdy}
1025   \else
1026     \def\istfilename{\gls@istfilebase.ist}
1027   \fi
1028 }
```

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

```
1029 \@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`

```
1030 \ifglsxindy
1031   \def\istfilename{\gls@istfilebase.xdy}
1032 \else
1033   \def\istfilename{\gls@istfilebase.ist}
1034 \fi
```

`\gls@istfilebase`

```
1035 \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`

```
1036 \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`

```
1037 \newcommand*{\glscompositor}{.}
```

`\lsSetCompositor` Sets the compositor.

```
1038 \newcommand*{\glsSetCompositor}[1]{%
1039   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
1040 \@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.

```
1041 \newcommand*{\@glsAlphacompositor}{\glscompositor}
```

AlphaCompositor Sets the alpha compositor.

```
1042 \ifglsxindy
1043   \newcommand*\glsSetAlphaCompositor[1]{%
1044     \renewcommand*\@glsAlphacompositor{#1}}
1045 \else
1046   \newcommand*\glsSetAlphaCompositor[1]{%
1047     \glsnoxywarning\glsSetAlphaCompositor}
1048 \fi
```

Can only be used before `\makeglossaries`

```
1049 \@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.

```
1050 \newcommand*{\gls@suffixF}{}
```

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

```
1051 \newcommand*{\glsSetSuffixF}[1]{%
1052   \renewcommand*{\gls@suffixF}{#1}}
```

Only has an effect when used before `\makeglossaries`

```
1053 \@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.

```
1054 \newcommand*{\gls@suffixFF}{}
```

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

```
1055 \newcommand*{\glsSetSuffixFF}[1]{%
1056   \renewcommand*{\gls@suffixFF}{#1}%
1057 }
```

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

```

1058 \ifcsundef{hyperlink}%
1059 {%
1060   \newcommand*{\glsnumberformat}[1]{#1}%
1061 }%
1062 {%
1063   \newcommand*{\glsnumberformat}[1]{\glsnumberformat{#1}}%
1064 }

```

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
1065 \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
1066 \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
1067 \newcommand*{\glossarypreamble}{%
1068   \csuse{@glossarypreamble@\currentglossary}%
1069 }

```

```

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

```

Code provided by Michael Pock.

```

1070 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
1071   \ifglossaryexists{#1}{%
1072     \csgdef{@glossarypreamble@#1}{#2}%
1073   }{%
1074     \GlossariesWarning{%
1075       Glossary ‘#1’ is not defined%
1076     }%
1077   }%
1078 }

```


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`

```
1079 \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`.

```
1080 \newcommand*{\glossarysection}[2][\@gls@title]{%
1081   \def\@gls@title{#2}%
1082   \ifcsundef{phantomsection}%
1083     {%
1084       \@glossarysection{#1}{#2}%
1085     }%
1086     {%
1087       \p@glossarysection{#1}{#2}%
1088     }%

1089   \glsglossarymark{\glossarytoctitle}%
1090 }
```

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

```
1091 \ifcsundef{glossarymark}%
1092 {%
1093   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1094 }%
1095 {%
1096   \@ifclassloaded{memoir}
1097   {%
1098     \newcommand{\glsglossarymark}[1]{%
1099       \ifglsucmark
1100         \markboth{\memUHead{#1}}{\memUHead{#1}}%
1101       \else
1102         \markboth{#1}{#1}%
1103       \fi
1104     }
1105   }%
1106   {%
1107     \newcommand{\glsglossarymark}[1]{%
1108       \ifglsucmark
1109         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1110       \else
1111         \markboth{#1}{#1}%
1112       \fi
1113     }
1114   }%
1115 }
```

```

1110      \else
1111      \mkboth{#1}{#1}%
1112      \fi
1113    }
1114  }
1115}

```

`\glossarymark` Provided for backward compatibility:

```

1116 \providecommand{\glossarymark}[1]{%
1117   \ifglsucmark
1118     \mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1119   \else
1120     \mkboth{#1}{#1}%
1121   \fi
1122 }

```

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`

```

1123 \newcommand*{\setglossarysection}[1]{%
1124 \setkeys{glossaries.sty}{section=#1}}

```

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

`\glossarysection`

```

1125 \newcommand*{\@glossarysection}[2]{%
1126   \ifdefempty\@glossarysecstar
1127   {%
1128     \csname\@glossarysec\endcsname[#1]{#2}%
1129   }%
1130   {%
1131     \csname\@glossarysec\endcsname*{#2}%
1132     \@gls@toc{#1}{\@glossarysec}%
1133   }%

```

Do automatic labelling if required

```

1134   \@glossaryseclabel
1135 }

```

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`

```

1136 \newcommand*{\@pglossarysection}[2]{%
1137   \glsclearpage

```

```

1138 \phantomsection
1139 \ifdefempty\@glossarysecstar
1140 {%
1141   \csname\@glossarysec\endcsname{#2}%
1142 }%
1143 {%
1144   \@gls@toc{#1}{\@glossarysec}%
1145   \csname\@glossarysec\endcsname*{#2}%
1146 }%

```

Do automatic labelling if required

```

1147 \@glossaryseclabel
1148 }

```

`\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.

```

1149 \newcommand*{\gls@doclearpage}{%
1150   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1151     {%
1152       \ifcsundef{cleardoublepage}%
1153       {%
1154         \clearpage
1155       }%
1156     {%
1157       \ifcsdef{if@openright}%
1158       {%
1159         \if@openright
1160           \cleardoublepage
1161         \else
1162           \clearpage
1163         \fi
1164       }%
1165     }%
1166     \cleardoublepage
1167   }%
1168 }%
1169 }%
1170 {}%
1171 }

```

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

```

1172 \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`

```
1173 \newcommand*{\@gls@toc}[2]{%
1174   \ifglstoc
1175     \ifglsnumberline
1176       \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1177     \else
1178       \addcontentsline{toc}{#2}{#1}%
1179     \fi
1180   \fi
1181 }
```

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

```
1182 \newcommand*{\glsnoindexwarning}[1]{%
1183   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1184 }
```

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

```
1185 \newcommand*{\glsnomakeindexwarning}[1]{%
1186   \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1187 }
```

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

```
1188 \ifglsxindy
1189   \edef\@xdyattributes{\string"default\string"}%
1190 \fi
```

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

```
1191 \ifglsxindy
1192   \edef\@xdyattributelist{}%
1193 \fi
```

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

```
1194 \ifglsxindy
1195   \def\@xdylocref{}
1196 \fi
```

`\@gls@ifinlist`

```
1197 \newcommand*{\@gls@ifinlist}[4]{%
1198   \def\@do@ifinlist##1,#1,##2\end@do@ifinlist{%
1199     \def\@gls@listsuffix{##2}%
1200     \ifx\@gls@listsuffix\@empty
```

```

1201      #4%
1202      \else
1203      #3%
1204      \fi
1205  }%
1206  \@do@ifinlist,#2,#1,\end@do@ifinlist
1207 }

```

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.

```

1208 \ifglxsindy
1209   \newcommand*{\@xdycounters}{\@glscounter}
1210   \newcommand*\GlsAddXdyCounters[1]{%
1211     \@for\@gls@ctr:=#1\do{%

```

Check if already in list before adding.

```

1212       \edef\@do@addcounter{%
1213         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1214         {%
1215           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1216             \noexpand\@gls@ctr}%
1217         }%
1218       }%
1219       \@do@addcounter
1220     }
1221   }

```

Only has an effect before `\writeist`:

```

1222   \@onlypremakeg\GlsAddXdyCounters
1223 \else
1224   \newcommand*\GlsAddXdyCounters[1]{%
1225     \glsnoxindywarning\GlsAddXdyAttribute
1226   }
1227 \fi

```

saddxdycounters Counters must all be identified before adding attributes.

```

1228 \newcommand*\@disabled@glssaddxdycounters{%
1229   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1230     can't be used after \string\GlsAddXdyAttribute}{Move all
1231     occurrences of \string\GlsAddXdyCounters\space before the first
1232     instance of \string\GlsAddXdyAttribute}%
1233 }

```

AddXdyAttribute Adds an attribute.

```

1234 \ifglxsindy

```

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

```

1235   \newcommand*\@glssaddxdyattribute[2]{%

```

Add to xindy attribute list

```
1236 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1237 \string"#2#1\string"}%
```

Add to xindy markup location.

```
1238 \expandafter\toks@\expandafter{\@xdylocref}%
1239 \edef\@xdylocref{\the\toks@ ^^J%
1240 (markup-locref
1241 :open \string"glstildechar n%
1242 \expandafter\string\csname glsX#2X#1\endcsname
1243 \string" ^^J
1244 :close \string"\string" ^^J
1245 :attr \string"#2#1\string")}%
```

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

```
1246 \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1247 \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1248 }%
1249 }
```

High-level command:

```
1250 \newcommand*\GlsAddXdyAttribute[1]{%
```

Add to comma-separated attribute list

```
1251 \ifx\@xdyattributelist\@empty
1252 \edef\@xdyattributelist{#1}%
1253 \else
1254 \edef\@xdyattributelist{\@xdyattributelist,#1}%
1255 \fi
```

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

```
1256 \@for\@this@counter:=\@xdycounters\do{%
1257 \protected@edef\gls@do@addxdyattribute{%
1258 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1259 }
1260 \gls@do@addxdyattribute
1261 }%
```

All occurrences of $\backslash\mathrm{GlsAddXdyCounters}$ must be used before this command

```
1262 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1263 }
```

Only has an effect before $\backslash\mathrm{writeist}$:

```
1264 \@onlypremakeg\GlsAddXdyAttribute
1265 \else
1266 \newcommand*\GlsAddXdyAttribute[1]{%
1267 \glsnoxindywarning\GlsAddXdyAttribute}
1268 \fi
```

$\backslash\mathrm{definedattributes}$ Add known attributes for all defined counters

```
1269 \ifglsxindy
1270 \newcommand*\@gls@addpredefinedattributes{%
```

```

1271 \GlsAddXdyAttribute{glsnumberformat}
1272 \GlsAddXdyAttribute{textrm}
1273 \GlsAddXdyAttribute{textsf}
1274 \GlsAddXdyAttribute{texttt}
1275 \GlsAddXdyAttribute{textbf}
1276 \GlsAddXdyAttribute{textmd}
1277 \GlsAddXdyAttribute{textit}
1278 \GlsAddXdyAttribute{textup}
1279 \GlsAddXdyAttribute{textsl}
1280 \GlsAddXdyAttribute{textsc}
1281 \GlsAddXdyAttribute{emph}
1282 \GlsAddXdyAttribute{glshypernumber}
1283 \GlsAddXdyAttribute{hyperrm}
1284 \GlsAddXdyAttribute{hypersf}
1285 \GlsAddXdyAttribute{hypertt}
1286 \GlsAddXdyAttribute{hyperbf}
1287 \GlsAddXdyAttribute{hypermd}
1288 \GlsAddXdyAttribute{hyperit}
1289 \GlsAddXdyAttribute{hyperup}
1290 \GlsAddXdyAttribute{hypersl}
1291 \GlsAddXdyAttribute{hypersc}
1292 \GlsAddXdyAttribute{hyperemph}

1293 \GlsAddXdyAttribute{glsignore}
1294 }
1295 \else
1296 \let\@gls@addpredefinedattributes\relax
1297 \fi

```

dyuseralphabets List of additional alphabets

```
1298 \def\@xdyuseralphabets{}
```

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

```

1299 \ifglsxindy
1300 \newcommand*\GlsAddXdyAlphabet[2]{%
1301 \edef\@xdyuseralphabets{%
1302 \@xdyuseralphabets ^^J
1303 (define-alphabet "#1" (#2))}}
1304 \else
1305 \newcommand*\GlsAddXdyAlphabet[2]{%
1306 \glsnoxindywarning\GlsAddXdyAlphabet}
1307 \fi

```

This code is only required for xindy:

```
1308 \ifglsxindy
```

dy@locationlist List of predefined location names.

```
1309 \newcommand*\@gls@xdy@locationlist{%
```

```

1310     roman-page-numbers,%
1311     Roman-page-numbers,%
1312     arabic-page-numbers,%
1313     alpha-page-numbers,%
1314     Alpha-page-numbers,%
1315     Appendix-page-numbers,%
1316     arabic-section-numbers%
1317 }

```

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.

```

1318 \protected@edef\@gls@roman{\@roman{0}\string"
1319     \string"roman-numbers-lowercase\string" :sep \string"}}%
1320 \@onelevel@sanitize\@gls@roman
1321 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1322     :sep \string"}%
1323 \@onelevel@sanitize\@tmp
1324 \ifx\@tmp\@gls@roman
1325     \expandafter
1326     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1327         \string"roman-numbers-lowercase\string"%
1328     }%
1329 \else
1330     \expandafter
1331     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1332         :sep \string"\@gls@roman\string"%
1333     }%
1334 \fi

```

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

```

1335 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1336     \string"roman-numbers-uppercase\string"%
1337 }%

```

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

```

1338 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1339     \string"arabic-numbers\string"%
1340 }%

```

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

```

1341 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1342     \string"alpha\string"%
1343 }%

```


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

```

1344 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1345     \string"ALPHA\string"%
1346 }%
```

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

```

1347 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1348     \string"ALPHA\string"
1349     :sep \string"\glsAlphacompositor\string"
1350     \string"arabic-numbers\string"%
1351 }
```

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

```

1352 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1353     \string"arabic-numbers\string"
1354     :sep \string"\glscompositor\string"
1355     \string"arabic-numbers\string"%
1356 }%
```

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

```

1357 \def\@xdyuserlocationdefs{}
```

erlocationnames List of additional user location names

```

1358 \def\@xdyuserlocationnames{}
```

End of xindy-only block:

```

1359 \fi
```

xdycrossrefhook Hook used after writing cross-reference class information.

```

1360 \ifglsxindy
1361 \newcommand\@xdycrossrefhook{}
1362 \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.)

```

1363 \ifglsxindy
1364 \newcommand*\GlsAddXdyLocation{[3][[]]{%
1365     \def\@gls@tmp{#1}%
1366     \ifx\@gls@tmp\@empty
1367         \edef\@xdyuserlocationdefs{%
1368             \@xdyuserlocationdefs ^^J%
1369             (define-location-class \string"#2\string"^^J\space\space
1370             \space(:sep \string"{}\glsopenbrace\string" #3
1371             :sep \string"\glsclosebrace\string"))
1372     }%
```

```

1373     \else
1374         \edef\@xdyuserlocationdefs{%
1375             \@xdyuserlocationdefs ^^J%
1376             (define-location-class \string"#2\string"^^J\space\space
1377             \space(:sep "\glsoopenbrace"
1378                 #1
1379                 :sep "\glsclosebrace\glsoopenbrace" #3
1380                 :sep "\glsclosebrace"))
1381         }%
1382     \fi

1383     \edef\@xdyuserlocationnames{%
1384         \@xdyuserlocationnames^^J\space\space\space
1385         \string"#2\string"}%
1386 }

```

Only has an effect before \writeist:

```

1387 \@onlypremakeg\GlsAddXdyLocation
1388 \else
1389     \newcommand*\{GlsAddXdyLocation}[2]{%
1390         \glsnnoxindywarning\GlsAddXdyLocation}
1391 \fi

```

ationclassorder Define location class order

```

1392 \ifglxindy
1393 \def\@xdylocationclassorder{^^J\space\space\space
1394     \string"roman-page-numbers\string"^^J\space\space\space
1395     \string"arabic-page-numbers\string"^^J\space\space\space
1396     \string"arabic-section-numbers\string"^^J\space\space\space
1397     \string"alpha-page-numbers\string"^^J\space\space\space
1398     \string"Roman-page-numbers\string"^^J\space\space\space
1399     \string"Alpha-page-numbers\string"^^J\space\space\space
1400     \string"Appendix-page-numbers\string"
1401     \@xdyuserlocationnames^^J\space\space\space
1402     \string"see\string"
1403 }
1404 \fi

```

Change the location order.

ationClassOrder

```

1405 \ifglxindy
1406     \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1407         \def\@xdylocationclassorder{#1}}
1408 \else
1409     \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1410         \glsnnoxindywarning\GlsSetXdyLocationClassOrder}
1411 \fi

```

\@xdysortrules Define sort rules

```

1412 \ifglxindy
1413   \def\@xdysortrules{}
1414 \fi

```

`\GlsAddSortRule` Add a sort rule

```

1415 \ifglxindy
1416   \newcommand*\GlsAddSortRule[2]{%
1417     \expandafter\toks@\expandafter{\@xdysortrules}%
1418     \protected@edef\@xdysortrules{\the\toks@ ^^J
1419       (sort-rule \string"#1\string" \string"#2\string"))}%
1420   }
1421 \else
1422   \newcommand*\GlsAddSortRule[2]{%
1423     \glsnxindywarning\GlsAddSortRule}
1424 \fi

```

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

```

1425 \ifglxindy
1426   \def\@xdyrequiredstyles{tex}
1427 \fi

```

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

```

1428 \ifglxindy
1429   \newcommand*\GlsAddXdyStyle[1]{%
1430     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1431 \else
1432   \newcommand*\GlsAddXdyStyle[1]{%
1433     \glsnxindywarning\GlsAddXdyStyle}
1434 \fi

```

`\GlsSetXdyStyles` Reset the list of required styles

```

1435 \ifglxindy
1436   \newcommand*\GlsSetXdyStyles[1]{%
1437     \edef\@xdyrequiredstyles{#1}}
1438 \else
1439   \newcommand*\GlsSetXdyStyles[1]{%
1440     \glsnxindywarning\GlsSetXdyStyles}
1441 \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.

```

1442 \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.

```

1443 \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.

```

1444 \ifglxsindy
1445   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1446     \ifglossaryexists{#1}{%
1447       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1448     }{%
1449       \PackageError{glossaries}{Can't set language type for
1450         glossary type '#1' --- no such glossary}{%
1451         You have specified a glossary type that doesn't exist}}
1452 \else
1453   \newcommand*\GlsSetXdyLanguage[2][]{%
1454     \glsnoxywarning\GlsSetXdyLanguage}
1455 \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.

```

1456 \def\@gls@codepage#1#2{}

```

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

```

1457 \ifglxsindy
1458   \newcommand*\GlsSetXdyCodePage[1]{%
1459     \renewcommand*\@gls@codepage{#1}%
1460   }

```

Suggested by egreg:

```

1461 \AtBeginDocument{%
1462   \ifx\gls@codepage\@empty
1463     \@ifpackageloaded{fontspec}{\def\gls@codepage{utf8}}{}%
1464   \fi
1465 }
1466 \else
1467   \newcommand*\GlsSetXdyCodePage[1]{%
1468     \glsnoxywarning\GlsSetXdyCodePage}
1469 \fi

```

`\xdylettergroups` Store letter group definitions.

```

1470 \ifglxsindy
1471   \ifglsexindy@glslnumbers
1472     \def\xdylettergroups{(define-letter-group
1473       \string"glslnumbers\string"^^J\space\space\space
1474       :prefixes (\string"0\string" \string"1\string"
1475       \string"2\string" \string"3\string" \string"4\string"
1476       \string"5\string" \string"6\string" \string"7\string"
1477       \string"8\string" \string"9\string")^^J\space\space\space
1478       \@xdynumbergrouporder)}
1479   \else
1480     \def\xdylettergroups{}

```

```

1481 \fi
1482 \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.

```

1483 \newcommand*\GlsAddLetterGroup[2]{%
1484   \expandafter\toks@\expandafter{\@xdylettergroups}%
1485   \protected@edef\@xdylettergroups{\the\toks@^^J%
1486   (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1487 }%

```

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.

```

1488 \newcommand*\forallglossaries[3][\@glo@types]{%
1489   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1490 }

```

\forallacronyms

```

1491 \newcommand*\forallacronyms[2]{%
1492   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1493 }

```

\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.

```

1494 \newcommand*\forglsentries[3][\glsdefaulttype]{%
1495   \edef\@glo@list{\csname glolist@#1\endcsname}%
1496   \@for#2:=\@glo@list\do
1497   {%
1498     \ifdefempty{#2}{#3}%
1499   }%
1500 }

```

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>}
```

```

1501 \newcommand*{\forallglsentries}[3][\@glo@types]{%
1502   \expandafter\forallglossaries\expandafter[#1]{\@@this@glo@}%
1503   {%
1504     \forglsentries[\@@this@glo@]{#2}{#3}%
1505   }%
1506 }

```

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

```

1507 \newcommand{\ifglossaryexists}[3]{%
1508   \ifcsundef{@glotype@#1@out}{#3}{#2}%
1509 }

```

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

\glsdetoklabel

```
1510 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

```

1511 \newcommand{\ifglentryexists}[3]{%
1512   \ifcsundef{glo@glsetoklabel{#1}@name}{#3}{#2}%
1513 }

```

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

54

```

1514 \newcommand*{\ifglsused}[3]{%
1515   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1516 }

```

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

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

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

```

1517 \newcommand{\glsdoifexists}[2]{%
1518   \ifglsentryexists{#1}{#2}{%
1519     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1520     has not been defined}{You need to define a glossary entry before you
1521     can use it.}}%
1522 }

```

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

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

```

1523 \newcommand{\glsdoifnoexists}[2]{%
1524   \ifglsentryexists{#1}{%
1525     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1526     been defined}{}}{#2}%
1527 }

```

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

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

```

1528 \newcommand{\glsdoifexistsorwarn}[2]{%
1529   \ifglsentryexists{#1}{#2}{%
1530     \GlossariesWarning{Glossary entry '\glsdetoklabel{#1}'
1531     has not been defined}%
1532   }%
1533 }

```

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

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

```

1534 \newcommand{\glsdoifexistsordo}[3]{%
1535   \ifglsentryexists{#1}{#2}{%
1536     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1537     has not been defined}{You need to define a glossary entry before you
1538     can use it.}}{#3}%

```

```

1539     #3%
1540   }%
1541 }

```

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

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

```

1542 \newcommand{\doifglossarynoexistsordo}[3]{%
1543   \ifglossaryexists{#1}%
1544   {%
1545     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1546     #3%
1547   }%
1548   {#2}%
1549 }

```

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

```

1550 \newcommand{\ifglshaschildren}[3]{%
1551   \glstoifexists{#1}%
1552   {%
1553     \def\do@glshaschildren{#3}%
1554     \edef\@gls@thislabel{\glstoiflabel{#1}}%
1555     \expandafter\for@glstentries\expandafter
1556     [\csname glo@\@gls@thislabel @type\endcsname]
1557     {\glo@label}%
1558     {%
1559       \letcs\glo@parent{glo@\glo@label @parent}%
1560       \ifdefequal\@gls@thislabel\glo@parent
1561       {%
1562         \def\do@glshaschildren{#2}%
1563         \@endfortrue
1564       }%
1565       {}%
1566     }%
1567     \do@glshaschildren
1568   }%
1569 }

```

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

```

1570 \newcommand{\ifglshasparent}[3]{%
1571   \glstoifexists{#1}%
1572   {%
1573     \ifcempty{glo@\glstoiflabel{#1}@parent}{#3}{#2}%

```



```

1574 }%
1575 }

\ifglshasdesc \ifglshasdesc{<label>}{<true part>}{<false part>}
1576 \newcommand*{\ifglshasdesc}[3]{%
1577 \ifcsequal{glo@glstdetoklabel{#1}@desc}%
1578 {#3}%
1579 {#2}%
1580 }

sdescsuppressed \ifglstdescsuppressed{<label>}{<true part>}{<false part>} Does <true part> if the descrip-
tion is just \nopostdesc otherwise does <false part>.
1581 \newcommand*{\ifglstdescsuppressed}[3]{%
1582 \ifcsequal{glo@glstdetoklabel{#1}@desc}{@no@post@desc}%
1583 {#2}%
1584 {#3}%
1585 }

\ifglshassymbol \ifglshassymbol{<label>}{<true part>}{<false part>}
1586 \newcommand*{\ifglshassymbol}[3]{%
1587 \letcs{\@glo@symbol}{glo@glstdetoklabel{#1}@symbol}%
1588 \ifdefempty\@glo@symbol
1589 {#3}%
1590 {%
1591 \ifdefequal\@glo@symbol\@gls@default@value
1592 {#3}%
1593 {#2}%
1594 }%
1595 }

\ifglshaslong \ifglshaslong{<label>}{<true part>}{<false part>}
1596 \newcommand*{\ifglshaslong}[3]{%
1597 \letcs{\@glo@long}{glo@glstdetoklabel{#1}@long}%
1598 \ifdefempty\@glo@long
1599 {#3}%
1600 {%
1601 \ifdefequal\@glo@long\@gls@default@value
1602 {#3}%
1603 {#2}%
1604 }%
1605 }

\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1606 \newcommand*{\ifglshasshort}[3]{%
1607 \letcs{\@glo@short}{glo@glstdetoklabel{#1}@short}%
1608 \ifdefempty\@glo@short
1609 {#3}%
1610 {%

```

```

1611 \ifdefequal\@glo@short\@gls@default@value
1612 {#3}%
1613 {#2}%
1614 }%
1615 }

```

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

```

1616 \newcommand*{\ifglshasfield}[4]{%
1617 \glsdoifexists{#2}%
1618 {%
1619 \letcs{\@glo@thisvalue}{glo\glsdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```

1620 \ifdef\@glo@thisvalue
1621 {%

```

Is defined, so now check if empty.

```

1622 \ifdefempty\@glo@thisvalue
1623 {%

```

Is empty, so doesn't have field set.

```

1624 #4%
1625 }%
1626 {%

```

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

```

1627 \ifdefequal\@glo@thisvalue\@gls@default@value
1628 {%

```

Value is set to the default value.

```

1629 #4%
1630 }%
1631 {%

```

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

```

1632 \let\glscurrentfieldvalue\@glo@thisvalue
1633 #3%
1634 }%
1635 }%
1636 }%
1637 {%

```

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

```
1638 \@gls@fetchfield{\@gls@thisfield}{#1}%
```

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

```

1639 \ifdef\@gls@thisfield
1640 {%

```

Is defined, so now check if empty.

```
1641      \letcs{\@glo@thisvalue}{glo@\glsdetoklabel{#2}@\@gls@thisfield}%
1642      \ifdefempty\@glo@thisvalue
1643      {%
```

Is empty so field hasn't been set.

```
1644          #4%
1645      }%
1646      {%
```

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

```
1647      \ifdefequal\@glo@thisvalue\@gls@default@value
1648      {%
```

Value is set to the default value.

```
1649          #4%
1650      }%
1651      {%
```

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

```
1652      \let\glscurrentfieldvalue\@glo@thisvalue
1653      #3%
1654      }%
1655      }%
1656      }%
1657      {%
```

Not defined.

```
1658      \GlossariesWarning{Unknown entry field '#1'}%
1659      #4%
1660      }%
1661      }%
1662      }%
1663 }
```

urrentfieldvalue

```
1664 \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

```
1665 \newcommand*{\@glo@types}{,}
```

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

```
1666 \newcommand*\@gls@provide@newglossary{%
1667   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%

```

Only need to do this once.

```
1668   \let\@gls@provide@newglossary\relax
1669 }
```

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

```
1670 \newcommand*\defglsentryfmt}[2][\glsdefaulttype]{%
1671   \csgdef{gls@#1@entryfmt}{#2}%
1672 }
```

`\gls@doentryfmt`

```
1673 \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.

```
1674 \newcommand*\@gls@forbidtexext}[1]{%
1675   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1676               or test {\ifdefstring{#1}{TEX}}}
1677   {%
1678     \def#1{nottex}%
1679     \PackageError{glossaries}%
1680       {Forbidden '.tex' extension replaced with '.nottex'}%
1681       {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1682         Don't use '.tex' as an extension for a temporary file.}%
1683   }%
1684   {%
1685   }%
1686 }
```

`\gls@gobbleopt` Discard optional argument.

```
1687 \newcommand*\gls@gobbleopt{}\new@ifnextchar[{\@gls@gobbleopt}{}
1688 \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`

```
1689 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}
```

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

```
1690 \newcommand*{\s@newglossary}[2]{%
1691   \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1692 }
```

`\ns@newglossary` Define the unstarred version.

```
1693 \newcommand*{\ns@newglossary}[5][glg]{%
1694   \doifglossarynoexistsordo{#2}%
1695   {%
```

Check if default has been set

```
1696   \ifundef\glsdefaultttype
1697   {%
1698     \gdef\glsdefaultttype{#2}%
1699   }{}}%
```

Add this to the list of glossary types:

```
1700   \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.

```
1701   \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1702   \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1703   \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1704   \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1705   \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
1706   \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
1707   \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1708   \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
```

```
1709   \@gls@provide@newglossary
```

```
1710   \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.

```
1711   \ifcsundef{gls@#2@entryfmt}%
1712   {%
1713     \defglsentryfmt[#2]{\glsentryfmt}%
1714   }%
1715   {}%
```

Define sort counter if required:

```
1716 \@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.)

```
1717 \ifnextchar[{\@gls@setcounter{#2}}{%
1718   {\@gls@setcounter{#2}[\glscounter]}}%
1719 }%
1720 {%
1721   \gls@gobbleopt
1722 }%
1723 }
```

`\altnewglossary`

```
1724 \newcommand*{\altnewglossary}[3]{%
1725   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1726 }
```

Only define new glossaries in the preamble:

```
1727 \@onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
1728 \@onlypremakeg\newglossary
```

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

`\@newglossary`

```
1729 \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`

```
1730 \def\@gls@setcounter#1[#2]{%
1731   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%
```

Add counter to xindy list, if not already added:

```
1732 \ifglsxindy
1733   \GlsAddXdyCounters{#2}%
1734 \fi
1735 }
```

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

`@gls@getcounter`

```
1736 \newcommand*{\@gls@getcounter}[1]{%
1737   \csname @glotype@#1@counter\endcsname
1738 }
```

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

```
1739 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1740 \@gls@do@acronymsdef
```

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

```
1741 \@gls@do@symbolsdef
```

```
1742 \@gls@do@numbersdef
```

```
1743 \@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.

```
1744 \newcommand*{\newignoredglossary}[1]{%
1745   \ifdefempty\@ignored@glossaries
1746   {%
1747     \edef\@ignored@glossaries{#1}%
1748   }%
1749   {%
1750     \eappto\@ignored@glossaries{,#1}%
1751   }%
1752   \csgdef{glolist@#1}{,}%
1753   \ifcsundef{gls@#1@entryfmt}%
1754   {%
1755     \defglsentryfmt[#1]{\glsentryfmt}%
1756   }%
1757   {}%
1758   \ifdefempty\@gls@nohyperlist
1759   {%
1760     \renewcommand*{\@gls@nohyperlist}{#1}%
1761   }%
1762   {%
1763     \eappto\@gls@nohyperlist{,#1}%
1764   }%
1765 }
```

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

```
1766 \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.

```
1767 \newcommand*{\ifignoredglossary}[3]{%
1768   \edef\@gls@igtype{#1}%
1769   \expandafter\DTLifinlist\expandafter
1770   {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1771 }
```

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.

```
1772 \define@key{glossentry}{name}{%  
1773 \def\@glo@name{#1}%  
1774 }
```

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`.

```
1775 \define@key{glossentry}{description}{%  
1776 \def\@glo@desc{#1}%  
1777 }
```

descriptionplural

```
1778 \define@key{glossentry}{descriptionplural}{%  
1779 \def\@glo@descplural{#1}%  
1780 }
```

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`.

```
1781 \define@key{glossentry}{sort}{%  
1782 \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.

```
1783 \define@key{glossentry}{text}{%  
1784 \def\@glo@text{#1}%  
1785 }
```

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.

```
1786 \define@key{glossentry}{plural}{%  
1787 \def\@glo@plural{#1}%  
1788 }
```


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.

```
1789 \define@key{glossentry}{first}{%
1790 \def\@glo@first{#1}%
1791 }
```

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.

```
1792 \define@key{glossentry}{firstplural}{%
1793 \def\@glo@firstplural{#1}%
1794 }
```

s@default@value

```
1795 \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).

```
1796 \define@key{glossentry}{symbol}{%
1797 \def\@glo@symbol{#1}%
1798 }
```

symbolplural

```
1799 \define@key{glossentry}{symbolplural}{%
1800 \def\@glo@symbolplural{#1}%
1801 }
```

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

```
1802 \define@key{glossentry}{type}{%
1803 \def\@glo@type{#1}}
```

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

```
1804 \define@key{glossentry}{counter}{%
1805   \ifcsundef{c@#1}%
1806   {%
1807     \PackageError{glossaries}%
1808     {There is no counter called ‘#1’}%
1809     {%
1810       The counter key should have the name of a valid counter
1811       as its value%
1812     }%
1813   }%
```

```

1814  {%
1815    \def\@glo@counter{#1}%
1816  }%
1817 }

```

see The see key specifies a list of cross-references

```

1818 \define@key{glossentry}{see}{%
1819   \gls@set@xr@key{see}{\@glo@see}{#1}%
1820 }

```

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

Assign a cross-reference key.

```

1821 \newcommand*{\gls@set@xr@key}[3]{%
1822   \renewcommand*{\gls@xr@key}{#1}%
1823   \gls@checkseeallowed
1824   \def#2{#3}%
1825   \@glo@seeautonumberlist
1826 }

```

\gls@xr@key

```

1827 \newcommand*{\gls@xr@key}{see}

```

checkseeallowed

```

1828 \newcommand*{\gls@checkseeallowed}{%
1829   \@gls@see@noindex
1830 }

```

ed@preambleonly

```

1831 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1832   \GlossariesWarning{glossaries}%
1833   {'\gls@xr@key' key doesn't have any effect when used in the document
1834     environment. Move the definition to the preamble
1835     after \string\makeglossaries\space
1836     or \string\makenoidxglossaries}%
1837 }

```

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

```

1838 \define@key{glossentry}{parent}{%
1839 \def\@glo@parent{#1}}

```

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

```

1840 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1841   \ifcase\nr\relax
1842     \def\@glo@prefix{\glsnonextpages}%
1843     \@gls@savenonumberlist{true}%
1844   \else

```

```

1845 \def\@glo@prefix{\glsnextpages}%
1846 \@gls@savenonumberlist{false}%
1847 \fi
1848 }

```

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.

```

1849 \newcommand*{\@gls@savenonumberlist}[1]{%

```

nitnonumberlist

```

1850 \newcommand*{\@gls@initnonumberlist}{}%

```

nitnonumberlist

```

1851 \newcommand*{\@gls@storenonumberlist}[1]{%

```

savenonumberlist Allow the nonumberlist value to be saved.

```

1852 \newcommand*{\@gls@enablesavenonumberlist}{%
1853 \renewcommand*{\@gls@initnonumberlist}{%
1854 \undef\@glo@nonumberlist
1855 }%
1856 \renewcommand*{\@gls@savenonumberlist}[1]{%
1857 \def\@glo@nonumberlist{##1}%
1858 }%
1859 \renewcommand*{\@gls@storenonumberlist}[1]{%
1860 \ifdef\@glo@nonumberlist
1861 {%
1862 \cslet{glo@glstetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
1863 }%
1864 }%
1865 }%
1866 \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1867 }

```

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

user1

```

1868 \define@key{glossentry}{user1}{%
1869 \def\@glo@useri{#1}%
1870 }

```

user2

```

1871 \define@key{glossentry}{user2}{%
1872 \def\@glo@userii{#1}%
1873 }

```

user3

```
1874 \define@key{glossentry}{user3}{%  
1875   \def\@glo@useriii{#1}%  
1876 }
```

user4

```
1877 \define@key{glossentry}{user4}{%  
1878   \def\@glo@useriv{#1}%  
1879 }
```

user5

```
1880 \define@key{glossentry}{user5}{%  
1881   \def\@glo@userv{#1}%  
1882 }
```

user6

```
1883 \define@key{glossentry}{user6}{%  
1884   \def\@glo@uservi{#1}%  
1885 }
```

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.

```
1886 \define@key{glossentry}{short}{%  
1887   \def\@glo@short{#1}%  
1888 }
```

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

```
1889 \define@key{glossentry}{shortplural}{%  
1890   \def\@glo@shortpl{#1}%  
1891 }
```

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

```
1892 \define@key{glossentry}{long}{%  
1893   \def\@glo@long{#1}%  
1894 }
```

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

```
1895 \define@key{glossentry}{longplural}{%  
1896   \def\@glo@longpl{#1}%  
1897 }
```

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

```
1898 \newcommand*{\@glsnname}{%  
1899   \PackageError{glossaries}{name key required in  
1900   \string\newglossaryentry\space for entry '@glo@label'}{You  
1901   haven't specified the entry name}}
```

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

```

1902 \newcommand*\@glsnodesc{%
1903   \PackageError{glossaries}
1904   {%
1905     description key required in \string\newglossaryentry\space
1906     for entry ‘\@glo@label’%
1907   }%
1908   {%
1909     You haven’t specified the entry description%
1910   }%
1911 }%

```

`lsdefaultplural` Now obsolete. Don’t use.

```

1912 \newcommand*\@glsdefaultplural{}

```

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

```

1913 \newcommand*\@gls@missingnumberlist}[1]{%
1914   ??%
1915   \ifglssavenumberlist
1916     \GlossariesWarning{Missing number list for entry ‘#1’.
1917       Maybe makeglossaries + rerun required}%
1918   \else
1919     \PackageError{glossaries}%
1920     {Package option ‘savenumberlist=true’ required}%
1921     {%
1922       You must use the ‘savenumberlist’ package option
1923       to reference location lists.%
1924     }%
1925   \fi
1926 }

```

`@glsdefaultsort` Define command to set default sort.

```

1927 \newcommand*\@glsdefaultsort{\@glo@name}

```

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

```

1928 \newcount\gls@level

```

`@noexpand@field`

```

1929 \newcommand{\@@gls@noexpand@field}[3]{%
1930   \expandafter\global\expandafter
1931   \let\csname glo@#1@#2\endcsname#3%
1932 }

```

`noexpand@fields`

```

1933 \newcommand{\@gls@noexpand@fields}[4]{%
1934   \ifcsdef{gls@assign@#3@field}
1935   {%
1936     \ifdefequal{#4}{\@gls@default@value}%

```

```

1937    {%
1938        \edef\@gls@value{\expandonce{#1}}%
1939        \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1940    }%
1941    {%
1942        \csuse{gls@assign@#3@field}{#2}{#4}%
1943    }%
1944 }%
1945 {%
1946     \ifdefequal{#4}{\@gls@default@value}%
1947     {%
1948         \edef\@gls@value{\expandonce{#1}}%
1949         \@gls@noexpand@field{#2}{#3}{\@gls@value}%
1950     }%
1951     {%
1952         \@gls@noexpand@field{#2}{#3}{#4}%
1953     }%
1954 }%
1955 }

```

ls@expand@field

```

1956 \newcommand{\@gls@expand@field}[3]{%
1957     \expandafter
1958     \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1959 }

```

s@expand@fields

```

1960 \newcommand{\@gls@expand@fields}[4]{%
1961     \ifcsdef{gls@assign@#3@field}
1962     {%
1963         \ifdefequal{#4}{\@gls@default@value}%
1964         {%
1965             \edef\@gls@value{\expandonce{#1}}%
1966             \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1967         }%
1968         {%
1969             \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
1970             {%
1971                 \@gls@expand@field{#2}{#3}{#4}%
1972             }%
1973             {%
1974                 \csuse{gls@assign@#3@field}{#2}{#4}%
1975             }%
1976         }%
1977     }%
1978     {%
1979         \ifdefequal{#4}{\@gls@default@value}%
1980         {%

```

```

1981      \@gls@expand@field{#2}{#3}{#1}%
1982    }%
1983    {%
1984      \@gls@expand@field{#2}{#3}{#4}%
1985    }%
1986  }%
1987 }

```

switexpandonce

```

1988 \def\@gls@expandonce{\expandonce}
1989 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1990   \def\@gls@tmp{#1}%
1991   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1992 }

```

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.

```
1993 \let\gls@assign@field\@gls@expand@fields
```

glsexpandfields

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

```

1994 \newcommand*{\glsexpandfields}{%
1995   \let\gls@assign@field\@gls@expand@fields
1996 }

```

snoexpandfields

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

```

1997 \newcommand*{\glsnoexpandfields}{%
1998   \let\gls@assign@field\@gls@noexpand@fields
1999 }

```

newglossaryentry

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

```
2000 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```

2001   \glsdoifnoexists{#1}%
2002   {%
2003     \gls@defglossaryentry{#1}{#2}%
2004   }%
2005 }

```

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.

```

2006 \newcommand*{\gls@defdocnewglossaryentry}{%
2007   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
2008   \let\newglossaryentry\new@glossaryentry
2009 }

```

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

```

2010 \newrobustcmd{\provideglossaryentry}[2]{%
2011   \ifglstryexists{#1}%
2012   {}%
2013   {%
2014     \gls@defglossaryentry{#1}{#2}%
2015   }%
2016 }
2017 \@onlypreamble{\provideglossaryentry}

```

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

```

2018 \newrobustcmd{\new@glossaryentry}[2]{%
2019   \ifundef\@gls@deffile
2020   {%
2021     \global\newwrite\@gls@deffile
2022     \immediate\openout\@gls@deffile=\jobname.glsdefs
2023   }%
2024   {}%
2025   \ifglstryexists{#1}{}%
2026   {%
2027     \gls@defglossaryentry{#1}{#2}%
2028   }%
2029   \@gls@writedef{#1}%
2030 }

```

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`.

```

2031 \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.

```

2032 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}

```

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

```

2033 \newcommand*{\gls@begindocdefs}{%
2034   \@gls@enablesavenonumberlist
2035   \edef\@gls@restoreat{\noexpand\catcode'\noexpand\@=\number\catcode'\@relax}%
2036   \makeatletter
2037   \InputIfFileExists{\jobname.glsdefs}{}{}%
2038   \@gls@restoreat

```



```

2039 \undef\@gls@restoreat
2040 \gls@defdocnewglossaryentry
2041 }

```

\@gls@writedef Write glossary entry definition to \@gls@deffile.

```

2042 \newcommand*{\@gls@writedef}[1]{%
2043   \immediate\write\@gls@deffile
2044   {%
2045     \string\ifglsentryexists{#1}{}\glspercentchar^^J%
2046     \expandafter\@gobble\string\{\glspercentchar^^J%
2047     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
2048     \expandafter\@gobble\string\{\glspercentchar%
2049   }%

  Write key value information:

2050   \@for\@gls@map:=\@gls@keymap\do
2051   {%
2052     \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
2053     \ifdef\glo@value
2054     {%
2055       \@onelevel@sanitize\glo@value
2056       \immediate\write\@gls@deffile
2057       {%
2058         \expandafter\@firstoftwo\@gls@map
2059         =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
2060         \glspercentchar
2061       }%
2062     }%
2063   }%
2064 }%

```

Provide hook:

```

2065 \gls@writedefhook
2066 \immediate\write\@gls@deffile
2067 {%
2068   \glspercentchar^^J%
2069   \expandafter\@gobble\string\}\glspercentchar^^J%
2070   \expandafter\@gobble\string\}\glspercentchar%
2071 }%
2072 }

```

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

```

2073 \newcommand*{\@gls@keymap}{%
2074   {name}{name},%
2075   {sort}{sortvalue},% unescaped sort value
2076   {type}{type},%
2077   {first}{first},%
2078   {firstplural}{firstpl},%
2079   {text}{text},%

```

```

2080 {plural}{plural},%
2081 {description}{desc},%
2082 {descriptionplural}{descplural},%
2083 {symbol}{symbol},%
2084 {symbolplural}{symbolplural},%
2085 {user1}{useri},%
2086 {user2}{userii},%
2087 {user3}{useriii},%
2088 {user4}{useriv},%
2089 {user5}{userv},%
2090 {user6}{uservi},%
2091 {long}{long},%
2092 {longplural}{longpl},%
2093 {short}{short},%
2094 {shortplural}{shortpl},%
2095 {counter}{counter},%
2096 {parent}{parent}%
2097 }

```

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

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

```
2098 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
2099 \edef\@gls@thisval{#2}%
```

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

```

2100 \@for\@gls@map:=\@gls@keymap\do{%
2101 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2102 \ifdefequal{\@this@key}{\@gls@thisval}%
2103 {%

```

Found it.

```
2104 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```

2105 \@endfortrue
2106 }%
2107 {}%
2108 }%
2109 }

```

`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.

```
2110 \newcommand*{\glsaddstoragekey}{\@ifstar\sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```

2111 \newcommand*{\@sglsaddstoragekey}[1]{%
2112   \key@ifundefined{glossentry}{#1}%
2113   {%
2114     \expandafter\newcommand\expandafter*\expandafter
2115       {\csname gls@assign@#1@field\endcsname}[2]{%
2116         \@gls@expand@field{##1}{#1}{##2}%
2117       }%
2118   }%
2119   }%
2120   \@glsaddstoragekey{#1}%
2121 }
```

Unstarred version doesn't override default expansion.

```

2122 \newcommand*{\@glsaddstoragekey}[3]{%
```

Check the specified key doesn't already exist.

```

2123   \key@ifundefined{glossentry}{#1}%
2124   {%
```

Set up the key.

```

2125     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2126     \appto\@gls@keymap{,{#1}{#1}}%
```

Set the default value.

```

2127     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```

2128     \appto\@newglossaryentryposthook{%
2129       \letcs{\@glo@tmp}{@glo@#1}%
2130       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2131     }%
```

Define the no-link commands.

```

2132     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2133   }%
2134   {%
2135     \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2136   }%
2137 }
```

\glsaddkey	$\backslash\mathrm{glsaddkey}\{\langle key\rangle\}\{\langle default\ value\rangle\}\{\langle no\ link\ cs\rangle\}\{\langle no\ link\ ucfirst\ cs\rangle\}$ $\{\langle link\ cs\rangle\}\{\langle link\ ucfirst\ cs\rangle\}\{\langle link\ allcaps\ cs\rangle\}$
------------	--

Allow user to add their own custom keys.

```

2138 \newcommand*{\glsaddkey}\@ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```

2139 \newcommand*{\@sglsaddkey}[1]{%
2140   \key@ifundefined{glossentry}{#1}%
```

```

2141  {%
2142    \expandafter\newcommand\expandafter*\expandafter
2143      {\csname gls@assign@#1@field\endcsname}[2]{%
2144        \@gls@expand@field{##1}{#1}{##2}%
2145      }%
2146  }%
2147  {}%
2148  \@gls@saddkey{#1}%
2149 }

```

Unstarred version doesn't override default expansion.

```

2150 \newcommand*{\@gls@saddkey}[7]{%

```

Check the specified key doesn't already exist.

```

2151  \key@ifundefined{glossentry}{#1}%
2152  {%

```

Set up the key.

```

2153    \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2154    \appto\@gls@keymap{,{#1}{#1}}%

```

Set the default value.

```

2155    \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

2156    \appto\@newglossaryentryposthook{%
2157      \letcs{\@glo@tmp}{@glo@#1}%
2158      \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2159    }%

```

Define the no-link commands.

```

2160    \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2161    \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2162    \ifcsdef{@gls@user@#1@}%
2163    {%
2164      \PackageError{glossaries}%
2165        {Can't define '\string#5' as helper command
2166         '\expandafter\string\csname @gls@user@#1\endcsname' already exists}%
2167      }%
2168    }%
2169    {%
2170      \expandafter\newcommand\expandafter*\expandafter
2171        {\csname @gls@user@#1\endcsname}[2][ ]{%
2172          \new@ifnextchar[%
2173            {\csuse{@gls@user@#1@}{##1}{##2}}%
2174            {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
2175      \csdef{@gls@user@#1@}##1##2[##3]{%
2176        \@gls@field@link{##1}{##2}{#3{##2}##3}%
2177      }%

```

```

2178     \newrobustcmd*{#5}{%
2179     \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2180 }%

```

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

```

2181     \ifcsdef{@Gls@user@#1@}%
2182     {%
2183     \PackageError{glossaries}%
2184     {Can't define '\string#6' as helper command
2185     '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
2186     }%
2187 }%
2188 {%
2189     \expandafter\newcommand\expandafter*\expandafter
2190     {\csname @Gls@user@#1\endcsname}[2][ ]{%
2191     \new@ifnextchar[%
2192     {\csuse{@Gls@user@#1@}{##1}{##2}}%
2193     {\csuse{@Gls@user@#1@}{##1}{##2}[ ]}%
2194     \csdef{@Gls@user@#1@}##1##2[##3]{%
2195     \@gls@field@link{##1}{##2}{#4{##2}##3}%
2196     }%
2197     \newrobustcmd*{#6}{%
2198     \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2199 }%

```

Finally the all caps version:

```

2200     \ifcsdef{@GLS@user@#1@}%
2201     {%
2202     \PackageError{glossaries}%
2203     {Can't define '\string#7' as helper command
2204     '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2205     }%
2206 }%
2207 {%
2208     \expandafter\newcommand\expandafter*\expandafter
2209     {\csname @GLS@user@#1\endcsname}[2][ ]{%
2210     \new@ifnextchar[%
2211     {\csuse{@GLS@user@#1@}{##1}{##2}}%
2212     {\csuse{@GLS@user@#1@}{##1}{##2}[ ]}%
2213     \csdef{@GLS@user@#1@}##1##2[##3]{%
2214     \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2215     }%
2216     \newrobustcmd*{#7}{%
2217     \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2218 }%
2219 }%
2220 {%
2221     \PackageError{glossaries}{Key '#1' already exists}{}%

```

```

2222 }%
2223 }

```

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

```

2224 \newcommand{\glsfieldxdef}[3]{%
2225   \glsdoifexists{#1}%
2226   {%
2227     \edef\@glo@label{\glsdetoklabel{#1}}%
2228     \ifcsdef{glo@\@glo@label @#2}%
2229     {%
2230       \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2231     }%
2232     {%
2233       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2234     }%
2235   }%
2236 }

```

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

```

2237 \newcommand{\glsfielddedef}[3]{%
2238   \glsdoifexists{#1}%
2239   {%
2240     \edef\@glo@label{\glsdetoklabel{#1}}%
2241     \ifcsdef{glo@\@glo@label @#2}%
2242     {%
2243       \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2244     }%
2245     {%
2246       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2247     }%
2248   }%
2249 }

```

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

```

2250 \newcommand{\glsfieldgdef}[3]{%
2251   \glsdoifexists{#1}%
2252   {%
2253     \edef\@glo@label{\glsdetoklabel{#1}}%
2254     \ifcsdef{glo@\@glo@label @#2}%
2255     {%

```

```

2256     \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2257 }%
2258 {%
2259     \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2260 }%
2261 }%
2262 }

```

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

```

2263 \newcommand{\glsfielddef}[3]{%
2264   \glsdoifexists{#1}%
2265   {%
2266     \edef\@glo@label{\glsdetoklabel{#1}}%
2267     \ifcsdef{glo@\@glo@label @#2}%
2268     {%
2269       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2270     }%
2271     {%
2272       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2273     }%
2274   }%
2275 }

```

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

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

```

2276 \newcommand{\glsfieldfetch}[3]{%
2277   \glsdoifexists{#1}%
2278   {%
2279     \edef\@glo@label{\glsdetoklabel{#1}}%
2280     \ifcsdef{glo@\@glo@label @#2}%
2281     {%
2282       \letcs#3{glo@\@glo@label @#2}%
2283     }%
2284     {%
2285       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2286     }%
2287   }%
2288 }

```

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

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

```

2289 \newcommand{\ifglsfieldeq}[5]{%
2290   \glsdoifexists{#1}%
2291   {%
2292     \edef\@glo@label{\glsdetoklabel{#1}}%
2293     \ifcsdef{glo@\@glo@label @#2}%
2294     {%
2295       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2296     }%
2297     {%
2298       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2299     }%
2300   }%
2301 }

```

`\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.

```

2302 \newcommand{\ifglsfieldddefeq}[5]{%
2303   \glsdoifexists{#1}%
2304   {%
2305     \edef\@glo@label{\glsdetoklabel{#1}}%
2306     \ifcsdef{glo@\@glo@label @#2}%
2307     {%
2308       \expandafter\ifdefstrequal
2309       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2310     }%
2311     {%
2312       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2313     }%
2314   }%
2315 }

```

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

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

```

2316 \newcommand{\ifglsfieldcseq}[5]{%
2317   \glsdoifexists{#1}%
2318   {%
2319     \edef\@glo@label{\glsdetoklabel{#1}}%
2320     \ifcsdef{glo@\@glo@label @#2}%
2321     {%
2322       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2323     }%
2324     {%
2325       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2326     }%

```



```
2327 }%
2328 }
```

gls.writedefhook

```
2329 \newcommand*{\gls.writedefhook}{}%
```

gls.assign@desc

```
2330 \newcommand*{\gls.assign@desc}[1]{%
2331   \gls.assign@field{#1}{desc}{\@glo@desc}%
2332   \gls.assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2333 }
```

newglossaryentry

```
2334 \newcommand{\longnewglossaryentry}[3]{%
2335   \glsdoifnoexists{#1}%
2336   {%
2337     \bgroup
2338     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2339     \long\def\@newglossaryentryprehook{%
2340       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2341       \@org@newglossaryentryprehook
2342     }%
2343     \renewcommand*{\gls.assign@desc}[1]{%
2344       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2345       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2346     }
2347     \gls@defglossaryentry{#1}{#2}%
2348   \egroup
2349 }%
2350 }
```

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

```
2351 \@onlypreamble{\longnewglossaryentry}
```

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

```
2352 \newcommand{\longprovideglossaryentry}[3]{%
2353   \ifglstryexists{#1}{%
2354     {\longnewglossaryentry{#1}{#2}{#3}}%
2355   }
2356 \@onlypreamble{\longprovideglossaryentry}
```

defglossaryentry

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

Defines a new entry without checking if it already exists.

```
2357 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of \GlsSetQuote:

```
2358 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2359 \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2360 \let\glslabel\@glo@label
```

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

```
2361 \let\@glo@name\@gls@noname
```

```
2362 \let\@glo@desc\@gls@nodesc
```

```
2363 \let\@glo@descplural\@gls@default@value
```

```
2364 \let\@glo@type\@gls@default@value
```

```
2365 \let\@glo@symbol\@gls@default@value
```

```
2366 \let\@glo@symbolplural\@gls@default@value
```

```
2367 \let\@glo@text\@gls@default@value
```

```
2368 \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.)

```
2369 \let\@glo@first\@gls@default@value
```

```
2370 \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2371 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2372 \let\@glo@counter\@gls@default@value
```

```
2373 \def\@glo@see{}%
```

```
2374 \def\@glo@parent{}%
```

```
2375 \def\@glo@prefix{}%
```

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

```
2376 \@gls@initnonumberlist
```

```
2377 \def\@glo@useri{}%
```

```
2378 \def\@glo@userii{}%
```

```
2379 \def\@glo@useriii{}%
```

```
2380 \def\@glo@useriv{}%
```

```
2381 \def\@glo@userv{}%
```

```
2382 \def\@glo@uservi{}%
```

```

2383 \def\@glo@short{}%
2384 \def\@glo@shortpl{}%
2385 \def\@glo@long{}%
2386 \def\@glo@longpl{}%

```

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

```
2387 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2388 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```

2389 \ifundef\glsdefaulttype
2390 {%
2391   \PackageError{glossaries}%
2392     {No default glossary type (have you used ‘nomain’ by mistake?)}%
2393     {If you use package option ‘nomain’ you must define
2394      a new glossary before you can define entries}%
2395 }%
2396 {}%

```

Assign type. This must be fully expandable

```

2397 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2398 \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.

```

2399 \ifcsundef{glolist@\@glo@type}%
2400 {%
2401   \PackageError{glossaries}%
2402     {Glossary type ‘\@glo@type’ has not been defined}%
2403     {You need to define a new glossary type, before making entries
2404      in it}%
2405 }%
2406 {}%

```

Check if it's an ignored glossary

```

2407 \ifignoredglossary\@glo@type
2408 {%

```

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

```

2409   \ifx\@glo@desc\@glsnodel
2410     \let\@glo@desc\@empty
2411   \fi
2412 }%
2413 {%
2414 }%
2415 \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
2416 \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
2417   \@glolist@{\@glo@label},}%
2418 }%

```

Initialise level to 0.

```

2419 \gls@level=0\relax

```

Has this entry been assigned a parent?

```

2420 \ifx\@glo@parent\@empty

```

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

```

2421 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2422 \else

```

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

```

2423 \ifdefequal\@glo@label\@glo@parent%
2424 {%
2425 \PackageError{glossaries}{Entry '@glo@label' can't be its own parent}{}%
2426 \def\@glo@parent{}%
2427 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2428 }%
2429 {%

```

Check the parent exists:

```

2430 \ifglentryexists{\@glo@parent}%
2431 {%

```

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

```

2432 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2433 \@glo@parent}%

```

Determine level.

```

2434 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2435 \advance\gls@level by 1\relax

```

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

```

2436 \ifx\@glo@name\@gls@name
2437 \expandafter\let\expandafter\@glo@name
2438 \csname glo@\@glo@parent @name\endcsname

```

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

```

2439 \ifx\@glo@plural\@gls@default@value
2440 \expandafter\let\expandafter\@glo@plural
2441 \csname glo@\@glo@parent @plural\endcsname
2442 \fi
2443 \fi
2444 }%
2445 {%

```

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

```

2446 \PackageError{glossaries}%
2447 {%
2448 Invalid parent '@glo@parent'
2449 for entry '@glo@label' - parent doesn't exist%
2450 }%
2451 {%
2452 Parent entries must be defined before their children%

```

```

2453     }%
2454     \def\@glo@parent{}%
2455     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2456 }%
2457 }%
2458 \fi

Set the level for this entry
2459 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

Define commands associated with this entry:
2460 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2461 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2462 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2463 \expandafter\gls@assign@field\expandafter
2464     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2465     {\@glo@label}{plural}{\@glo@plural}%
2466 \expandafter\gls@assign@field\expandafter
2467     {\csname glo@\@glo@label @text\endcsname}%
2468     {\@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.
2469 \ifx\@glo@first\@gls@default@value
2470     \expandafter\gls@assign@field\expandafter
2471         {\csname glo@\@glo@label @plural\endcsname}%
2472         {\@glo@label}{firstpl}{\@glo@firstplural}%
2473 \else
2474     \expandafter\gls@assign@field\expandafter
2475         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2476         {\@glo@label}{firstpl}{\@glo@firstplural}%
2477 \fi

2478 \ifcsundef{@glotype@\@glo@type @counter}%
2479 {%
2480     \def\@glo@defaultcounter{\glscounter}%
2481 }%
2482 {%
2483     \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
2484 }%
2485 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2486 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2487 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2488 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2489 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2490 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2491 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2492 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2493 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2494 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2495 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%

```

```

2496 \ifx\@glo@name\@glsnoname
2497   \@glsnoname
2498   \let\@gloname\@gls@default@value
2499 \fi
2500 \gls@assign@field{}\@glo@label{name}\@glo@name}%

```

Set default numberlist if not defined:

```

2501 \ifcsundef{glo@\@glo@label @numberlist}%
2502 {%
2503   \csxdef{glo@\@glo@label @numberlist}{%
2504     \noexpand\@gls@missingnumberlist{\@glo@label}}%
2505   }%
2506   {}%

```

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

```

2507 \@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.

```

2508 \def\@glo@@desc{\@glo@first}%
2509 \ifx\@glo@desc\@glo@@desc
2510   \let\@glo@desc\@glo@first
2511 \fi
2512 \ifx\@glo@desc\@glsnodelc
2513   \@glsnodelc
2514   \let\@glodelc\@gls@default@value
2515 \fi
2516 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2517 \@gls@defsort{\@glo@type}\@glo@label}%

2518 \def\@glo@@symbol{\@glo@text}%
2519 \ifx\@glo@symbol\@glo@@symbol
2520   \let\@glo@symbol\@glo@text
2521 \fi
2522 \gls@assign@field{relax}\@glo@label{symbol}\@glo@symbol}%
2523 \expandafter
2524   \gls@assign@field\expandafter
2525   {\csname glo@\@glo@label @symbol\endcsname}
2526   {\@glo@label{symbolplural}\@glo@symbolplural}%

```

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

```

2527 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2528   \noexpand\global
2529   \noexpand\let\expandafter\noexpand
2530   \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2531   }%
2532 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2533   \noexpand\global

```

```

2534      \noexpand\let\expandafter\noexpand
2535      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2536  }%
2537  \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```

2538  \@glo@autosee

```

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

```

2539  \ifignoredglossary\@glo@type
2540  {%
2541    \csdef{glo@\@glo@label @index}{}%
2542  }
2543  {%
2544    \do@glo@storeentry{\@glo@label}%
2545  }%

```

Define entry counters if enabled:

```

2546  \@newglossaryentry@defcounters

```

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

```

2547  \@newglossaryentryposthook
2548 }

```

\@glo@autosee Automatically implement \glssee.

```

2549 \newcommand*{\@glo@autosee}{%
2550   \ifdefvoid\@glo@see{%
2551     {%
2552       \protected@edef\@do@glssee{%
2553         \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2554         \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
2555       \@do@glssee
2556     }%
2557     \@glo@autoseehook
2558 }%

```

glo@autoseehook

```

2559 \newcommand*{\@glo@autoseehook}{}

```

aryentryprehook Allow extra information to be added to glossary entries:

```

2560 \newcommand*{\@newglossaryentryprehook}{}

```

ryentryposthook Allow extra information to be added to glossary entries:

```

2561 \newcommand*{\@newglossaryentryposthook}{}

```

try@defcounters

```

2562 \newcommand*{\@newglossaryentry@defcounters}{}

```

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

```

2563 \newcommand*{\glsmoveentry}[2]{%
2564   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2565   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2566   \def\glo@list{,%}
2567   \forglsentries[\glo@type]{\glo@label}%
2568   {%

2569     \ifdefequal\@glo@thislabel\glo@label
2570       {\eappto\glo@list{\glo@label,}}%
2571     }%
2572     \cslet\glo@list@\glo@type{\glo@list}%
2573     \csdef{glo@\@glo@thislabel @type}{#2}%
2574 }

```

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

```

2575 \ifglxindy
2576   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2577 \else
2578   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2579 \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.)

```

2580 \ifglxindy
2581   \newcommand*{\@glossarysubentryfield}{%
2582     \string\subglossentry}
2583 \else
2584   \newcommand*{\@glossarysubentryfield}{%
2585     \string\subglossentry}
2586 \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.)

```

2587 \newcommand{\@glo@storeentry}[1]{%

```

Escape makeindex/xindy special characters in the label:

```

2588   \edef\@glo@esclabel{#1}%
2589   \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters


```

2590 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2591 \@gls@checkmkidxchars\@glo@sort
    Same again for the name string. Escape any special characters in the prefix
2592 \@gls@checkmkidxchars\@glo@prefix
    Get the parent, if one exists
2593 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
    Write the information to the glossary file.
2594 \ifglxsindy
    Store using xindy syntax.
2595 \ifx\@glo@parent\@empty
    Entry doesn't have a parent
2596 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2597 (\string"\@glo@sort\string" %
2598 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2599 }%
2600 \else
    Entry has a parent
2601 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2602 \csname glo@\@glo@parent @index\endcsname
2603 (\string"\@glo@sort\string" %
2604 \string"\@glo@prefix\@glossarysubentryfield
2605 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2606 }%
2607 \fi
2608 \else
    Store using makeindex syntax.
2609 \ifx\@glo@parent\@empty
    Sanitize \@glo@prefix
2610 \@onelevel@sanitize\@glo@prefix
    Entry doesn't have a parent
2611 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2612 \@glo@sort\@gls@actualchar\@glo@prefix
2613 \@glossaryentryfield{\@glo@esclabel}%
2614 }%
2615 \else
    Entry has a parent
2616 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2617 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2618 \@glo@sort\@gls@actualchar\@glo@prefix
2619 \@glossarysubentryfield
2620 {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2621 }%
2622 \fi
2623 \fi
2624 }

```

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`

```
2625 \AtBeginDocument{%
2626   \ifpackageloaded{amsmath}%
2627   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2628   }%
2629 }
2630 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2631   \ifmeasuring@
2632   \else
2633     #1%
2634   \fi
2635 }
2636 \newcommand*\gls@ifnotmeasuring[1]{#1}
```

`\glspatchtabularx` Patch `\TX@trial` (as per David Carlisle's answer in <http://tex.stackexchange.com/a/94895>). This does nothing if `\TX@trial` hasn't been defined.

```
2637 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2638   \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2639 }
2640 \newcommand*\glspatchtabularx{%
2641   \ifdef\TX@trial
2642   {%
2643     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2644     \let\glspatchtabularx\relax
2645   }%
2646   {%
2647 }
```

`\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.

```
2648 \newcommand*{\glsreset}[1]{%
2649   \gls@ifnotmeasuring
2650   {%
2651     \glsdoifexists{#1}%
2652     {%
2653       \@glsreset{#1}%
2654     }%
2655   }%
2656 }
```

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

```

2657 \newcommand*{\glslocalreset}[1]{%
2658   \gls@ifnotmeasuring
2659   {%
2660     \glsdoifexists{#1}%
2661     {%
2662       \@glslocalreset{#1}%
2663     }%
2664   }%
2665 }

```

`\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.

```

2666 \newcommand*{\glsunset}[1]{%
2667   \gls@ifnotmeasuring
2668   {%
2669     \glsdoifexists{#1}%
2670     {%
2671       \@glsunset{#1}%
2672     }%
2673   }%
2674 }

```

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

```

2675 \newcommand*{\glslocalunset}[1]{%
2676   \gls@ifnotmeasuring
2677   {%
2678     \glsdoifexists{#1}%
2679     {%
2680       \@glslocalunset{#1}%
2681     }%
2682   }%
2683 }

```

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

```

2684 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2685 \newcommand*{\@glslocalunset}[1]{%
2686   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2687 }

```

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

```

2688 \newcommand*{\@glsunset}{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2689 \newcommand*{\@@glsunset}[1]{%
2690   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2691 }

```

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

```
2692 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`\@@glslocalreset` Local reset without checks.

```
2693 \newcommand*{\@@glslocalreset}[1]{%
2694   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2695 }
```

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

```
2696 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2697 \newcommand*{\@@glsreset}[1]{%
2698   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2699 }
```

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

`\glsresetall`

```
2700 \newcommand*{\glsresetall}[1][\@glo@types]{%
2701   \forallglsentries[#1]{\@glsentry}%
2702   {%
2703     \glsreset{\@glsentry}%
2704   }%
2705 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2706 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2707   \forallglsentries[#1]{\@glsentry}%
2708   {%
2709     \glslocalreset{\@glsentry}%
2710   }%
2711 }
```

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

`\glsunsetall`

```
2712 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2713   \forallglsentries[#1]{\@glsentry}%
2714   {%
2715     \glsunset{\@glsentry}%
2716   }%
2717 }
```

As above, but with only a local effect:

lslocalunsetall

```

2718 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2719   \forallglsentries[#1]{\@glsentry}%
2720   {%
2721     \glslocalunset{\@glsentry}%
2722   }%
2723 }

```

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.

```

2724 \newcommand*{\@newglossaryentry@defcounters}{%
2725   \csdef{glo@\@glo@label @currcount}{0}%
2726   \csdef{glo@\@glo@label @prevcount}{0}%
2727 }

```

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

```

2728 \newcommand*{\glsenableentrycount}{%

```

Enable new entry fields.

```

2729   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters

```

Disable `\newglossaryentry` in the document environment.

```

2730   \renewcommand*{\gls@defdocnewglossaryentry}{%
2731     \renewcommand*{\newglossaryentry}[2]{%
2732       \PackageError{glossaries}{\string\newglossaryentry\space
2733         may only be used in the preamble when entry counting has
2734         been activated}{If you use \string\glsenableentrycount\space
2735         you must place all entry definitions in the preamble not in
2736         the document environment}%
2737     }%
2738   }%

```

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

```

2739   \newcommand*{\glsentrycurrcount}[1]{%
2740     \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2741     {0}{\@gls@entry@field{##1}{currcount}}%
2742   }%
2743   \newcommand*{\glsentryprevcount}[1]{%

```

```

2744 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2745 {0}{\@gls@entry@field{##1}{prevcount}}}%
2746 }%

```

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

```

2747 \renewcommand*{\@glsunset}[1]{%
2748   \@glsunset{##1}%
2749   \@gls@increment@currcount{##1}%
2750 }%
2751 \renewcommand*{\@glslocalunset}[1]{%
2752   \@glslocalunset{##1}%
2753   \@gls@local@increment@currcount{##1}%
2754 }%
2755 \renewcommand*{\@glsreset}[1]{%
2756   \@glsreset{##1}%
2757   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2758 }%
2759 \renewcommand*{\@glslocalreset}[1]{%
2760   \@glslocalreset{##1}%
2761   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2762 }%

```

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.)

```

2763 \def\@cgl's@##1##2[##3]{%
2764   \ifnum\glsentryprevcount{##2}=1\relax
2765     \cgl'sformat{##2}{##3}%
2766     \glsunset{##2}%
2767   \else
2768     \@gls@{##1}{##2}[##3]%
2769   \fi
2770 }%

```

Similarly for the analogous commands. No case change plural:

```

2771 \def\@cgl'spl@##1##2[##3]{%
2772   \ifnum\glsentryprevcount{##2}=1\relax
2773     \cgl'splformat{##2}{##3}%
2774     \glsunset{##2}%
2775   \else
2776     \@cgl'spl@{##1}{##2}[##3]%
2777   \fi
2778 }%

```

First letter uppercase singular:

```

2779 \def\@cGls@##1##2[##3]{%
2780   \ifnum\glsentryprevcount{##2}=1\relax
2781     \cGlsformat{##2}{##3}%
2782     \glsunset{##2}%
2783   \else
2784     \@Gls@{##1}{##2}[##3]%
2785   \fi

```

2786 }%

First letter uppercase plural:

```
2787 \def\@cGlspl@##1##2[##3]{%
2788   \ifnum\glsentryprevcount{##2}=1\relax
2789     \cGlsplformat{##2}{##3}%
2790     \glsunset{##2}%
2791   \else
2792     \@Glspl@{##1}{##2}[##3]%
2793   \fi
2794 }%
```

Write information to aux file at the end of the document

```
2795 \AtEndDocument{\@gls@write@entrycounts}%
```

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

```
2796 \renewcommand*{\@gls@entry@count}[2]{%
2797   \csxdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
2798 }%
```

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

```
2799 \let\glsenableentrycount\relax
2800 }
2801 \@onlypreamble\glsenableentrycount
```

ement@currcount

```
2802 \newcommand*{\@gls@increment@currcount}[1]{%
2803   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2804     \number\numexpr\glsentrycurrcount{##1}+1}%
2805 }
```

ement@currcount

```
2806 \newcommand*{\@gls@local@increment@currcount}[1]{%
2807   \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2808     \number\numexpr\glsentrycurrcount{##1}+1}%
2809 }
```

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.)

```
2810 \newcommand*{\@gls@write@entrycounts}{%
2811   \immediate\write\@auxout
2812     {\string\providecommand*\string\@gls@entry@count}[2]{}}%
2813   \forallglsentries{\@glsentry}{%
2814     \ifglsused{\@glsentry}%
2815     {\immediate\write\@auxout
2816       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}}%
2817     {}}%
```

```

2818 }%
2819 }

gls@entry@count Default behaviour is to ignore arguments. Activated by \glsenableentrycount.
2820 \newcommand*{\@gls@entry@count}[2]{%

\cgl's Define command that works like \gls but behaves differently if the entry count function is
enabled. (If not enabled, it behaves the same as \gls but issues a warning.)
2821 \newrobustcmd*{\cgl's}{\@gls@hyp@opt\@cgl's}

\@cgl's Defined the un-starred form. Need to determine if there is a final optional argument
2822 \newcommand*{\@cgl's}[2][{}]{%
2823 \new@ifnextchar[{\@cgl's@{#1}{#2}}{\@cgl's@{#1}{#2}[]}%
2824 }

\@cgl's@ Read in the final optional argument. This defaults to same behaviour as \gls but issues a
warning.
2825 \def\@cgl's@#1#2[#3]{%
2826 \GlossariesWarning{\string\cgl's\space is defaulting to
2827 \string\gls\space since you haven't enabled entry counting}%
2828 \@gls@{#1}{#2}[#3]%
2829 }

\cgl'sformat Format used by \cgl's if entry only used once on previous run. The first argument is the label,
the second argument is the insert text.
2830 \newcommand*{\cgl'sformat}[2]{%
2831 \ifgl'shaslong{#1}{\gl'sentrylong{#1}}{\gl'sentryfirst{#1}}#2%
2832 }

\cGls Define command that works like \Gls but behaves differently if the entry count function is
enabled. (If not enabled, it behaves the same as \Gls but issues a warning.)
2833 \newrobustcmd*{\cGls}{\@gls@hyp@opt\@cGls}

\@cGls Defined the un-starred form. Need to determine if there is a final optional argument
2834 \newcommand*{\@cGls}[2][{}]{%
2835 \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2}[]}%
2836 }

\@cGls@ Read in the final optional argument. This defaults to same behaviour as \Gls but issues a
warning.
2837 \def\@cGls@#1#2[#3]{%
2838 \GlossariesWarning{\string\cGls\space is defaulting to
2839 \string\Gls\space since you haven't enabled entry counting}%
2840 \@Gls@{#1}{#2}[#3]%
2841 }

```


`\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.

```
2842 \newcommand*{\cGlsformat}[2]{%
2843   \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2844 }
```

`\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.)

```
2845 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}
```

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

```
2846 \newcommand*{\@cglsp1}[2][{}]{%
2847   \new@ifnextchar[\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2}[{}]}%
2848 }
```

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

```
2849 \def\@cglsp1@#1#2[#3]{%
2850   \GlossariesWarning{\string\cglsp1\space is defaulting to
2851     \string\glsp1\space since you haven't enabled entry counting}%
2852   \@glsp1@{#1}{#2}[#3]%
2853 }
```

`\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.

```
2854 \newcommand*{\cglsp1format}[2]{%
2855   \ifglshaslong{#1}{\glsp1long{#1}}{\glsp1firstplural{#1}}#2%
2856 }
```

`\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.)

```
2857 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}
```

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

```
2858 \newcommand*{\@cGlspl}[2][{}]{%
2859   \new@ifnextchar[\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[{}]}%
2860 }
```

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

```
2861 \def\@cGlspl@#1#2[#3]{%
2862   \GlossariesWarning{\string\cGlspl\space is defaulting to
2863     \string\Glspl\space since you haven't enabled entry counting}%
2864   \@Glspl@{#1}{#2}[#3]%
2865 }
```

`\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.

```
2866 \newcommand*{\cGlsplformat}[2]{%
2867   \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2868 }
```

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`

```
2869 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2870   \let\@gls@default\glsdefaulttype
2871   \def\glsdefaulttype{#1}\input{#2}%
2872   \let\glsdefaulttype\@gls@default
2873 }
```

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

```
2874 \@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`

```
2875 \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 \LaTeX code that can be used in the preamble.

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

```
2876 \newcommand*{\glsentryfmt}{%
2877   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2878 }
```

Format that provides backwards compatibility:

```
2879 \newcommand*{\@gls@default@entryfmt}[2]{%
2880   \ifdefempty\glscustomtext
2881   {%
2882     \glsifplural
2883     {%
```

Plural form

```
2884     \glscapscase
2885     {%
```

Don't adjust case

```
2886     \ifglsused\glslabel
2887     {%
```

Subsequent use

```
2888         #2{\glsentryplural{\glslabel}}%
2889         {\glsentrydescplural{\glslabel}}%
2890         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2891     }%
2892     {%
```

First use

```
2893         #1{\glsentryfirstplural{\glslabel}}%
2894         {\glsentrydescplural{\glslabel}}%
2895         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2896     }%
2897     }%
2898     {%
```

Make first letter upper case

```
2899     \ifglsused\glslabel
2900     {%
```

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.)

```
2901     \ifbool{glscompatible-3.07}%
2902     {%
2903       \protected@edef\@glo@etext{%
2904         #2{\glsentryplural{\glslabel}}%
2905         {\glsentrydescplural{\glslabel}}%
2906         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2907       \xmakefirstuc\@glo@etext
2908     }%
```

```

2909      {%
2910      #2{\Glsentryplural{\glslabel}}%
2911      {\glsentrydescplural{\glslabel}}%
2912      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2913      }%
2914      }%
2915      {%

First use
2916      \ifbool{glscompatible-3.07}%
2917      {%
2918      \protected@edef\@glo@etext{%
2919      #1{\glsentryfirstplural{\glslabel}}%
2920      {\glsentrydescplural{\glslabel}}%
2921      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2922      \xmakefirstuc\@glo@etext
2923      }%
2924      {%
2925      #1{\Glsentryfirstplural{\glslabel}}%
2926      {\glsentrydescplural{\glslabel}}%
2927      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2928      }%
2929      }%
2930      }%
2931      {%

```

Make all upper case

```

2932      \ifglsused\glslabel
2933      {%

```

Subsequent use

```

2934      \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2935      {\glsentrydescplural{\glslabel}}%
2936      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2937      }%
2938      {%

```

First use

```

2939      \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2940      {\glsentrydescplural{\glslabel}}%
2941      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2942      }%
2943      }%
2944      }%
2945      {%

```

Singular form

```

2946      \glscapscase
2947      {%

```

Don't adjust case

```

2948      \ifglsused\glslabel

```

```

2949      {%
Subsequent use
2950      #2{\glsentrytext{\glslabel}}%
2951      {\glsentrydesc{\glslabel}}%
2952      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2953      }%
2954      {%

First use
2955      #1{\glsentryfirst{\glslabel}}%
2956      {\glsentrydesc{\glslabel}}%
2957      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2958      }%
2959      }%
2960      {%

Make first letter upper case
2961      \ifglused\glslabel
2962      {%

Subsequent use
2963      \ifbool{glscompatible-3.07}%
2964      {%
2965      \protected@edef\@glo@etext{%
2966      #2{\glsentrytext{\glslabel}}%
2967      {\glsentrydesc{\glslabel}}%
2968      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2969      \xmakefirstuc\@glo@etext
2970      }%
2971      {%
2972      #2{\Glsentrytext{\glslabel}}%
2973      {\glsentrydesc{\glslabel}}%
2974      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2975      }%
2976      }%
2977      {%

First use
2978      \ifbool{glscompatible-3.07}%
2979      {%
2980      \protected@edef\@glo@etext{%
2981      #1{\glsentryfirst{\glslabel}}%
2982      {\glsentrydesc{\glslabel}}%
2983      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2984      \xmakefirstuc\@glo@etext
2985      }%
2986      {%
2987      #1{\Glsentryfirst{\glslabel}}%
2988      {\glsentrydesc{\glslabel}}%
2989      {\glsentrysymbol{\glslabel}}{\glsinsert}%

```

```

2990      }%
2991      }%
2992      }%
2993      {%

```

Make all upper case

```

2994      \ifglsused\glslabel
2995      {%

```

Subsequent use

```

2996      \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2997      {\glsentrydesc{\glslabel}}}%
2998      {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2999      }%
3000      {%

```

First use

```

3001      \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
3002      {\glsentrydesc{\glslabel}}}%
3003      {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
3004      }%
3005      }%
3006      }%
3007      }%
3008      {%

```

Custom text provided in \glsdisp

```

3009      \ifglsused{\glslabel}%
3010      {%

```

Subsequent use

```

3011      #2{\glscustomtext}%
3012      {\glsentrydesc{\glslabel}}}%
3013      {\glsentrysymbol{\glslabel}}{}%
3014      }%
3015      {%

```

First use

```

3016      #1{\glscustomtext}%
3017      {\glsentrydesc{\glslabel}}}%
3018      {\glsentrysymbol{\glslabel}}{}%
3019      }%
3020      }%
3021      }

```

`\glsentryfmt` Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

```

3022 \newcommand*{\glsentryfmt}{%
3023   \ifdefempty\glscustomtext
3024   {%
3025     \glsifplural
3026     {%

```

Plural form

3027 \glscapscase
3028 {%

Don't adjust case

3029 \ifglused\glslabel
3030 {%

Subsequent use

3031 \glentryplural{\glslabel}\glinsert
3032 }%
3033 {%

First use

3034 \glentryfirstplural{\glslabel}\glinsert
3035 }%
3036 }%
3037 {%

Make first letter upper case

3038 \ifglused\glslabel
3039 {%

Subsequent use.

3040 \Glsentryplural{\glslabel}\glinsert
3041 }%
3042 {%

First use

3043 \Glsentryfirstplural{\glslabel}\glinsert
3044 }%
3045 }%
3046 {%

Make all upper case

3047 \ifglused\glslabel
3048 {%

Subsequent use

3049 \mfirstucMakeUppercase
3050 {\glentryplural{\glslabel}\glinsert}%
3051 }%
3052 {%

First use

3053 \mfirstucMakeUppercase
3054 {\glentryfirstplural{\glslabel}\glinsert}%
3055 }%
3056 }%
3057 }%
3058 {%

Singular form

```
3059 \glscapscase
3060 {%
```

Don't adjust case

```
3061 \ifglused\glslabel
3062 {%
```

Subsequent use

```
3063 \glentrytext{\glslabel}\glsinsert
3064 }%
3065 {%
```

First use

```
3066 \glentryfirst{\glslabel}\glsinsert
3067 }%
3068 }%
3069 {%
```

Make first letter upper case

```
3070 \ifglused\glslabel
3071 {%
```

Subsequent use

```
3072 \Glentrytext{\glslabel}\glsinsert
3073 }%
3074 {%
```

First use

```
3075 \Glentryfirst{\glslabel}\glsinsert
3076 }%
3077 }%
3078 {%
```

Make all upper case

```
3079 \ifglused\glslabel
3080 {%
```

Subsequent use

```
3081 \mfirstucMakeUppercase{\glentrytext{\glslabel}\glsinsert}%
3082 }%
3083 {%
```

First use

```
3084 \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glsinsert}%
3085 }%
3086 }%
3087 }%
3088 }%
3089 {%
```

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

```
3090 \glscustomtext\glsinsert
```



```

3091 }%
3092 }

```

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

```

3093 \newcommand*{\glsgenacfmt}{%
3094   \ifdefempty\glscustomtext
3095   {%
3096     \ifglused\glslabel
3097     {%

```

Subsequent use:

```

3098     \glsifplural
3099     {%

```

Subsequent plural form:

```

3100     \glscapscase
3101     {%

```

Subsequent plural form, don't adjust case:

```

3102     \acronymfont{\glstryshortpl{\glslabel}}\glsinsert
3103     }%
3104     {%

```

Subsequent plural form, make first letter upper case:

```

3105     \acronymfont{\Glstryshortpl{\glslabel}}\glsinsert
3106     }%
3107     {%

```

Subsequent plural form, all caps:

```

3108     \mfirstucMakeUppercase
3109     {\acronymfont{\glstryshortpl{\glslabel}}\glsinsert}%
3110     }%
3111     }%
3112     {%

```

Subsequent singular form

```

3113     \glscapscase
3114     {%

```

Subsequent singular form, don't adjust case:

```

3115     \acronymfont{\glstryshort{\glslabel}}\glsinsert
3116     }%
3117     {%

```

Subsequent singular form, make first letter upper case:

```

3118     \acronymfont{\Glstryshort{\glslabel}}\glsinsert
3119     }%
3120     {%

```

Subsequent singular form, all caps:

```

3121     \mfirstucMakeUppercase
3122     {\acronymfont{\glstryshort{\glslabel}}\glsinsert}%

```

```

3123      }%
3124      }%
3125      }%
3126      {%

```

First use:

```

3127      \glsifplural
3128      {%

```

First use plural form:

```

3129      \glscapscase
3130      {%

```

First use plural form, don't adjust case:

```

3131      \genplacrformat{\glslabel}{\glsinsert}%
3132      }%
3133      {%

```

First use plural form, make first letter upper case:

```

3134      \Genplacrformat{\glslabel}{\glsinsert}%
3135      }%
3136      {%

```

First use plural form, all caps:

```

3137      \mfirstucMakeUppercase
3138      {\genplacrformat{\glslabel}{\glsinsert}}%
3139      }%
3140      }%
3141      {%

```

First use singular form

```

3142      \glscapscase
3143      {%

```

First use singular form, don't adjust case:

```

3144      \genacrformat{\glslabel}{\glsinsert}%
3145      }%
3146      {%

```

First use singular form, make first letter upper case:

```

3147      \Genacrformat{\glslabel}{\glsinsert}%
3148      }%
3149      {%

```

First use singular form, all caps:

```

3150      \mfirstucMakeUppercase
3151      {\genacrformat{\glslabel}{\glsinsert}}%
3152      }%
3153      }%
3154      }%
3155      }%
3156      {%

```

User supplied text.

```
3157 \glscustomtext
3158 }%
3159 }
```

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

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

```
3160 \newcommand*{\genacrfullformat}[2]{%
3161   \glentrylong{#1}#2\space
3162   (\protect\firstacronymfont{\glentryshort{#1}})%
3163 }
```

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

As above but makes the first letter upper case.

```
3164 \newcommand*{\Genacrfullformat}[2]{%
3165   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3166   \xmakefirstuc\gls@text
3167 }
```

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

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

```
3168 \newcommand*{\genplacrfullformat}[2]{%
3169   \glentrylongpl{#1}#2\space
3170   (\protect\firstacronymfont{\glentryshortpl{#1}})%
3171 }
```

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

As above but makes the first letter upper case.

```
3172 \newcommand*{\Genplacrfullformat}[2]{%
3173   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3174   \xmakefirstuc\gls@text
3175 }
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
3176 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3177 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```

3178 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3179   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3180   Use \string\defglsentryfmt\space instead}%
3181   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3182   \edef\@gls@doentrydef{%
3183     \noexpand\defglsentryfmt[#1]{%
3184       \noexpand\ifcsdef{gls@#1@displayfirst}%
3185       {%
3186         \noexpand\@@gls@default@entryfmt
3187         {\noexpand\csuse{gls@#1@displayfirst}}}%
3188         {\noexpand\csuse{gls@#1@display}}}%
3189       }%
3190       {%
3191         \noexpand\@@gls@default@entryfmt
3192         {\noexpand\glsdisplayfirst}%
3193         {\noexpand\csuse{gls@#1@display}}}%
3194       }%
3195     }%
3196   }%
3197   \@gls@doentrydef
3198 }
```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```

3199 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3200   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3201   Use \string\defglsentryfmt\space instead}%
3202   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3203   \edef\@gls@doentrydef{%
3204     \noexpand\defglsentryfmt[#1]{%
3205       \noexpand\ifcsdef{gls@#1@display}%
3206       {%
3207         \noexpand\@@gls@default@entryfmt
3208         {\noexpand\csuse{gls@#1@displayfirst}}}%
3209         {\noexpand\csuse{gls@#1@display}}}%
3210       }%
3211       {%
3212         \noexpand\@@gls@default@entryfmt
3213         {\noexpand\csuse{gls@#1@displayfirst}}}%
3214         {\noexpand\glsdisplay}%
3215       }%
3216     }%
3217   }%
3218   \@gls@doentrydef
3219 }
```

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.

```
3220 \define@key{glslink}{counter}{%
3221   \ifcsundef{c@#1}%
3222   {%
3223     \PackageError{glossaries}%
3224     {There is no counter called '#1'}%
3225     {%
3226       The counter key should have the name of a valid counter
3227       as its value%
3228     }%
3229   }%
3230   {%
3231     \def\@gls@counter{#1}%
3232   }%
3233 }
```

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.

```
3234 \define@key{glslink}{format}{%
3235   \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.

```
3236 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3237 \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.

```
3238 \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.

```
3239 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3240 \newcommand*{\glsifhyper}[2]{%
3241 \glslinkvar{#1}{#2}{#1}%
3242 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3243 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3244 }
```

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

```
3245 \newcommand*{\@gls@hyp@opt}[1]{%
3246 \let\glslinkvar\@firstofthree
3247 \let\@gls@hyp@opt@cs#1\relax
3248 \@ifstar{\s@gls@hyp@opt}%
3249 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3250 }
```

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

```
3251 \newcommand*{\s@gls@hyp@opt}[1] []{%
3252 \let\glslinkvar\@secondofthree
3253 \@gls@hyp@opt@cs[hyper=false,#1]}
```

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

```
3254 \newcommand*{\p@gls@hyp@opt}[1] []{%
3255 \let\glslinkvar\@thirdofthree
3256 \@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

```
3257 \newrobustcmd*{\glslink}{%  
3258 \@gls@hyp@opt\@gls@@link  
3259 }
```

\@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.

```
3260 \newcommand*{\@gls@@link}[3][\@gls@link]{%  
3261 \glsdoifexistsordo{#2}%  
3262 {%  
3263 \let\do@gls@link@checkfirsthyper\relax  
3264 \@gls@link[#1]{#2}{#3}%  
3265 }%
```

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

```
3266 \glstextformat{#3}%  
3267 }%  
  
3268 \glspostlinkhook  
3269 }
```

glspostlinkhook

```
3270 \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.

```
3271 \newcommand*{\@gls@link@checkfirsthyper}{%  
3272 \ifglsused{\glslabel}%  
3273 {%  
3274 }%  
3275 {%  
3276 \gls@checkisacronymlist\glstype  
3277 \ifglshyperfirst  
3278 \if@glsisacronymlist  
3279 \ifglsacrfootnote  
3280 \KV@glslink@hyperfalse  
3281 \fi  
3282 \fi  
3283 \else  
3284 \KV@glslink@hyperfalse  
3285 \fi  
3286 }%
```

Allow user to hook into this

```
3287 \glslinkcheckfirsthyperhook  
3288 }
```

`checkfirsthyperhook` Allow used to hook into the `\@gls@link@checkfirsthyper` macro
3289 `\newcommand*{\glslinkcheckfirsthyperhook}{}`

`linkpostsetkeys`
3290 `\newcommand*{\glslinkpostsetkeys}{}`

`\glsifhyperon` Check the value of the hyper key:
3291 `\newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}`

`disablehyperinlist` Disable hyperlink if in the “nohyper” list.
3292 `\newcommand*{\do@glssdisablehyperinlist}{%`
3293 `\expandafter\DTLifinlist\expandafter{\gls@type}{\@gls@nohyperlist}%`
3294 `{\KV@glslink@hyperfalse}{}}%`
3295 `}`

`let@glslink@opts` Hook to set default options for `\@glslink`.
3296 `\newcommand*{\@gls@setdefault@glslink@opts}{}`

`\@gls@link`
3297 `\def\@gls@link[#1]#2#3{%`
Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with tabularx).
3298 `\leavevmode`
3299 `\edef\glslabel{\glsdetoklabel{#2}}%`
Save options in `\@gls@link@opts` and label in `\@gls@link@label`
3300 `\def\@gls@link@opts{#1}%`
3301 `\let\@gls@link@label\glslabel`
3302 `\def\@glsnumberformat{glsnumberformat}%`
3303 `\edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%`
If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default
3304 `\edef\gls@type{\csname glo@\glslabel @type\endcsname}%`
Save original setting
3305 `\let\org@ifKV@glslink@hyper\ifKV@glslink@hyper`
Set defaults:
3306 `\@gls@setdefault@glslink@opts`
Switch off hyper setting if the glossary type has been identified in nohyperlist.
3307 `\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`.
3308 `\do@gls@link@checkfirsthyper`
3309 `\setkeys{glslink}{#1}%`
Add a hook for the user to customise things after the keys have been set.
3310 `\glslinkpostsetkeys`


```

    Store the entry's counter in \theglsentrycounter
3311 \gls@saveentrycounter
    Define sort key if necessary:
3312 \gls@setsort{\glslabel}%
    (De-tok'ing done by \@do@wrglossary)
3313 \@do@wrglossary{#2}%
3314 \ifKV@glslink@hyper
3315 \glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3316 \else
3317 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3318 \fi
    Restore original setting
3319 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3320 }

\glolinkprefix
3321 \newcommand*{\glolinkprefix}{glo:}

glsentrycounter Set default value of entry counter
3322 \def\glsentrycounter{\glscounter}%

saveentrycounter Need to check if using equation counter in align environment:
3323 \newcommand*{\@gls@saveentrycounter}{%
3324 \def\@gls@Hcounter{}}%
    Are we using equation counter?
3325 \ifthenelse{\equal{\@gls@counter}{equation}}{%
3326 {
    If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as
    may be inside an inner environment.)
3327 \ifcsundef{xatlevel@}%
3328 {%
3329 \edef\theglsentrycounter{\expandafter\noexpand
3330 \csname the\@gls@counter\endcsname}%
3331 }%
3332 {%
3333 \ifx\xatlevel@\@empty
3334 \edef\theglsentrycounter{\expandafter\noexpand
3335 \csname the\@gls@counter\endcsname}%
3336 \else
3337 \savecounters@
3338 \advance\c@equation by 1\relax
3339 \edef\theglsentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3340 \ifcsundef{theH\@gls@counter}%
3341 {%
3342   \def\@gls@Hcounter{\theglsentrycounter}%
3343   }%
3344   {%
3345     \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3346     }%
3347     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3348     \restorecounters@
3349   \fi
3350 }%
3351 }%
3352 {%

```

Not using equation counter so no special measures:

```

3353 \edef\theglsentrycounter{\expandafter\noexpand
3354   \csname the\@gls@counter\endcsname}%
3355 }%

```

Check if hyperref version of this counter

```

3356 \ifx\@gls@Hcounter\@empty
3357 \ifcsundef{theH\@gls@counter}%
3358 {%
3359   \def\theHglentrycounter{\theglsentrycounter}%
3360   }%
3361   {%
3362     \protected@edef\theHglentrycounter{\expandafter\noexpand
3363       \csname theH\@gls@counter\endcsname}%
3364     }%
3365   \fi
3366 }

```

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.

```

3367 \def\@set@glo@numformat#1#2#3#4{%
3368   \expandafter\@glo@check@mkidxrangechar#3\@nil
3369   \protected@edef#1{%
3370     \@glo@prefix setentrycounter[#4]{#2}%
3371     \expandafter\string\csname\@glo@suffix\endcsname
3372   }%
3373   \@gls@checkmkidxchars#1%
3374 }

```

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.

```

3375 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3376 \if#1(\relax
3377   \def\@glo@prefix{()%
3378   \if\relax#2\relax
3379     \def\@glo@suffix{glsnumberformat}%
3380   \else
3381     \def\@glo@suffix{#2}%
3382   \fi
3383 \else
3384   \if#1)\relax
3385     \def\@glo@prefix{}}}%
3386   \if\relax#2\relax
3387     \def\@glo@suffix{glsnumberformat}%
3388   \else
3389     \def\@glo@suffix{#2}%
3390   \fi
3391 \else
3392   \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3393 \fi
3394 \fi}

```

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

```

3395 \newcommand*{\@gls@escbsdq}[1]{%
3396   \def\@gls@checkedmkidx{%
3397     \let\gls@xdystring=#1\relax
3398     \@onelevel@sanitize\gls@xdystring
3399     \edef\do@gls@xdycheckbackslash{%
3400       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3401       \@backslashchar\@backslashchar\noexpand\null}%
3402     \do@gls@xdycheckbackslash
3403     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3404     \def\@gls@checkedmkidx{%
3405       \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3406       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3407   \@for\@gls@tmp:=\gls@protected@pagefmts\do
3408   {%
3409     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3410     \@onelevel@sanitize\@gls@sanitized@tmp
3411     \edef\gls@dostsubst{%
3412       \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3413       {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3414     }%
3415     \gls@dostsubst
3416   }%

```

Assign to required control sequence

```

3417   \let#1=\gls@xdystring

```

3418 }

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

checkmkidxchars

```
3419 \newcommand{\@gls@checkmkidxchars}[1]{%
3420   \ifglxsindy
3421     \@gls@escbsdq{#1}%
3422   \else
3423     \def\@gls@checkedmkidx{%
3424       \expandafter\@gls@checkquote#1\@nil""\null
3425       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3426     \def\@gls@checkedmkidx{%
3427       \expandafter\@gls@checkescquote#1\@nil\\"\null
3428       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3429     \def\@gls@checkedmkidx{%
3430       \expandafter\@gls@checkescactual#1\@nil\??\null
3431       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3432     \def\@gls@checkedmkidx{%
3433       \expandafter\@gls@checkactual#1\@nil??\null
3434       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3435     \def\@gls@checkedmkidx{%
3436       \expandafter\@gls@checkbar#1\@nil||\null
3437       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3438     \def\@gls@checkedmkidx{%
3439       \expandafter\@gls@checkescbar#1\@nil\\|\null
3440       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3441     \def\@gls@checkedmkidx{%
3442       \expandafter\@gls@checklevel#1\@nil!!\null
3443       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3444     \fi
3445 }
```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```
3446 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
3447 \newtoks\@gls@tmpb
```

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

```
3448 \def\@gls@checkquote#1"#2"#3\null{%
3449   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3450   \toks@={#1}%
3451   \ifx\null#2\null
3452   \ifx\null#3\null
3453     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3454   \def\@gls@checkquote{\relax}%
3455   \else
```

```

3456 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3457 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3458 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
3459 \fi
3460 \else
3461 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3462 \@gls@quotechar\@gls@quotechar}%
3463 \ifx\null#3\null
3464 \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
3465 \else
3466 \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
3467 \fi
3468 \fi
3469 \@@gls@checkquote
3470 }

```

s@checkescquote Do the same for \":

```

3471 \def\@gls@checkescquote#1\"#2\"#3\null{%
3472 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3473 \toks@={#1}%
3474 \ifx\null#2\null
3475 \ifx\null#3\null
3476 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3477 \def\@@gls@checkescquote{\relax}%
3478 \else
3479 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3480 \@gls@quotechar\string\"@gls@quotechar
3481 \@gls@quotechar\string\"@gls@quotechar}%
3482 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3483 \fi
3484 \else
3485 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3486 \@gls@quotechar\string\"@gls@quotechar}%
3487 \ifx\null#3\null
3488 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3489 \else
3490 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3491 \fi
3492 \fi
3493 \@@gls@checkescquote
3494 }

```

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

```

3495 \def\@gls@checkescactual#1\?#2\?#3\null{%
3496 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3497 \toks@={#1}%
3498 \ifx\null#2\null
3499 \ifx\null#3\null
3500 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```

```

3501 \def\@gls@checkescactual{\relax}%
3502 \else
3503 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3504 \@gls@quotechar\string"\@gls@actualchar
3505 \@gls@quotechar\string"\@gls@actualchar}%
3506 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3507 \fi
3508 \else
3509 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3510 \@gls@quotechar\string"\@gls@actualchar}%
3511 \ifx\null#3\null
3512 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3513 \else
3514 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3515 \fi
3516 \fi
3517 \@gls@checkescactual
3518 }

```

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

```

3519 \def\@gls@checkescbar#1\|#2\|#3\null{%
3520 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3521 \toks@={#1}%
3522 \ifx\null#2\null
3523 \ifx\null#3\null
3524 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3525 \def\@gls@checkescbar{\relax}%
3526 \else
3527 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3528 \@gls@quotechar\string"\@gls@encapchar
3529 \@gls@quotechar\string"\@gls@encapchar}%
3530 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3531 \fi
3532 \else
3533 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3534 \@gls@quotechar\string"\@gls@encapchar}%
3535 \ifx\null#3\null
3536 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3537 \else
3538 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3539 \fi
3540 \fi
3541 \@gls@checkescbar
3542 }

```

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

```

3543 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3544 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3545 \toks@={#1}%

```

```

3546 \ifx\null#2\null
3547 \ifx\null#3\null
3548 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3549 \def\@gls@checkesclevel{\relax}%
3550 \else
3551 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3552 \@gls@quotechar\string"\@gls@levelchar
3553 \@gls@quotechar\string"\@gls@levelchar}%
3554 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3555 \fi
3556 \else
3557 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3558 \@gls@quotechar\string"\@gls@levelchar}%
3559 \ifx\null#3\null
3560 \def\@gls@checkesclevel{\@gls@checkesclevel#2!!\null}%
3561 \else
3562 \def\@gls@checkesclevel{\@gls@checkesclevel#2!#3\null}%
3563 \fi
3564 \fi
3565 \@gls@checkesclevel
3566 }

```

\@gls@checkbar and for |:

```

3567 \def\@gls@checkbar#1|#2|#3\null{%
3568 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3569 \toks@={#1}%
3570 \ifx\null#2\null
3571 \ifx\null#3\null
3572 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3573 \def\@gls@checkbar{\relax}%
3574 \else
3575 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3576 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3577 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3578 \fi
3579 \else
3580 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3581 \@gls@quotechar\@gls@encapchar}%
3582 \ifx\null#3\null
3583 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3584 \else
3585 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3586 \fi
3587 \fi
3588 \@gls@checkbar
3589 }

```

@gls@checklevel and for !:

```

3590 \def\@gls@checklevel#1!#2!#3\null{%

```

```

3591 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3592 \toks@={#1}%
3593 \ifx\null#2\null
3594   \ifx\null#3\null
3595     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3596     \def\@gls@checklevel{\relax}%
3597   \else
3598     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3599     \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3600     \def\@gls@checklevel{\@gls@checklevel#3\null}%
3601   \fi
3602 \else
3603   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3604   \@gls@quotechar\@gls@levelchar}%
3605   \ifx\null#3\null
3606     \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3607   \else
3608     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3609   \fi
3610 \fi
3611 \@gls@checklevel
3612 }

```

gls@checkactual and for ?:

```

3613 \def\@gls@checkactual#1?#2?#3\null{%
3614   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3615   \toks@={#1}%
3616   \ifx\null#2\null
3617     \ifx\null#3\null
3618       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3619       \def\@gls@checkactual{\relax}%
3620     \else
3621       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3622       \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3623       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3624     \fi
3625   \else
3626     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3627     \@gls@quotechar\@gls@actualchar}%
3628     \ifx\null#3\null
3629       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3630     \else
3631       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3632     \fi
3633   \fi
3634   \@gls@checkactual
3635 }

```

s@xdycheckquote As before but for use with xindy


```

3636 \def\@gls@xdycheckquote#1"#2"#3\null{%
3637   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3638   \toks@={#1}%
3639   \ifx\null#2\null
3640     \ifx\null#3\null
3641       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3642       \def\@gls@xdycheckquote{\relax}%
3643     \else
3644       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3645         \string"\string"}%
3646       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3647     \fi
3648   \else
3649     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3650       \string"}%
3651     \ifx\null#3\null
3652       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3653     \else
3654       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3655     \fi
3656   \fi
3657   \@gls@xdycheckquote
3658 }

```

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

```

3659 \edef\def\@gls@xdycheckbackslash{%
3660   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3661     ##2\@backslashchar##3\noexpand\null{%
3662     \noexpand\@gls@tmpb=\noexpand\expandafter
3663       {\noexpand\@gls@checkedmkidx}%
3664     \noexpand\toks@={##1}%
3665     \noexpand\ifx\noexpand\null##2\noexpand\null
3666     \noexpand\ifx\noexpand\null##3\noexpand\null
3667       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3668         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3669       \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3670     \noexpand\else
3671       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3672         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3673         \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3674     \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3675       \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3676     \noexpand\fi
3677   \noexpand\else
3678     \noexpand\edef\noexpand\@gls@checkedmkidx{%
3679       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3680       \@backslashchar\@backslashchar}%
3681   \noexpand\ifx\noexpand\null##3\noexpand\null
3682     \noexpand\def\noexpand\@gls@xdycheckbackslash{%

```

```

3683 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3684 \@backslashchar\noexpand\null}%
3685 \noexpand\else
3686 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3687 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3688 ##3\noexpand\null}%
3689 \noexpand\fi
3690 \noexpand\fi
3691 \noexpand\@gls@xdycheckbackslash
3692 }%
3693 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3694 \def@gls@xdycheckbackslash

```

\glsdohypertarget

```

3695 \newlength@gls@tmplen
3696 \newcommand*{\glsdohypertarget}[2]{%
3697 \@glsshowtarget{#1}%
3698 \settoheight@gls@tmplen{#2}%
3699 \raisebox@gls@tmplen{\hypertarget{#1}{}}#2%
3700 }

```

\glsdohyperlink

```

3701 \newcommand*{\glsdohyperlink}[2]{%
3702 \@glsshowtarget{#1}%
3703 \hyperlink{#1}{#2}%
3704 }

```

\glsdonohyperlink

```

3705 \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.

```

3706 \ifcsundef{hyperlink}%
3707 {%
3708 \let@glslink@glsdonohyperlink
3709 }%
3710 {%
3711 \let@glslink@glsdohyperlink
3712 }

```

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

```

3713 \ifcsundef{hypertarget}%
3714 {%
3715 \let@glstarget@secondoftwo
3716 }%

```

```

3717 {%
3718   \let\@glstarget\glstdohypertarget
3719 }

```

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

`\glsdisablehyper`

```

3720 \newcommand{\glsdisablehyper}{%
3721   \KV@glslink@hyperfalse
3722   \let\@glslink\glstdonohyperlink
3723   \let\@glstarget\@secondoftwo
3724 }

```

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

`\glsenablehyper`

```

3725 \newcommand{\glsenablehyper}{%
3726   \KV@glslink@hypertrue
3727   \let\@glslink\glstdohyperlink
3728   \let\@glstarget\glstdohypertarget
3729 }

```

Provide some convenience commands if not already defined:

```

3730 \providecommand{\@firstofthree}[3]{#1}
3731 \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`

```
3732 \newrobustcmd*\gls{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```

3733 \newcommand*\@gls[2][ ]{%
3734   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}]%
3735 }

```

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

```
3736 \def\@gls@#1#2[#3]{%
3737   \glsdoifexists{#2}%
3738   {%
3739     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3740     \let\glsifplural\@secondoftwo
3741     \let\glscapscase\@firstofthree
3742     \let\glscustomtext\@empty
3743     \def\glsinsert{#3}%
```

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

```
3744   \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.

```
3745   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3746   \ifKV@glslink@local
3747     \glslocalunset{#2}%
3748   \else
3749     \glsunset{#2}%
3750   \fi
3751 }%
```

```
3752 \glspostlinkhook
3753 }
```

`\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`

```
3754 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}
```

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

```
3755 \newcommand*{\@Gls}[2][]{%
3756   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[]}%
3757 }
```

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

```
3758 \def\@Gls@#1#2[#3]{%
3759   \glsdoifexists{#2}%
3760   {%
3761     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3762 \let\glsifplural\@secondoftwo
3763 \let\glsifcaps\@secondofthree
3764 \let\glsifcustomtext\@empty
3765 \def\glsinsert{#3}%

```

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

```

3766 \def\@glo@text{\csname gls@\gls@type @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.

```

3767 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3768 \ifKV@glslink@local
3769 \glslocalunset{#2}%
3770 \else
3771 \glsunset{#2}%
3772 \fi
3773 }%

```

```

3774 \gls@postlinkhook
3775 }

```

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

\GLS

```

3776 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

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

```

3777 \newcommand*{\@GLS}[2][]{%
3778 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}%
3779 }

```

\@GLS@ Read in the final optional argument:

```

3780 \def\@GLS@#1#2[#3]{%
3781 \glsdoifexists{#2}%
3782 {%
3783 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3784 \let\glsifplural\@secondoftwo
3785 \let\glsifcaps\@thirdofthree
3786 \let\glsifcustomtext\@empty
3787 \def\glsinsert{#3}%

```

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

```

3788 \def\@glo@text{\csname gls@\gls@type @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.

```

3789 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```
3790 \ifKV@glslink@local
3791 \glsllocalunset{#2}%
3792 \else
3793 \glsunset{#2}%
3794 \fi
3795 }%

3796 \glspostlinkhook
3797 }
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```
3798 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}
```

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

```
3799 \newcommand*{\@glspl}[2][\@gls@hyp@opt\@glspl]{%
3800 \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}[]}%
3801 }
```

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

```
3802 \def\@glspl@#1#2[#3]{%
3803 \glsoifexists{#2}%
3804 {%
3805 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3806 \let\glsifplural\@firstoftwo
3807 \let\glsupcase\@firstofthree
3808 \let\glscustomtext\@empty
3809 \def\glsinsert{#3}%
```

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

```
3810 \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.

```
3811 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3812 \ifKV@glslink@local
3813 \glsllocalunset{#2}%
3814 \else
3815 \glsunset{#2}%
3816 \fi
3817 }%

3818 \glspostlinkhook
3819 }
```

`\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`

```
3820 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}
```

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

```
3821 \newcommand*{\@Glspl}[2] [] {%
3822   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2} []}]%
3823 }
```

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

```
3824 \def\@Glspl@#1#2[#3] {%
3825   \glsdoifexists{#2}%
3826   {%
3827     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3828     \let\glsifplural\@firstoftwo
3829     \let\glsapscase\@secondofthree
3830     \let\glscustomtext\@empty
3831     \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`.

```
3832   \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.

```
3833   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3834   \ifKV@glslink@local
3835     \glslocalunset{#2}%
3836   \else
3837     \glsunset{#2}%
3838   \fi
3839 }%
```

```
3840 \glspostlinkhook
3841 }
```

`\GLSp1` behaves like `\glsp1` except that all the link text is converted to uppercase.

`\GLSp1`

```
3842 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3843 \newcommand*{\@GLSp1}[2] [] {%
3844   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}]%
3845 }
```

`\@GLSp1` Read in the final optional argument:

```

3846 \def\@GLSp1@#1#2[#3]{%
3847   \glsdoifexists{#2}%
3848   {%
3849     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3850     \let\glsifplural\@firstoftwo
3851     \let\glscapscase\@thirdofthree
3852     \let\glscustomtext\@empty
3853     \def\glsinsert{#3}%

```

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

```

3854   \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.

```

3855   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3856   \ifKV@glslink@local
3857     \glslocalunset{#2}%
3858   \else
3859     \glsunset{#2}%
3860   \fi
3861 }%

```

```

3862 \glspostlinkhook
3863 }

```

`\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:

```

3864 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3865 \newcommand*{\@glsdisp}[3][ ]{%
3866   \glsdoifexists{#2}{%

3867     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3868     \let\glsifplural\@secondoftwo
3869     \let\glscapscase\@firstofthree
3870     \def\glscustomtext{#3}%
3871     \def\glsinsert{}%

```


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

```
3872 \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.

```
3873 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3874 \ifKV@glslink@local
3875 \glslocalunset{#2}%
3876 \else
3877 \glsunset{#2}%
3878 \fi
3879 }%
```

```
3880 \glspostlinkhook
3881 }
```

`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.

```
3882 \newcommand*{\@gls@link@nocheckfirsthyper}{}
```

`@gls@field@link`

```
3883 \newcommand{\@gls@field@link}[3]{%
3884 \glsdoifexists{#2}%
3885 {%
3886 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
3887 \@gls@link[#1]{#2}{#3}%
3888 }%
3889 \glspostlinkhook
3890 }
```

`\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`

```
3891 \newrobustcmd*{\glstext}{\@gls@hyp@opt\@glstext}
```

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

```
3892 \newcommand*{\@glstext}[2][{}]{%
3893 \new@ifnextchar[\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[{}]}}
```

Read in the final optional argument:

```
3894 \def\@glstext@#1#2[#3]{%
3895 \@gls@field@link{#1}{#2}{\glstext{#2}#3}%
3896 }
```

`\GLStext` behaves like `\glstext` except the text is converted to uppercase.

`\GLStext`

```
3897 \newrobustcmd*{\GLStext}{\@gls@hyp@opt\@GLStext}
```

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

```
3898 \newcommand*{\@GLStext}[2] [] {%
```

```
3899   \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3900 \def\@GLStext@#1#2[#3] {%
```

```
3901   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstrytext{#2}#3}}%
```

```
3902 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
3903 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@Glstext}
```

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

```
3904 \newcommand*{\@Glstext}[2] [] {%
```

```
3905   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3906 \def\@Glstext@#1#2[#3] {%
```

```
3907   \@gls@field@link{#1}{#2}{\glstrytext{#2}#3}}%
```

```
3908 }
```

`\glsfirst` behaves like `\gls` except it always uses the value given by the first key and it doesn't mark the entry as used.

`\glsfirst`

```
3909 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3910 \newcommand*{\@glsfirst}[2] [] {%
```

```
3911   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3912 \def\@glsfirst@#1#2[#3] {%
```

```
3913   \@gls@field@link{#1}{#2}{\glstryfirst{#2}#3}}%
```

```
3914 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
3915 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3916 \newcommand*{\@Glsfirst}[2] [] {%
```

```
3917   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3918 \def\@Glsfirst@#1#2[#3]{%
3919   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3920 }
```

\GLSfirst behaves like \Glsfirst except it displays the text in uppercase.

\GLSfirst

```
3921 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3922 \newcommand*{\@GLSfirst}[2] [] {%
3923   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3924 \def\@GLSfirst@#1#2[#3]{%
3925   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryfirst{#2}#3}}%
3926 }
```

\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

```
3927 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3928 \newcommand*{\@glsplural}[2] [] {%
3929   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3930 \def\@glsplural@#1#2[#3]{%
3931   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3932 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
3933 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3934 \newcommand*{\@Glsplural}[2] [] {%
3935   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3936 \def\@Glsplural@#1#2[#3]{%
3937   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
3938 }
```

\GLSplural behaves like \glsplural except that the text is converted to uppercase.

\GLSplural

```
3939 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
3940 \newcommand*{\@GLSplural}[2] [] {%  
3941   \new@ifnextchar [{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3942 \def\@GLSplural@#1#2[#3] {%  
3943   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}{#3}}}%  
3944 }
```

\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

```
3945 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3946 \newcommand*{\@glsfirstplural}[2] [] {%  
3947   \new@ifnextchar [{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3948 \def\@glsfirstplural@#1#2[#3] {%  
3949   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}{#3}}%  
3950 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
3951 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3952 \newcommand*{\@Glsfirstplural}[2] [] {%  
3953   \new@ifnextchar [{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3954 \def\@Glsfirstplural@#1#2[#3] {%  
3955   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}{#3}}%  
3956 }
```

\GLSfirstplural behaves like \glsfirstplural except that the link text is converted to uppercase.

\GLSfirstplural

```
3957 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3958 \newcommand*{\@GLSfirstplural}[2] [] {%  
3959   \new@ifnextchar [{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3960 \def\@GLSfirstplural@#1#2[#3] {%  
3961   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}{#3}}}%  
3962 }
```

`\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`

```
3963 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3964 \newcommand*{\@glsname}[2][\@glsname@{#1}{#2}]{\@glsname@{#1}{#2}[]}}
```

```
3965 \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3966 \def\@glsname@#1#2[#3]{%
```

```
3967 \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
```

```
3968 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
3969 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3970 \newcommand*{\@Glsname}[2][\@Glsname@{#1}{#2}]{\@Glsname@{#1}{#2}[]}}
```

```
3971 \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3972 \def\@Glsname@#1#2[#3]{%
```

```
3973 \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
```

```
3974 }
```

`\GLSname` behaves like `\glsname` except that the link text is converted to uppercase.

`\GLSname`

```
3975 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3976 \newcommand*{\@GLSname}[2][\@GLSname@{#1}{#2}]{\@GLSname@{#1}{#2}[]}}
```

```
3977 \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3978 \def\@GLSname@#1#2[#3]{%
```

```
3979 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
```

```
3980 }
```

`\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`

```
3981 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3982 \newcommand*{\@glsdesc}[2][\@glsdesc@{#1}{#2}]{\@glsdesc@{#1}{#2}[]}}
```

```
3983 \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3984 \def\@glsdesc@#1#2[#3]{%
3985   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3986 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3987 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3988 \newcommand*{\@Glsdesc}[2][\%]
3989   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3990 \def\@Glsdesc@#1#2[#3]{%
3991   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3992 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3993 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3994 \newcommand*{\@GLSdesc}[2][\%]
3995   \new@ifnextchar[{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3996 \def\@GLSdesc@#1#2[#3]{%
3997   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3998 }
```

\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

```
3999 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
4000 \newcommand*{\@glsdescplural}[2][\%]
4001   \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4002 \def\@glsdescplural@#1#2[#3]{%
4003   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
4004 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
4005 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
4006 \newcommand*{\@Glsdescplural}[2] [] {%
4007   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4008 \def\@Glsdescplural@#1#2[#3] {%
4009   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
4010 }
```

`\Glsdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\Glsdescplural`

```
4011 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
4012 \newcommand*{\@Glsdescplural}[2] [] {%
4013   \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4014 \def\@Glsdescplural@#1#2[#3] {%
4015   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
4016 }
```

`\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`

```
4017 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

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

```
4018 \newcommand*{\@glsymbol}[2] [] {%
4019   \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4020 \def\@glsymbol@#1#2[#3] {%
4021   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
4022 }
```

`\Glsymbol` behaves like `\glsymbol` except that the first letter is converted to uppercase.

`\Glsymbol`

```
4023 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

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

```
4024 \newcommand*{\@Glsymbol}[2] [] {%
4025   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4026 \def\@Glsymbol@#1#2[#3] {%
4027   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
4028 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
4029 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
4030 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
4031   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4032 \def\@GLSsymbol@#1#2[#3] {%
```

```
4033   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}}%
```

```
4034 }
```

`\glssymbolplural` behaves like `\gls` except it always uses the value given by the symbol-plural key and it doesn't mark the entry as used.

`glssymbolplural`

```
4035 \newrobustcmd*{\glssymbolplural}{\@gls@hyp@opt\@glssymbolplural}
```

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

```
4036 \newcommand*{\@glssymbolplural}[2] [] {%
```

```
4037   \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4038 \def\@glssymbolplural@#1#2[#3] {%
```

```
4039   \@gls@field@link{#1}{#2}{\glstentrysymbolplural{#2}#3}}%
```

```
4040 }
```

`\Glsymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`Glsymbolplural`

```
4041 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

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

```
4042 \newcommand*{\@Glsymbolplural}[2] [] {%
```

```
4043   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4044 \def\@Glsymbolplural@#1#2[#3] {%
```

```
4045   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}}%
```

```
4046 }
```

`\GLSsymbolplural` behaves like `\glssymbolplural` except that the link text is converted to uppercase.

`GLSsymbolplural`

```
4047 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
4048 \newcommand*{\@GLSsymbolplural}[2] [] {%
```

```
4049   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```


Read in the final optional argument:

```
4050 \def\@GLSsymbolplural@#1#2[#3]{%
4051   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
4052 }
```

`\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`

```
4053 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
4054 \newcommand*{\@glsuseri}[2] [] {%
4055   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4056 \def\@glsuseri@#1#2[#3]{%
4057   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
4058 }
```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```
4059 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
4060 \newcommand*{\@Glsuseri}[2] [] {%
4061   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4062 \def\@Glsuseri@#1#2[#3]{%
4063   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
4064 }
```

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
4065 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
4066 \newcommand*{\@GLSuseri}[2] [] {%
4067   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4068 \def\@GLSuseri@#1#2[#3]{%
4069   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
4070 }
```

`\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`

```
4071 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
4072 \newcommand*{\@glsuserii}[2] [] {%  
4073   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4074 \def\@glsuserii@#1#2[#3] {%  
4075   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%  
4076 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
4077 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
4078 \newcommand*{\@GLSuserii}[2] [] {%  
4079   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4080 \def\@GLSuserii@#1#2[#3] {%  
4081   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%  
4082 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
4083 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
4084 \newcommand*{\@GLSuserii}[2] [] {%  
4085   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4086 \def\@GLSuserii@#1#2[#3] {%  
4087   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%  
4088 }
```

\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

```
4089 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
4090 \newcommand*{\@glsuseriii}[2] [] {%  
4091   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4092 \def\@glsuseriii@#1#2[#3] {%  
4093   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%  
4094 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
4095 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
4096 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
4097   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
4098 \def\@Glsuseriii@#1#2[#3] {%
```

```
4099   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
4100 }
```

\Glsuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
4101 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
4102 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
4103   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]
```

Read in the final optional argument:

```
4104 \def\@GLSuseriii@#1#2[#3] {%
```

```
4105   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
```

```
4106 }
```

\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

```
4107 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
4108 \newcommand*{\@glsuseriv}[2] [] {%
```

```
4109   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}]
```

Read in the final optional argument:

```
4110 \def\@glsuseriv@#1#2[#3] {%
```

```
4111   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
4112 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\GLSuseriv

```
4113 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
4114 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
4115   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}]
```

Read in the final optional argument:

```
4116 \def\@GLSuseriv@#1#2[#3] {%
```

```
4117   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
```

```
4118 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
4119 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
4120 \newcommand*{\@GLSuseriv}[2] [] {%
```

```
4121   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4122 \def\@GLSuseriv@#1#2[#3] {%
```

```
4123   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
```

```
4124 }
```

\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

```
4125 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
4126 \newcommand*{\@glsuserv}[2] [] {%
```

```
4127   \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4128 \def\@glsuserv@#1#2[#3] {%
```

```
4129   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}}%
```

```
4130 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
4131 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
4132 \newcommand*{\@Glsuserv}[2] [] {%
```

```
4133   \new@ifnextchar[{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4134 \def\@Glsuserv@#1#2[#3] {%
```

```
4135   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}}%
```

```
4136 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4137 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
4138 \newcommand*{\@GLSuserv}[2] [] {%
```

```
4139   \new@ifnextchar[{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4140 \def\@GLSuserv@#1#2[#3]{%
4141   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4142 }
```

\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

```
4143 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4144 \newcommand*{\@glsuservi}[2][ ]{%
4145   \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
4146 \def\@glsuservi@#1#2[#3]{%
4147   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
4148 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4149 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4150 \newcommand*{\@Glsuservi}[2][ ]{%
4151   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
4152 \def\@Glsuservi@#1#2[#3]{%
4153   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
4154 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4155 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
4156 \newcommand*{\@GLSuservi}[2][ ]{%
4157   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
4158 \def\@GLSuservi@#1#2[#3]{%
4159   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
4160 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4161 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
4162 \newcommand*{\ns@acrshort}[2] [] {%
4163   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} [] }%
4164 }
```

Read in the final optional argument:

```
4165 \def\@acrshort#1#2[#3] {%
4166   \glsdoifexists{#2}%
4167   {%
4168     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4169     \let\glsifplural\@secondoftwo
4170     \let\glscapscase\@firstofthree
4171     \let\glsinsert\@empty
4172     \def\glscustomtext{%
4173       \acronymfont{\glentryshort{#2}}#3%
4174     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4175   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4176   }%
4177   \glspostlinkhook
4178 }
```

\Acrshort

```
4179 \newrobustcmd*{\Acrshort}{\@gl@hyp@opt\ns@Acrshort}
```

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

```
4180 \newcommand*{\ns@Acrshort}[2] [] {%
4181   \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} [] }%
4182 }
```

Read in the final optional argument:

```
4183 \def\@Acrshort#1#2[#3] {%
4184   \glsdoifexists{#2}%
4185   {%
4186     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4187     \def\glslabel{#2}%
4188     \let\glsifplural\@secondoftwo
4189     \let\glscapscase\@secondofthree
4190     \let\glsinsert\@empty
4191     \def\glscustomtext{%
4192       \acronymfont{\Glsentryshort{#2}}#3%
4193     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4194   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4195   }%
```

```

4196 \glspostlinkhook
4197 }

```

\ACRshort

```

4198 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\@ns@ACRshort}

```

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

```

4199 \newcommand*{\ns@ACRshort}[2][\%
4200 \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
4201 }

```

Read in the final optional argument:

```

4202 \def\@ACRshort#1#2[#3]{%
4203 \glsdoifexists{#2}%
4204 {%
4205 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4206 \def\glslabel{#2}%
4207 \let\glsifplural\@secondoftwo
4208 \let\glsapscase\@thirdofthree
4209 \let\glsinsert\@empty
4210 \def\glscustomtext{%
4211 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4212 }%

```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```

4213 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4214 }%
4215 \glspostlinkhook
4216 }

```

Short plural:

\acrshortpl

```

4217 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\@ns@acrshortpl}

```

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

```

4218 \newcommand*{\ns@acrshortpl}[2][\%
4219 \new@ifnextchar[\@acrshortpl{#1}{#2}]{\@acrshortpl{#1}{#2}[]}%
4220 }

```

Read in the final optional argument:

```

4221 \def\@acrshortpl#1#2[#3]{%
4222 \glsdoifexists{#2}%
4223 {%
4224 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4225 \def\glslabel{#2}%
4226 \let\glsifplural\@firstoftwo
4227 \let\glsupcase\@firstofthree
4228 \let\glsinsert\@empty
4229 \def\glscustomtext{%
4230 \acronymfont{\glsentryshortpl{#2}}#3%
4231 }%

Call \@gls@link Note that \@gls@link sets \glstyp.
4232 \@gls@link[#1]{#2}{\csname gls@\glstyp @entryfmt\endcsname}%
4233 }%

4234 \glspostlinkhook
4235 }

```

\Acrshortpl

```

4236 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4237 \newcommand*{\ns@Acrshortpl}[2] [] {%
4238 \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
4239 }

Read in the final optional argument:
4240 \def\@Acrshortpl#1#2[#3] {%
4241 \glsdoifexists{#2}%
4242 {%
4243 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4244 \def\glslabel{#2}%
4245 \let\glsifplural\@firstoftwo
4246 \let\glsupcase\@secondofthree
4247 \let\glsinsert\@empty
4248 \def\glscustomtext{%
4249 \acronymfont{\Glsentryshortpl{#2}}#3%
4250 }%

Call \@gls@link Note that \@gls@link sets \glstyp.
4251 \@gls@link[#1]{#2}{\csname gls@\glstyp @entryfmt\endcsname}%
4252 }%

4253 \glspostlinkhook
4254 }

```

\ACRshortpl

```

4255 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
4256 \newcommand*{\ns@ACRshortpl}[2] [] {%
4257 \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
4258 }

```


Read in the final optional argument:

```
4259 \def\@ACRshortpl#1#2[#3]{%
4260   \glsdoifexists{#2}%
4261   {%
4262     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4263     \def\glslabel{#2}%
4264     \let\glsifplural\@firstoftwo
4265     \let\glscapscase\@thirdofthree
4266     \let\glsinsert\@empty
4267     \def\glscustomtext{%
4268       \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
4269     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4270   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4271   }%
4272   \glspostlinkhook
4273 }
```

\acrlong

```
4274 \newrobustcmd*{\acrlong}{\@gl@hyp@opt\@ns@acrlong}
```

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

```
4275 \newcommand*{\@ns@acrlong}[2][ ]{%
4276   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[ ]}%
4277 }
```

Read in the final optional argument:

```
4278 \def\@acrlong#1#2[#3]{%
4279   \glsdoifexists{#2}%
4280   {%
4281     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4282     \def\glslabel{#2}%
4283     \let\glsifplural\@secondoftwo
4284     \let\glscapscase\@firstofthree
4285     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4286   \def\glscustomtext{%
4287     \glsentrylong{#2}#3%
4288   }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4289   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4290   }%
```

```

4291 \glspostlinkhook
4292 }

```

\Acrlong

```

4293 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}

```

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

```

4294 \newcommand*{\ns@Acrlong}[2][\]{%
4295   \new@ifnextchar[\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[\]}%
4296 }

```

Read in the final optional argument:

```

4297 \def\@Acrlong#1#2[#3]{%
4298   \glsdoifexists{#2}%
4299   {%

```

```

4300     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4301     \def\glslabel{#2}%
4302     \let\glsifplural\@secondoftwo
4303     \let\glsapscase\@secondofthree
4304     \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4305     \def\glscustomtext{%
4306       \Glsentrylong{#2}#3%
4307     }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4308     \@gls@link{#1}{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4309   }%

```

```

4310 \glspostlinkhook
4311 }

```

\ACRlong

```

4312 \newrobustcmd*{\ACRlong}{\@gls@hyp@opt\@ns@ACRlong}

```

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

```

4313 \newcommand*{\ns@ACRlong}[2][\]{%
4314   \new@ifnextchar[\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[\]}%
4315 }

```

Read in the final optional argument:

```

4316 \def\@ACRlong#1#2[#3]{%
4317   \glsdoifexists{#2}%
4318   {%

```

```

4319     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4320 \def\glslabel{#2}%
4321 \let\glsifplural\@secondoftwo
4322 \let\glsupcase\@thirdofthree
4323 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4324 \def\glscustomtext{%
4325     \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4326 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4327 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4328 }%

```

```

4329 \glspostlinkhook
4330 }

```

Short plural:

\acrlongpl

```

4331 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}

```

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

```

4332 \newcommand*{\ns@acrlongpl}[2][\%
4333 \new@ifnextchar[\@acrlongpl{#1}{#2}]{\@acrlongpl{#1}{#2}[]}%
4334 }

```

Read in the final optional argument:

```

4335 \def\@acrlongpl#1#2[#3]{%
4336 \glsdoifexists{#2}%
4337 {%
4338     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4339     \def\glslabel{#2}%
4340     \let\glsifplural\@firstoftwo
4341     \let\glsupcase\@firstofthree
4342     \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4343 \def\glscustomtext{%
4344     \glsentrylongpl{#2}#3%
4345 }%

```

Call \@gls@link. Note that \@gls@link sets \glstype.

```

4346 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4347 }%

```

```

4348 \glspostlinkhook
4349 }

```

\Acrlongpl

```
4350 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```

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

```
4351 \newcommand*{\ns@Acrlongpl}[2][\%  
4352 \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%  
4353 }
```

Read in the final optional argument:

```
4354 \def\@Acrlongpl#1#2[#3]{%  
4355 \glsdoifexists{#2}%  
4356 {%  
  
4357 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4358 \def\glslabel{#2}%  
4359 \let\glsifplural\@firstoftwo  
4360 \let\glsapscase\@secondofthree  
4361 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4362 \def\glscustomtext{%  
4363 \Glsentrylongpl{#2}#3%  
4364 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4365 \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%  
4366 }%  
  
4367 \glspostlinkhook  
4368 }
```

\ACRlongpl

```
4369 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\ns@ACRlongpl}
```

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

```
4370 \newcommand*{\ns@ACRlongpl}[2][\%  
4371 \new@ifnextchar[\@ACRlongpl{#1}{#2}]{\@ACRlongpl{#1}{#2}[]}%  
4372 }
```

Read in the final optional argument:

```
4373 \def\@ACRlongpl#1#2[#3]{%  
4374 \glsdoifexists{#2}%  
4375 {%  
  
4376 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4377 \def\glslabel{#2}%  
4378 \let\glsifplural\@firstoftwo  
4379 \let\glsapscase\@thirdofthree  
4380 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4381 \def\glscustomtext{%
4382 \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
4383 }%

Call \@gls@link. Note that \@gls@link sets \glstype.
4384 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4385 }%

4386 \glspostlinkhook
4387 }

```

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>}
```

```

4388 \newcommand*{\@gls@entry@field}[2]{%
4389 \csname glo@\glsdetoklabel{#1}@#2\endcsname
4390 }

```

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

```

4391 \newcommand*{\glsletentryfield}[3]{%
4392 \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
4393 }

```

`\Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{\<label>}{\<field>}
```

```

4394 \newcommand*{\@Gls@entry@field}[2]{%
4395 \glsdoifexistsordo{#1}%
4396 {%
4397 \letcs{\@glo@text}{glo@\glsdetoklabel{#1}@#2}%
4398 \ifdef\@glo@text
4399 {%
4400 \xmakefirstuc{\@glo@text}%
4401 }%
4402 {%
4403 ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4404 entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry

```

```

4405     label and the field name}%
4406   }%
4407 }%
4408 {%
4409   ??%
4410 }%
4411 }

```

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`

```

4412 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4413 \newrobustcmd*{\Glsentryname}[1]{%
4414   \@Gls@entryname{#1}%
4415 }

```

`\@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:

```

4416 \newcommand*{\@Gls@entryname}[1]{%
4417   \@Gls@entry@field{#1}{name}%
4418 }

```

`\ls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

4419 \newcommand*{\@Gls@acrentryname}[1]{%
4420   \ifglshaslong{#1}%
4421   {%
4422     \letcs\@glo@text{glo\@glsdetoklabel{#1}@name}%
4423     \expandafter\@gls@getbody\@glo@text{}\@nil
4424     \expandafter\ifx\@gls@body\glsentrylong\relax
4425     \expandafter\Glsentrylong\@gls@rest
4426   }else
4427     \expandafter\ifx\@gls@body\glsentryshort\relax
4428     \expandafter\Glsentryshort\@gls@rest
4429   }else
4430     \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.

```

4431     {%
4432       \let\glsentryshort\Glsentryshort
4433       \@glo@text
4434     }%
4435   }else

```

```

4436      \xmakefirstuc{\@glo@text}%
4437      \fi
4438      \fi
4439      \fi
4440  }%
4441  {%

```

Not an acronym

```

4442      \@Gls@entry@field{#1}{name}%
4443  }%
4444 }

```

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`

```

4445 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4446 \newrobustcmd*{\Glsentrydesc}[1]{%
4447   \@Gls@entry@field{#1}{desc}%
4448 }

```

Plural form:

`entrydescplural`

```

4449 \newcommand*{\glsentrydescplural}[1]{%
4450   \@gls@entry@field{#1}{descplural}%
4451 }

```

`entrydescplural`

```

4452 \newrobustcmd*{\Glsentrydescplural}[1]{%
4453   \@Gls@entry@field{#1}{descplural}%
4454 }

```

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`

```

4455 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}

```

`\Glsentrytext`

```

4456 \newrobustcmd*{\Glsentrytext}[1]{%
4457   \@Gls@entry@field{#1}{text}%
4458 }

```

Get the plural form:

\glentryplural

```
4459 \newcommand*{\glentryplural}[1]{%
4460   \@gls@entry@field{#1}{plural}%
4461 }
```

\Glsentryplural

```
4462 \newrobustcmd*{\Glsentryplural}[1]{%
4463   \@Gls@entry@field{#1}{plural}%
4464 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

\glentrysymbol

```
4465 \newcommand*{\glentrysymbol}[1]{%
4466   \@gls@entry@field{#1}{symbol}%
4467 }
```

\Glsentrysymbol

```
4468 \newrobustcmd*{\Glsentrysymbol}[1]{%
4469   \@Gls@entry@field{#1}{symbol}%
4470 }
```

Plural form:

trysymbolplural

```
4471 \newcommand*{\glentrysymbolplural}[1]{%
4472   \@gls@entry@field{#1}{symbolplural}%
4473 }
```

trysymbolplural

```
4474 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4475   \@Gls@entry@field{#1}{symbolplural}%
4476 }
```

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

```
4477 \newcommand*{\glentryfirst}[1]{%
4478   \@gls@entry@field{#1}{first}%
4479 }
```

\Glsentryfirst

```
4480 \newrobustcmd*{\Glsentryfirst}[1]{%
4481   \@Gls@entry@field{#1}{first}%
4482 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

entryfirstplural

```
4483 \newcommand*{\glsentryfirstplural}[1]{%
4484   \@gls@entry@field{#1}{firstpl}%
4485 }
```

entryfirstplural

```
4486 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4487   \@Gls@entry@field{#1}{firstpl}%
4488 }
```

entrytitlecase

```
4489 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4490   \glsfieldfetch{#1}{#2}{\@gls@value}%
4491   \xcapitalisewords{\@gls@value}%
4492 }
4493 \ifdef\texorpdfstring
4494 {
4495   \newcommand*{\glsentrytitlecase}[2]{%
4496     \texorpdfstring
4497       {\@glsentrytitlecase{#1}{#2}}%
4498       {\@gls@entry@field{#1}{#2}}%
4499   }
4500 }
4501 {
4502   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4503 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
4504 \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

```
4505 \newcommand*{\glsentrysort}[1]{%
4506   \@gls@entry@field{#1}{sort}%
4507 }
```

\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.

```
4508 \newcommand*{\glsentryuseri}[1]{%
4509   \@gls@entry@field{#1}{useri}%
4510 }
```

\Glsentryuseri

```
4511 \newrobustcmd*{\Glsentryuseri}[1]{%
```

```

4512 \@Gls@entry@field{#1}{useri}%
4513 }

```

`\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.

```

4514 \newcommand*{\glentryuserii}[1]{%
4515 \@Gls@entry@field{#1}{userii}%
4516 }

```

`\Glsentryuserii`

```

4517 \newrobustcmd*{\Glsentryuserii}[1]{%
4518 \@Gls@entry@field{#1}{userii}%
4519 }

```

`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.

```

4520 \newcommand*{\glentryuseriii}[1]{%
4521 \@Gls@entry@field{#1}{useriii}%
4522 }

```

`Glsentryuseriii`

```

4523 \newrobustcmd*{\Glsentryuseriii}[1]{%
4524 \@Gls@entry@field{#1}{useriii}%
4525 }

```

`\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.

```

4526 \newcommand*{\glentryuseriv}[1]{%
4527 \@Gls@entry@field{#1}{useriv}%
4528 }

```

`\Glsentryuseriv`

```

4529 \newrobustcmd*{\Glsentryuseriv}[1]{%
4530 \@Gls@entry@field{#1}{useriv}%
4531 }

```

`\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.

```

4532 \newcommand*{\glentryuserv}[1]{%
4533 \@Gls@entry@field{#1}{userv}%
4534 }

```

`\Glsentryuserv`

```

4535 \newrobustcmd*{\Glsentryuserv}[1]{%
4536 \@Gls@entry@field{#1}{userv}%
4537 }

```

`\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.

```
4538 \newcommand*{\glentryuservi}[1]{%
4539   \@gls@entry@field{#1}{uservi}%
4540 }
```

`\Glsentryuservi`

```
4541 \newrobustcmd*{\Glsentryuservi}[1]{%
4542   \@Gls@entry@field{#1}{uservi}%
4543 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4544 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4545 \newrobustcmd*{\Glsentryshort}[1]{%
4546   \@Gls@entry@field{#1}{short}%
4547 }
```

`glentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4548 \newcommand*{\glentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`Glsentryshortpl`

```
4549 \newrobustcmd*{\Glsentryshortpl}[1]{%
4550   \@Gls@entry@field{#1}{shortpl}%
4551 }
```

`\glentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4552 \newcommand*{\glentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4553 \newrobustcmd*{\Glsentrylong}[1]{%
4554   \@Gls@entry@field{#1}{long}%
4555 }
```

`\glentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4556 \newcommand*{\glentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4557 \newrobustcmd*{\Glsentrylongpl}[1]{%
4558   \@Gls@entry@field{#1}{longpl}%
4559 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4560 \newcommand*{\glsentryfull}[1]{%
4561   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4562 }
```

`\Glsentryfull`

```
4563 \newrobustcmd*{\Glsentryfull}[1]{%
4564   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4565 }
```

`\glsentryfullpl`

```
4566 \newcommand*{\glsentryfullpl}[1]{%
4567   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4568 }
```

`\Glsentryfullpl`

```
4569 \newrobustcmd*{\Glsentryfullpl}[1]{%
4570   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4571 }
```

`entrynumberlist` Displays the number list as is.

```
4572 \newcommand*{\glsentrynumberlist}[1]{%
4573   \glsdoifexists{#1}%
4574   {%
4575     \@gls@entry@field{#1}{numberlist}%
4576   }%
4577 }
```

`splaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4578 \@ifpackageloaded{hyperref} {%
4579   \newcommand*{\glsdisplaynumberlist}[1]{%
4580     \GlossariesWarning
4581     {%
4582       \string\glsdisplaynumberlist\space
4583       doesn't work with hyperref.^^JUsing
4584       \string\glsentrynumberlist\space instead%
4585     }%
4586     \glsentrynumberlist{#1}%
4587   }%
4588 }%
4589 {%
4590   \newcommand*{\glsdisplaynumberlist}[1]{%
4591     \glsdoifexists{#1}%
4592     {%
4593       \bgroup
```



```

4629 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4630 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

This key is only used by `\glsaddall`:

```

4631 \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`

```

4632 \newrobustcmd*{\glsadd}[2] [] {%

```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4633 \@gls@adjustmode
4634 \glsdoifexists{#2}%
4635 {%
4636 \def\@glsnumberformat{glsnumberformat}%
4637 \edef\@gls@counter{\csname glo@glstetoklabel{#2}@counter\endcsname}%
4638 \setkeys{glossadd}{#1}%

```

Store the entry's counter in `\theglsentrycounter`

```

4639 \@gls@saveentrycounter

```

Define sort key if necessary:

```

4640 \@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.

```

4641 \@@do@wrglossary{#2}%
4642 }%
4643 }

```

`@gls@adjustmode`

```

4644 \newcommand*{\@gls@adjustmode}{}
4645 \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`

```

4646 \newrobustcmd*{\glsaddall}[1] [] {%
4647 \edef\@glo@type{\@glo@types}%

```

```

4648 \setkeys{glossadd}{#1}%
4649 \forallglsentries[\@glo@type]{\@glo@entry}{%
4650   \glsadd[#1]{\@glo@entry}%
4651 }%
4652 }

```

```

\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.

```

4653 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4654   \forallglsentries[#1]{\@glo@entry}%
4655   {%
4656     \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4657   }%
4658 }

```

`\glsignore`

```

4659 \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 `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.`. This is done to prevent any problem characters in `\glsymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```

4660 \edef\glsopenbrace{\expandafter\@gobble\string\{ }

```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```

4661 \edef\glsclosebrace{\expandafter\@gobble\string\} }

```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4662 \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.

```
4663 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4664 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4665 \edef\glstildechar{\string~}
```

`@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4666 \ifglsxindy
4667   \newcommand*{\@glsfirstletter}{A}
4668 \fi
```

`letterAfterDigits` Sets the first letter to come after the digits 0,...,9. The starred version sanitizes.

```
4669 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}{%
4670   \@ifstar\s@GlsSetXdyFirstLetterAfterDigits\@GlsSetXdyFirstLetterAfterDigits}
4671 \ifglsxindy
4672   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4673     \renewcommand*{\@glsfirstletter}{#1}}
4674   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}[1]{%
4675     \renewcommand*{\@glsfirstletter}{#1}%
4676     \@onelevel@sanitize\@glsfirstletter
4677   }
4678 \else
4679   \newcommand*{\@GlsSetXdyFirstLetterAfterDigits}[1]{%
4680     \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4681   \newcommand*{\s@GlsSetXdyFirstLetterAfterDigits}{%
4682     \@GlsSetXdyFirstLetterAfterDigits
4683   }
4684 \fi
```

`numbergrouporder` Specifies the order of the number group.

```
4685 \ifglsxindy
4686   \newcommand*{\@xdynumbergrouporder}{:before \string"\@glsfirstletter\string"}
4687 \fi
```

`numberGroupOrder` Sets the relative location of the number group. The starred version sanitizes.

```
4688 \newcommand*{\GlsSetXdyNumberGroupOrder}[1]{%
4689   \@ifstar\s@GlsSetXdyNumberGroupOrder\@GlsSetXdyNumberGroupOrder
4690 }
4691 \ifglsxindy
4692   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4693     \renewcommand*{\@xdynumbergrouporder}{#1}%

```



```

4694 }
4695 \newcommand*{\s@GlsSetXdyNumberGroupOrder}[1]{%
4696   \renewcommand*{\@xdynumbergrouporder}{#1}%
4697   \@onelevel@sanitize\@xdynumbergrouporder
4698 }
4699 \else
4700   \newcommand*{\@GlsSetXdyNumberGroupOrder}[1]{%
4701     \glsnoindywarning\GlsSetXdyNumberGroupOrder}
4702   \newcommand*{\s@GlsSetXdyNumberGroupOrder}{%
4703     \@GlsSetXdyNumberGroupOrder}
4704 \fi

```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```

4705 \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.

```

4706 \ifglsxindy
4707   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4708     \renewcommand*{\@glsminrange}{#1}}
4709 \else
4710   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4711     \glsnoindywarning\GlsSetXdyMinRangeLength}
4712 \fi

```

`\writeist`

```

4713 \ifglsxindy

```

Code to use if xindy is required.

```

4714 \def\writeist{%

```

Define write register if not already defined

```

4715   \ifundef{\glswrite}{\newwrite\glswrite}{}%

```

Update attributes list

```

4716   \@gls@addpredefinedattributes

```

Open the file.

```

4717   \openout\glswrite=\istfilename

```

Write header comment at the start of the file

```

4718   \write\glswrite{;; xindy style file created by the glossaries

```

```

4719     package}%

```

```

4720   \write\glswrite{;; for document '\jobname' on

```

```

4721     \the\year-\the\month-\the\day}%

```

Specify the required styles

```

4722   \write\glswrite{^^J; required styles^^J}

```

```

4723   \@for\@xdystyle:=\@xdyrequiredstyles\do{%

```

```

4724     \ifx\@xdystyle\@empty

```

```

4725     \else

```

```

4726         \protected@write\glswrite{}\{(require
4727         \string"\@xdystyle.xdy\string")}\}%
4728     \fi
4729 }%

```

List the allowed attributes (possible values used by the format key)

```

4730     \write\glswrite{^^J%
4731     ; list of allowed attributes (number formats)^^J}%
4732     \write\glswrite{(define-attributes ((\@xdyattributes)))}%

```

Define any additional alphabets

```

4733     \write\glswrite{^^J; user defined alphabets^^J}%
4734     \write\glswrite{\@xdyuseralphabets}%

```

Define location classes.

```

4735     \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.

```

4736     \@for\@gls@classI:=\@gls@xdy@locationlist\do{%

```

Case where $\langle Hprefix \rangle$ is empty:

```

4737         \protected@write\glswrite{}\{(define-location-class
4738         \string"\@gls@classI\string"^^J\space\space\space
4739         (
4740             :sep "{"
4741             \csname \@gls@xdy@Lclass@\@gls@classI\endcsname\space
4742             :sep "}"
4743         )
4744         ^^J\space\space\space
4745         :min-range-length \@glsminrange^^J%
4746     )
4747 }%

```

Nested iteration over all classes:

```

4748     {%
4749     \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4750         \protected@write\glswrite{}\{(define-location-class
4751         \string"\@gls@classII-\@gls@classI\string"
4752         ^^J\space\space\space
4753         (
4754             :sep "{"
4755             \csname \@gls@xdy@Lclass@\@gls@classII\endcsname\space
4756             :sep "{"
4757             \csname \@gls@xdy@Lclass@\@gls@classI\endcsname\space
4758             :sep "}"
4759         )
4760         ^^J\space\space\space
4761         :min-range-length \@glsminrange^^J%
4762     )
4763     }%
4764 }%

```

```

4765     }%
4766 }%

```

User defined location classes (needs checking for new location format).

```

4767 \write\glswrite{^^J; user defined location classes}%
4768 \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.)

```

4769 \write\glswrite{^^J; define cross-reference class^^J}%
4770 \write\glswrite{(define-crossref-class \string"see\string"
4771 :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.)

```

4772 \write\glswrite{(markup-crossref-list
4773 :class \string"see\string"^^J\space\space\space
4774 :open \string"\string\glsseeformat\string"
4775 :close \string"{}\string")}%

```

Provide hook to write extra material here (used by glossaries-extra to define a seealso class).

```

4776 \@xdycrossrefhook

```

List the order to sort the classes.

```

4777 \write\glswrite{^^J; define the order of the location classes}%
4778 \write\glswrite{(define-location-class-order
4779 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4780 \write\glswrite{^^J; define the glossary markup^^J}%
4781 \write\glswrite{(markup-index^^J\space\space\space
4782 :open \string"\string
4783 \glossarysection[\string\glossarytoctitle]{\string
4784 \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)

```

4785 \@for\@this@ctr:=\@xdycounters\do{%
4786   {%
4787     \@for\@this@attr:=\@xdyattributelist\do{%
4788       \protected@write\glswrite{}\string\providecommand*%
4789       \expandafter\string
4790       \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4791       {%
4792         \string\setentrycounter
4793         [\expandafter\@gobble\string\#1]{\@this@ctr}%
4794         \expandafter\string

```

```

4795         \csname\@this@attr\endcsname
4796         {\expandafter\@gobble\string\#2}%
4797     }%
4798 }%
4799 }%
4800 }%
4801 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4802 \write\glswrite{%
4803     \string\begin
4804     {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4805     \space\space:close \string"\glpercentchar\glstildechar n\string
4806     \end{theglossary}\string\glossarypostamble
4807     \glstildechar n\string" ^^J\space\space\space
4808     :tree)}}%

```

Specify what to put between letter groups

```

4809 \write\glswrite{(markup-letter-group-list
4810     :sep \string"\string\glsgroupskip\glstildechar n\string"))}%

```

Specify what to put between entries

```

4811 \write\glswrite{(markup-indexentry
4812     :open \string"\string\relax \string\glresetentrylist
4813     \glstildechar n\string"))}%

```

Specify how to format entries

```

4814 \write\glswrite{(markup-locclass-list :open
4815     \string"\glsoopenbrace\string\glossaryentrynumbers
4816     \glsoopenbrace\string\relax\space \string"^^J\space\space\space
4817     :sep \string", \string"
4818     :close \string"\glsclosebrace\glsclosebrace\string"))}%

```

Specify how to separate location numbers

```

4819 \write\glswrite{(markup-locref-list
4820     :sep \string"\string\delimN\space\string"))}%

```

Specify how to indicate location ranges

```

4821 \write\glswrite{(markup-range
4822     :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.

```

4823 \@onelevel@sanitize\gls@suffiF
4824 \@onelevel@sanitize\gls@suffiFF
4825 \ifx\gls@suffiF\@empty
4826 \else
4827     \write\glswrite{(markup-range
4828         :close "\gls@suffiF" :length 1 :ignore-end)}}%
4829 \fi
4830 \ifx\gls@suffiFF\@empty

```

```

4831 \else
4832 \write\glswrite{(markup-range
4833 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4834 \fi

Specify how to format locations.
4835 \write\glswrite{^^J; define format to use for locations^^J}%
4836 \write\glswrite{@xdylocref}%

Specify how to separate letter groups.
4837 \write\glswrite{^^J; define letter group list format^^J}%
4838 \write\glswrite{(markup-letter-group-list
4839 :sep \string\string\glsgroupskip\glstildechar n\string)}}%

Define letter group headings.
4840 \write\glswrite{^^J; letter group headings^^J}%
4841 \write\glswrite{(markup-letter-group
4842 :open-head \string\string\glsgroupheading
4843 \glsopenbrace\string^^J\space\space\space
4844 :close-head \string\glsclosebrace\string)}}%

Define additional letter groups.
4845 \write\glswrite{^^J; additional letter groups^^J}%
4846 \write\glswrite{@xdylettergroups}%

Define additional sort rules
4847 \write\glswrite{^^J; additional sort rules^^J}%
4848 \write\glswrite{@xdysortrules}%

Hook for any additional information:
4849 \@gls@writeisthook

Close the style file
4850 \closeout\glswrite

Suppress any further calls.
4851 \let\writeist\relax
4852 }
4853 \else

Code to use if makeindex is required.
4854 \edef\@gls@actualchar{\string?}
4855 \edef\@gls@encapchar{\string|}
4856 \edef\@gls@levelchar{\string!}
4857 \edef\@gls@quotechar{\string"}%
4858 \let\GlsSetQuote\gls@nosetquote
4859 \def\writeist{\relax
4860 \ifundef{\glswrite}{\newwrite\glswrite}{\relax
4861 \openout\glswrite=\istfilename
4862 \write\glswrite{\glspercentchar\space makeindex style file
4863 created by the glossaries package}
4864 \write\glswrite{\glspercentchar\space for document
4865 '\jobname' on \the\year-\the\month-\the\day}

```

```

4866 \write\glswrite{actual '@gls@actualchar'}
4867 \write\glswrite{encap '@gls@encapchar'}
4868 \write\glswrite{level '@gls@levelchar'}
4869 \write\glswrite{quote '@gls@quotechar'}
4870 \write\glswrite{keyword \string\string\glossaryentry\string}
4871 \write\glswrite{preamble \string\string\glossarysection[\string
4872 \glossarytoctitle]{\string\glossarytitle}\string
4873 \glossarypreamble\string\n\string\begin{theglossary}\string
4874 \glossaryheader\string\n\string}
4875 \write\glswrite{postamble \string\string%\string\n\string
4876 \end{theglossary}\string\glossarypostamble\string\n
4877 \string}
4878 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
4879 \string}
4880 \write\glswrite{item_0 \string\string%\string\n\string}
4881 \write\glswrite{item_1 \string\string%\string\n\string}
4882 \write\glswrite{item_2 \string\string%\string\n\string}
4883 \write\glswrite{item_01 \string\string%\string\n\string}
4884 \write\glswrite{item_x1
4885 \string\string\relax \string\glsresetentrylist\string\n
4886 \string}
4887 \write\glswrite{item_12 \string\string%\string\n\string}
4888 \write\glswrite{item_x2
4889 \string\string\relax \string\glsresetentrylist\string\n
4890 \string}

4891 \write\glswrite{delim_0 \string\string\{\string
4892 \glossaryentrynumbers\string\{\string\relax \string}
4893 \write\glswrite{delim_1 \string\string\{\string
4894 \glossaryentrynumbers\string\{\string\relax \string}
4895 \write\glswrite{delim_2 \string\string\{\string
4896 \glossaryentrynumbers\string\{\string\relax \string}
4897 \write\glswrite{delim_t \string\string\}\string\}\string}
4898 \write\glswrite{delim_n \string\string\delimN \string}
4899 \write\glswrite{delim_r \string\string\delimR \string}
4900 \write\glswrite{headings_flag 1}
4901 \write\glswrite{heading_prefix
4902 \string\string\glsgroupheading\string\{\string}
4903 \write\glswrite{heading_suffix
4904 \string\string\}\string\relax
4905 \string\glsresetentrylist \string}
4906 \write\glswrite{symhead_positive \string\glsymbols\string}
4907 \write\glswrite{numhead_positive \string\glsnumbers\string}
4908 \write\glswrite{page_compositor \string\glscompositor\string}
4909 \@gls@escbsdq\gls@suffixF
4910 \@gls@escbsdq\gls@suffixFF
4911 \ifx\gls@suffixF\@empty
4912 \else
4913 \write\glswrite{suffix_2p \string\gls@suffixF\string}
4914 \fi

```

```

4915 \ifx\gls@suffixFF\@empty
4916 \else
4917 \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4918 \fi

```

Hook for any additional information:

```

4919 \@gls@writeisthook

```

Close the file and disable \writeist.

```

4920 \closeout\glswrite
4921 \let\writeist\relax
4922 }
4923 \fi

```

SetWriteIstHook Allow user to append information to the style file.

```

4924 \newcommand*{\GlsSetWriteIstHook}[1]{\renewcommand*{\@gls@writeisthook}{#1}}
4925 \@onlypremakeg\GlsSetWriteIstHook

```

ls@writeisthook

```

4926 \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.

```

4927 \ifglxsindy
4928 \newcommand*{\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4929 \newcommand*{\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4930 \else
4931 \newcommand*{\GlsSetQuote}[1]{\edef\@gls@quotechar{\string#1}%

```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```

4932 \@ifpackageloaded{tracklang}%
4933 {%
4934 \IfTrackedLanguage{german}%
4935 {%
4936 \def\@gls@extramakeindexopts{-g}%
4937 }%
4938 }%
4939 }%
4940 {}%

```

Need to redefine \@gls@checkquote

```

4941 \edef\@gls@docheckquotedef{%
4942 \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
4943 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4944 \noexpand\toks@={####1}%
4945 \noexpand\ifx\noexpand\null####2\noexpand\null
4946 \noexpand\ifx\noexpand\null####3\noexpand\null
4947 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4948 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4949 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%

```

```

4950 \noexpand\else
4951 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4952 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4953 \noexpand\@gls@quotetchar\noexpand\@gls@quotetchar
4954 \noexpand\@gls@quotetchar\noexpand\@gls@quotetchar}%
4955 \noexpand\def\noexpand\@gls@checkquote{%
4956 \noexpand\@gls@checkquote####3\noexpand\null}%
4957 \noexpand\fi
4958 \noexpand\else
4959 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4960 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4961 \noexpand\@gls@quotetchar\noexpand\@gls@quotetchar}%
4962 \noexpand\ifx\noexpand\null####3\noexpand\null
4963 \noexpand\def\noexpand\@gls@checkquote{%
4964 \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
4965 \noexpand\else
4966 \noexpand\def\noexpand\@gls@checkquote{%
4967 \noexpand\@gls@checkquote####2#1####3\noexpand\null}%
4968 \noexpand\fi
4969 \noexpand\fi
4970 \noexpand\@gls@checkquote
4971 }%
4972 }%
4973 \@gls@docheckquotedef
4974 \edef\@gls@docheckquotedef{%
4975 \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
4976 \noexpand\def\noexpand\@gls@checkedmkidx{%
4977 \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
4978 #1#1\noexpand\null
4979 \noexpand\expandafter\noexpand\@gls@updatechecked
4980 \noexpand\@gls@checkedmkidx{####1}%
4981 \noexpand\def\noexpand\@gls@checkedmkidx{%
4982 \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
4983 \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
4984 \noexpand\null
4985 \noexpand\expandafter\noexpand\@gls@updatechecked
4986 \noexpand\@gls@checkedmkidx{####1}%
4987 \noexpand\def\noexpand\@gls@checkedmkidx{%
4988 \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
4989 \noexpand\?\noexpand\?\noexpand\null
4990 \noexpand\expandafter\noexpand\@gls@updatechecked
4991 \noexpand\@gls@checkedmkidx{####1}%
4992 \noexpand\def\noexpand\@gls@checkedmkidx{%
4993 \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
4994 \noexpand\?\noexpand\?\noexpand\null
4995 \noexpand\expandafter\noexpand\@gls@updatechecked
4996 \noexpand\@gls@checkedmkidx{####1}%
4997 \noexpand\def\noexpand\@gls@checkedmkidx{%
4998 \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil

```



```

4999      \noexpand|\noexpand|\noexpand\null
5000      \noexpand\expandafter\noexpand\@gls@updatechecked
5001      \noexpand\@gls@checkedmkidx{####1}%
5002      \noexpand\def\noexpand\@gls@checkedmkidx{%
5003      \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
5004      \noexpand\\ \noexpand\\ \noexpand\null
5005      \noexpand\expandafter\noexpand\@gls@updatechecked
5006      \noexpand\@gls@checkedmkidx{####1}%
5007      \noexpand\def\noexpand\@gls@checkedmkidx{%
5008      \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
5009      \noexpand!\noexpand!\noexpand\null
5010      \noexpand\expandafter\noexpand\@gls@updatechecked
5011      \noexpand\@gls@checkedmkidx{####1}%
5012  }%
5013 }%
5014 \@gls@docheckquotedef
5015 \edef\@gls@docheckquotedef{%
5016   \noexpand\def\noexpand\@gls@checkescquote####1%
5017   \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
5018   ####3\noexpand\null{%
5019   \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
5020   \noexpand\toks@={####1}%
5021   \noexpand\ifx\noexpand\null####2\noexpand\null
5022   \noexpand\ifx\noexpand\null####3\noexpand\null
5023   \noexpand\edef\noexpand\@gls@checkedmkidx{%
5024     \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
5025   \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
5026   \noexpand\else
5027   \noexpand\edef\noexpand\@gls@checkedmkidx{%
5028     \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5029     \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5030       \csname#1\endcsname}\noexpand\@gls@quotechar
5031     \noexpand\@gls@quotechar\noexpand\string\expandonce{%
5032       \csname#1\endcsname}\noexpand\@gls@quotechar}%
5033   \noexpand\def\noexpand\@gls@checkescquote{%
5034     \noexpand\@gls@checkescquote####3\noexpand\null}%
5035   \noexpand\fi
5036   \noexpand\else
5037   \noexpand\edef\noexpand\@gls@checkedmkidx{%
5038     \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
5039     \noexpand\@gls@quotechar\noexpand\string
5040     \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
5041   \noexpand\ifx\noexpand\null####3\noexpand\null
5042   \noexpand\def\noexpand\@gls@checkescquote{%
5043     \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
5044     \expandonce{\csname#1\endcsname}\noexpand\null}%
5045   \noexpand\else
5046   \noexpand\def\noexpand\@gls@checkescquote{%
5047     \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%

```

```

5048         ###3\noexpand\null}%
5049     \noexpand\fi
5050     \noexpand\fi
5051     \noexpand\@@gls@checkescquote
5052 }%
5053 }%
5054 \@gls@docheckquotedef
5055 }
5056 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
5057 {\string\GlsSetQuote\space not permitted here}%
5058 {Move \string\GlsSetQuote\space earlier in the preamble, as
5059  soon as possible after glossaries.sty has been loaded}}
5060 \fi

```

ramakeindexopts

```

5061 \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`

```

5062 \newcommand{\noist}{%
  Update attributes list
5063 \@gls@addpredefinedattributes
5064 \let\writeist\relax
5065 }

```

`\@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`

```

5066 \newcommand*{\@makeglossary}[1]{%
5067 \ifglossaryexists{#1}%
5068 {%

```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```

5069 \ifglssavewrites
5070 \expandafter\newtoks\csname glo@#1@filetok\endcsname
5071 \else
5072 \expandafter\newwrite\csname glo@#1@file\endcsname

```

```

5073     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
5074     \fi
5075     \@gls@renewglossary
5076     \writeist
5077 }%
5078 {%
5079     \PackageError{glossaries}%
5080     {Glossary type ‘#1’ not defined}%
5081     {New glossaries must be defined before using \string\makeglossary}%
5082 }%
5083 }

```

`\@glsopenfile` Open write file associated with the given glossary.

```

5084 \newcommand*{\@glsopenfile}[2]{%
5085     \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
5086     \PackageInfo{glossaries}{Writing glossary file
5087         \jobname.\csname @glotype@#2@out\endcsname}%
5088 }

```

`\@closegls`

```

5089 \newcommand*{\@closegls}[1]{%
5090     \closeout\csname glo@#1@file\endcsname
5091 }

```

`\@gls@automake`

```

5092 \ifglxindy
5093 \newcommand*{\@gls@automake}[1]{%
5094     \ifglossaryexists{#1}
5095     {%
5096         \@closegls{#1}%
5097         \ifdefstring{\glsorder}{letter}%
5098         {\def\@gls@order{-M ord/letorder }}%
5099         {\let\@gls@order\@empty}%
5100         \ifcsundef{@xdy@#1@language}%
5101         {\let\@gls@langmod\@xdy@main@language}%
5102         {\letcs\@gls@langmod{@xdy@#1@language}}%
5103         \edef\@gls@dothiswrite{\noexpand\write18{xindy
5104             -I xindy
5105             \@gls@order
5106             -L \@gls@langmod\space
5107             -M \@gls@istfilebase\space
5108             -C \@gls@codepage\space
5109             -t \jobname.\csuse{@glotype@#1@log}
5110             -o \jobname.\csuse{@glotype@#1@in}
5111             \jobname.\csuse{@glotype@#1@out}}}%
5112         }%
5113         \@gls@dothiswrite
5114     }%
5115     {%

```

```

5116     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5117 }%
5118 }
5119 \else
5120 \newcommand*{\@gls@automake}[1]{%
5121   \ifglossaryexists{#1}
5122   {%
5123     \@closegls{#1}%
5124     \ifdefstring{\glsorder}{letter}%
5125     {\def\@gls@order{-l }}%
5126     {\let\@gls@order\@empty}%
5127     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5128       -s \istfilename\space
5129       -t \jobname.\csuse{\glotype@#1@log}
5130       -o \jobname.\csuse{\glotype@#1@in}
5131       \jobname.\csuse{\glotype@#1@out}}}%
5132   }%
5133   \@gls@dothiswrite
5134 }%
5135 {%
5136   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5137 }%
5138 }
5139 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

5140 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

5141 \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`

```

5142 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```

5143   \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.

```

5144   \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{} }
5145   \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{} }

```

If `\@gls@extramakeindexopts` has been defined, write it:

```

5146   \ifundef\@gls@extramakeindexopts
5147   {}%
5148   {%
5149     \protected@write\@auxout{}{\string\providecommand

```

```

5150     \string\@gls@extramakeindexopts[1]{}}
5151     \protected@write\@auxout{}{\string\@gls@extramakeindexopts
5152     {\@gls@extramakeindexopts}}%
5153 }%

Write the name of the style file to the aux file (needed by makeglossaries)
5154 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
5155 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

Iterate through each glossary type and activate it.
5156 \@for\@glo@type:=\@glo@types\do{%
5157     \ifthenelse{equal{\@glo@type}{}}{ }{%
5158         \@makeglossary{\@glo@type}}%
5159 }%

New glossaries must be created before \makeglossaries so disable \newglossary.
5160 \renewcommand*\newglossary[4][]{%
5161     \PackageError{glossaries}{New glossaries
5162     must be created before \string\makeglossaries}{You need
5163     to move \string\makeglossaries\space after all your
5164     \string\newglossary\space commands}}%

Any subsequence instances of this command should have no effect
5165 \let\@makeglossary\relax
5166 \let\makeglossary\relax
5167 \let\makeglossaries\relax

Disable all commands that have no effect after \makeglossaries
5168 \@disable@onlypremakeg

Allow see key:
5169 \let\gls@checkseeallowed\relax

Suppress warning about no \makeglossaries
5170 \let\warn@nomakeglossaries\relax

Activate warning about missing \printglossary
5171 \def\warn@noprintglossary{%
5172     \ifdefstring{\@glo@types}{,}%
5173     {%
5174         \GlossariesWarningNoLine{No glossaries have been defined}%
5175     }%
5176     {%
5177         \GlossariesWarningNoLine{No \string\printglossary\space
5178         or \string\printglossaries\space
5179         found. ^^J(Remove \string\makeglossaries\space if you
5180         don't want any glossaries.) ^^JThis document will not
5181         have a glossary}%
5182     }%
5183 }%

Declare list parser for \glsdisplaynumberlist
5184 \ifglssavenumberlist

```

```

5185 \edef\@gls@dodolistparser{\noexpand\DeclareListParser
5186   {\noexpand\glsnumlistparser}{\delimN}}}%
5187 \@gls@dodolistparser
5188 \fi

```

Prevent user from also using `\makenoidxglossaries`

```
5189 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in `\printgloss` family:

```

5190 \renewcommand*{\@printgloss@setsort}{%
5191   \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5192 }%

```

Check the automake setting:

```

5193 \ifglautomake
5194   \renewcommand*{\@gls@doautomake}{%
5195     \@for\@gls@type:=\@glo@types\do{%
5196       \ifdefempty{\@gls@type}{}%
5197       {\@gls@automake{\@gls@type}}%
5198     }%
5199   }%
5200 \fi

```

Check the sort setting:

```

5201 \@glo@check@sortallowed\makeglossaries
5202 }

```

Must occur in the preamble:

```
5203 \@onlypreamble{\makeglossaries}
```

`\glswrite` The definition of `\glswrite` has now been moved to `\makeglossaries` so that it's only defined if needed.

The `\makeglossary` command is redefined to be identical to `\makeglossaries`. (This is done to reinforce the message that you must either use `\@makeglossary` for all the glossaries or for none of them.)

`\makeglossary`

```
5204 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```

5205 \AtEndDocument{%
5206   \warn@nomakeglossaries
5207   \warn@noprintglossary
5208 }

```

`noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
5209 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5210 \renewcommand{\@gls@noref@warn}[1]{%
5211   \GlossariesWarning{Empty glossary for
5212   \string\printnoidxglossary[type={##1}].
5213   Rerun may be required (or you may have forgotten to use
5214   commands like \string\gls)}}%
5215 }%
```

Don't escape makeindex/xindy characters

```
5216 \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
5217 \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5218 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5219 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5220 \renewcommand{\@do@seeglossary}[2]{%
5221   \edef\@gls@label{\glsdetoklabel{##1}}%
5222   \protected@write\@auxout{}{%
5223     \string\@gls@reference
5224     {\csname glo@\@gls@label @type\endcsname}%
5225     {\@gls@label}%
5226     {%
5227       \string\glsseeformat##2}%
5228     }%
5229   }%
5230 }%
```

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.

```
5231 \AtBeginDocument
5232 {%
5233   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5234 }%
```

Change warning about no glossaries

```
5235 \def\warn@noprntglossary{%
5236   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5237   or \string\printnoidxglossaries ^^J
5238   found. (Remove \string\makenoidxglossaries\space if you
5239   don't want any glossaries.)^^JThis document will not have a glossary}%
5240 }%
```

Suppress warning about no \makeglossaries

```
5241 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5242 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5243 \renewcommand*\@printgloss@setsort}{%
5244 \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
5245 \def\@glo@sorttype{\@glo@default@sorttype}%
5246 }%
```

All entries must be defined in the preamble:

```
5247 \renewcommand*\new@glossaryentry[2]{%
5248 \PackageError{glossaries}{Glossary entries must be
5249 defined in the preamble^^Jwhen you use
5250 \string\makenoidxglossaries}%
5251 {Either move your definitions to the preamble or use
5252 \string\makeglossaries}%
5253 }%
```

Redefine \glsentrynumberlist

```
5254 \renewcommand*\glsentrynumberlist[1]{%
5255 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5256 \ifdef\@gls@loclist
5257 {%
5258 \glsnoidxloclist{\@gls@loclist}%
5259 }%
5260 {%
5261 ??\glsdoifexists{##1}%
5262 {%
5263 \GlossariesWarning{Missing location list for ‘##1’. Either
5264 a rerun is required or you haven’t referenced the entry}%
5265 }%
5266 }%
5267 }%
```

Redefine \glsdisplaynumberlist

```
5268 \renewcommand*\glsdisplaynumberlist[1]{%
5269 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5270 \ifdef\@gls@loclist
5271 {%
5272 \def\@gls@noidxloclist@sep{%
5273 \def\@gls@noidxloclist@sep{%
5274 \def\@gls@noidxloclist@sep{%
5275 \glsnumlistsep
5276 }%
5277 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5278 }%
5279 }%
5280 \def\@gls@noidxloclist@finalsep{}%
5281 \def\@gls@noidxloclist@prev{}%
5282 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5283 \@gls@noidxloclist@finalsep
5284 \@gls@noidxloclist@prev
```



```

5285 }%
5286 {%
5287   ??\glsdoifexists{##1}%
5288   {%
5289     \GlossariesWarning{Missing location list for ‘##1’. Either
5290       a rerun is required or you haven’t referenced the entry}%
5291   }%
5292 }%
5293 }%

```

Provide a generic way of iterating through the number list:

```

5294 \renewcommand*{\glsnumberlistloop}[3]{%
5295   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5296   \let\@gls@org\glsnoidxdisplayloc\glsnoidxdisplayloc
5297   \let\@gls@org\glsseeformat\glsseeformat
5298   \let\glsnoidxdisplayloc##2\relax
5299   \let\glsseeformat##3\relax
5300   \ifdef\@gls@loclist
5301   {%
5302     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5303   }%
5304   {%
5305     ??\glsdoifexists{##1}%
5306     {%
5307       \GlossariesWarning{Missing location list for ‘##1’. Either
5308         a rerun is required or you haven’t referenced the entry}%
5309     }%
5310   }%
5311   \let\glsnoidxdisplayloc\@gls@org\glsnoidxdisplayloc
5312   \let\glsseeformat\@gls@org\glsseeformat
5313 }%

```

Modify sanitize sort function

```

5314 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5315 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5316 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5317 \@glo@check@sortallowed\makenoidxglossaries
5318 }

```

Preamble-only command:

```

5319 \@onlypreamble{\makenoidxglossaries}

```

`\glsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```

5320 \newcommand*{\glsnumberlistloop}[2]{%
5321   \PackageError{glossaries}{\string\glsnumberlistloop\space
5322     only works with \string\makenoidxglossaries}{}%
5323 }

```

```

listloophandler  Handler macro for \glsnumberlistloop. (The argument should be in the form \glsnoidxdisplayloc
                  {\langle prefix\rangle}{\langle counter\rangle}{\langle format\rangle}{\langle n\rangle})
5324 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5325   #1%
5326 }

@makeglossaries  Can't use both \makeglossaries and \makenoidxglossaries
5327 \newcommand*{\@no@makeglossaries}{%
5328   \PackageError{glossaries}{You can't use both
5329     \string\makeglossaries\space and \string\makenoidxglossaries}%
5330   {Either use one or other (or none) of those commands but not both
5331     together.}%
5332 }

@gls@noref@warn  Warning when no instances of \@gls@reference found.
5333 \newcommand{\@gls@noref@warn}[1]{%
5334   \GlossariesWarning{\string\makenoidxglossaries\space
5335     is required to make \string\printnoidxglossary[type={#1}] work}%
5336 }

@s@noidxglossary  Write the glossary information to the aux file:
5337 \newcommand*{\gls@noidxglossary}{%
5338   \protected@write\@auxout{}{%
5339     \string\@gls@reference
5340     {\csname glo@\@gls@label @type\endcsname}%
5341     {\@gls@label}%
5342     {\string\glsnoidxdisplayloc
5343       {\@glo@counterprefix}%
5344       {\@gls@counter}%
5345       {\@glsnumberformat}%
5346       {\@glslocref}%
5347     }%
5348   }%
5349 }

```

1.14 Writing information to associated files

`\istfile` Deprecated.

```
5350 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if `savewrites=true`.

```

5351 \AtEndDocument{%
5352   \glswritefiles
5353 }

```

`\@glswritefiles` Only write the files if `savewrites=true`

```
5354 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
5355 \forall glossaries{\@glo@type}{%
    Check for empty glossaries (patch provided by Patrick Häcker)
5356 \ifcsundef{glo@\@glo@type @filetok}%
5357 {%
5358 \def\gls@tmp{}}%
5359 }%
5360 {%
5361 \edef\gls@tmp{\expandafter\the
5362 \csname glo@\@glo@type @filetok\endcsname}%
5363 }%
5364 \ifx\gls@tmp\@empty
5365 \ifx\@glo@type\glsdefaulttype
5366 \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5367 entries.^^JRemember to use package option ‘nomain’ if
5368 you
5369 don’t want to^^Juse the main glossary}%
5370 \else
5371 \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5372 entries}%
5373 \fi
5374 \else
5375 \@glsopenfile{\glswrite}{\@glo@type}%
5376 \immediate\write\glswrite{%
5377 \expandafter\the
5378 \csname glo@\@glo@type @filetok\endcsname}%
5379 \immediate\closeout\glswrite
5380 \fi
5381 }%
5382 }
```

As from v4.10, the `\glossary` command is 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).

In v4.10, the redefinition of `\glossary` was removed since it wasn’t intended as a user level command, 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.)

`\glossary`

```
5383 \if@gls@docloaded
5384 \else
5385 \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
5386 \fi
```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
5387 \newcommand*{\gls@glossary}[1]{%
5388   \@gls@glossary{#1}%
5389 }
```

`\@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 `\glspercentchar` would be needed with the sort sanitization switched off.) This means that it's safe to simply ignore the second argument.

```
5390 \newcommand*{\@gls@glossary}[2]{%
5391   \if@gls@debug
5392     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5393   \fi
5394 }
```

This is a convenience command to set `\@gls@glossary`. It's used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`\@renewglossary`

```
5395 \newcommand{\@gls@renewglossary}{%
5396   \gdef\@gls@glossary##1{\@bsphack\begin@group\gls@wrglossary{##1}}%
5397   \let\@gls@renewglossary\@empty
5398 }
```

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`

```

5399 \newcommand*{\gls@wrglossary}[2]{%
5400   \ifglssavewrites
5401     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5402     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5403       \expandafter{\@gls@tmp^^J}%
5404   \else
5405     \ifcsdef{glo@#1@file}%
5406     {%
5407       \expandafter\protected@write\csname glo@#1@file\endcsname{%
5408         \gls@disablepagerefexpansion}{#2}%
5409     }%
5410     {%
5411       \ifignoredglossary{#1}{}%
5412       {%
5413         \GlossariesWarning{No file defined for glossary ‘#1’}%
5414       }%
5415     }%
5416   \fi
5417 \endgroup\@esphack
5418 }

```

\@do@wrglossary

```

5419 \newcommand*{\@do@wrglossary}[1]{%
5420   \glswriteentry{#1}{\@do@wrglossary{#1}}%
5421 }

```

\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.

```

5422 \newcommand*{\glswriteentry}[2]{%
5423   \ifglindexonlyfirst
5424     \ifglused{#1}{}{#2}%
5425   \else
5426     #2%
5427   \fi
5428 }

```

protected@pagefmts List of page formats to be protected against expansion.

```

5429 \newcommand{\gls@protected@pagefmts}{%
5430   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5431 }

```

pagerefexpansion

```

5432 \newcommand*{\gls@disablepagerefexpansion}{%
5433   \@for\@gls@this:=\gls@protected@pagefmts\do
5434   {%
5435     \expandafter\let\@gls@this\relax
5436   }%
5437 }

```

```

\gls@alphpage
5438 \newcommand*{\gls@alphpage}{\@alph\c@page}

\gls@Alphpage
5439 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

\gls@numberpage
5440 \newcommand*{\gls@numberpage}{\number\c@page}

\gls@arabicpage
5441 \newcommand*{\gls@arabicpage}{\@arabic\c@page}

\gls@romanpage
5442 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

\gls@Romanpage
5443 \newcommand*{\gls@Romanpage}{\@Roman\c@page}

```

```

protectedpagefmt \glsaddprotectedpagefmt{<cs name>}

```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument ($\langle csname \rangle \c@page$ must be valid).

```

5444 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5445   \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}}%
5446   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5447   \eappto\@wrglossarynumberhook{%
5448     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5449     \expandonce{\csname#1\endcsname}%
5450     \noexpand\def\expandonce{\csname#1\endcsname}{%
5451       \noexpand\@wrglossary@pageformat
5452       \expandonce{\csname gls#1page\endcsname}%
5453       \expandonce{\csname org@gls#1\endcsname}%
5454     }%
5455   }%
5456 }

```

```

ssarynumberhook Hook used by \@@do@wrglossary
5457 \newcommand*\@wrglossarynumberhook{}

```

```

sary@pageformat
5458 \newcommand{\@wrglossary@pageformat}[3]{%
5459   \ifx#3\c@page #1\else #2#3\fi
5460 }

```

`@@do@wrglossary` Write the glossary entry in the appropriate format.

```
5461 \newcommand*{\@@do@wrglossary}[1]{%
5462   \ifglseclocations
5463     \@@do@esc@wrglossary{#1}%
5464   \else
5465     \@@do@noesc@wrglossary{#1}%
5466   \fi
5467 }
```

`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.

```
5468 \newcommand*{\@@do@noesc@wrglossary}[1]{%
```

Don't fully expand yet.

```
5469   \expandafter\def\expandafter\@glslocdef\expandafter{\theglsentrycounter}%
5470   \expandafter\def\expandafter\@glsHlocdef\expandafter{\theHglentrycounter}%
```

Find the prefix if `\@glsHlocdef` and `\@glslocdef` aren't the same.

```
5471   \ifx\@glsHlocdef\@glslocdef
5472     \def\@glo@counterprefix{}%
5473   \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.)

```
5474     \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5475       {\@glslocdef}{\@glsHlocdef}}%
5476     }%
5477     \@do@gls@getcounterprefix
5478   \fi
```

De-tok label if required

```
5479   \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5480   \@@do@@wrglossary
5481 }
```

`owprimitivemods` Conditional to determine whether or not `\@@do@esc@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
5482 \newif\ifglswrallowprimitivemods
5483 \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).

```
5484 \newcommand*{\@@do@esc@wrglossary}[1]{% please read documented code!
5485   \begingroup
```


First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions (scoped):

```
5486 \let\gls@orgthe\the
5487 \let\gls@orgnumber\number
5488 \let\gls@orgarabic\@arabic
5489 \let\gls@orgromannumeral\romannumeral
5490 \let\gls@orgalph\@alph
5491 \let\gls@orgAlph\@Alph
5492 \let\gls@orgRoman\@Roman
```

Redefine:

```
5493 \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.

```
5494 \def\gls@the##1{%
5495   \ifx##1\c@page \gls@numberpage\else\gls@orgthe##1\fi}%
5496 \def\the{\expandafter\gls@the}%
5497 \def\gls@number##1{%
5498   \ifx##1\c@page \gls@numberpage\else\gls@orgnumber##1\fi}%
5499 \def\number{\expandafter\gls@number}%
5500 \fi
5501 \def\@arabic##1{%
5502   \ifx##1\c@page \gls@arabicpage\else\gls@orgarabic##1\fi}%
5503 \def\romannumeral##1{%
5504   \ifx##1\c@page \gls@romanpage\else\gls@orgromannumeral##1\fi}%
5505 \def\@Roman##1{%
5506   \ifx##1\c@page \gls@Romanpage\else\gls@orgRoman##1\fi}%
5507 \def\@alph##1{%
5508   \ifx##1\c@page \gls@alphpage\else\gls@orgalph##1\fi}%
5509 \def\@Alph##1{%
5510   \ifx##1\c@page \gls@Alphpage\else\gls@orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5511 \@wrglossarynumberhook
```

Prevent expansion:

```
5512 \gls@disablepagerefexpansion
```

Now store location in `\@glslocref`:

```
5513 \protected@xdef\@glslocref{\theglsentrycounter}%
5514 \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.

```
5515 \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
5516 \expandafter\ifx\theglsentrycounter\theglsentrycounter\relax
5517 \def\@glo@counterprefix{}%

```

```

5518 \else
5519   \protected@edef\@glsHlocref{\theHglSentrycounter}%
5520   \@gls@checkmkidxchars\@glsHlocref
5521   \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5522     {\@glslocref}{\@glsHlocref}%
5523   }%
5524   \@do@gls@getcounterprefix
5525 \fi

```

De-tok label if required

```

5526 \edef\@gls@label{\glsdetoklabel{#1}}%

```

Write the information to file:

```

5527 \@do@@wrglossary
5528 }

```

@do@@wrglossary

```

5529 \newcommand*{\@do@@wrglossary}{%

```

Determine whether to use xindy or makeindex syntax

```

5530 \ifglSxindy

```

Need to determine if the formatting information starts with a (or) indicating a range.

```

5531 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5532 \def\@glo@range{}%
5533 \expandafter\if\@glo@prefix(\relax
5534   \def\@glo@range{:open-range}%
5535 \else
5536   \expandafter\if\@glo@prefix)\relax
5537   \def\@glo@range{:close-range}%
5538 \fi
5539 \fi

```

Write to the glossary file using xindy syntax.

```

5540 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5541   (indexentry :tkey (\csname glo@\@gls@label @index\endcsname)
5542     :locref \string"{\@glo@counterprefix}{\@glslocref}\string" %
5543     :attr \string"\@gls@counter\@glo@suffix\string"
5544     \@glo@range
5545   )
5546   }%
5547 \else

```

Convert the format information into the format required for makeindex

```

5548 \set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5549 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5550 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5551 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5552   \@gls@encapchar\@glo@numfmt}{\@glslocref}}%

```

```

5553 \fi
5554 }

```

`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.)

```

5555 \newcommand*\@gls@getcounterprefix[2]{%
5556 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5557 \ifx\@gls@thisloc\@gls@thisHloc
5558 \def\@glo@counterprefix{}%
5559 \else
5560 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5561 \def\@glo@tmp{##2}%
5562 \ifx\@glo@tmp\@empty
5563 \def\@glo@counterprefix{}%
5564 \else
5565 \def\@glo@counterprefix{##1}%
5566 \fi
5567 }%
5568 \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5569 \ifx\@glo@counterprefix\@empty
5570 \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5571 prefixing~Jlocation ‘#1’. You need to modify the
5572 definition of \string\theH\@gls@counter~Jotherwise you
5573 will get the warning: “name{\@gls@counter.#1}’ has been~J
5574 referenced but does not exist”}%
5575 \fi
5576 \fi
5577 }

```

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.

```

5578 \newcommand{\@do@seeglossary}[2]{%
5579 \def\@gls@xref{#2}%
5580 \@onelevel@sanitize\@gls@xref
5581 \@gls@checkmkidxchars\@gls@xref
5582 \ifglxindy
5583 \gls@glossary{\csname glo@#1@type\endcsname}{%
5584 (indexentry
5585 :tkey (\csname glo@#1@index\endcsname)
5586 :xref (\string"\@gls@xref\string")}

```

```

5587      :attr \string"see\string"
5588    )
5589  }%
5590 \else
5591   \gls@glossary{\csname glo@#1@type\endcsname}{%
5592   \string\glossaryentry{\csname glo@#1@index\endcsname
5593   \@gls@encapchar glsseeformat\@gls@xref}{Z}}}%
5594 \fi
5595 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5596 \def\@gls@fixbraces#1#2#3\@nil{%
5597   \ifx#2[\relax
5598     \@gls@fixbraces#1#2#3\@end@fixbraces
5599   \else
5600     \def#1{{#2#3}}%
5601   \fi
5602 }

```

`@@gls@fixbraces`

```

5603 \def@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5604   \def#1{[#2]{#3}}%
5605 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

5606 \DeclareRobustCommand*\glssee[3][\seename]{%
5607   \@do@seeglossary{#2}{#1}{#3}}
5608 \newcommand*\@glssee[3][\seename]{%
5609   \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.

```

5610 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5611   \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

5612 \DeclareRobustCommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

5613   \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

5614   \let\@gls@donext\relax

```

Iterate through the labels

```

5615   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5616     \ifx\@xfor@nextelement\@nnil
5617       \@gls@dolast

```

```

5618 \else
5619 \@gls@donext
5620 \fi

```

Display the entry for this label. (Expanding label as it's a temporary control sequence that's used elsewhere.)

```

5621 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%

Update separators
5622 \let\@gls@dolast\glsseelastsep
5623 \let\@gls@donext\glsseesep
5624 }%
5625 }

```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5626 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
5627 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5628 \DeclareRobustCommand*{\glsseeitem}[1]{\gls hyperlink[\glsseeitemformat{#1}]{#1}}
```

`\glsseeitemformat` As from v3.0, default is to use `\glsentrytext` instead of `\glsentryname`. (To avoid problems with the name key being sanitized, although this is no longer a problem now.)

```
5629 \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.

```

5630 \newcommand*{\gls@save@numberlist}[1]{%
5631 \ifglssavenumberlist
5632 \toks@{#1}%
5633 \edef\@do@writeaux@info{%
5634 \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5635 }%
5636 \@onelevel@sanitize\@do@writeaux@info
5637 \protected@write\@auxout{}{\@do@writeaux@info}%
5638 \fi
5639 }

```

`\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.)

```
5640 \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.

```
5641 \ifcsundef{printglossary}{}%
```

```
5642 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5643 \@gls@warnonglossdefined
```

```
5644 \undef\printglossary
```

```
5645 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5646 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
```

```
5647 \@printglossary{#1}{\@print@glossary}%
```

```
5648 }
```

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`

```
5649 \newcommand*{\printglossaries}{%
```

```
5650 \foralllglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
```

```
5651 }
```

`\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.

```
5652 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
```

```
5653 \@printglossary{#1}{\@print@noidx@glossary}%
```

```
5654 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
5655 \newcommand*{\printnoidxglossaries}{%
```

```
5656 \foralllglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
```

```
5657 }
```

`\printgloss@setsort` Initialise to do nothing.

```
5658 \newcommand*{\@printgloss@setsort}{}
```

preglossaryhook

```
5659 \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.

```
5660 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5661 \def\@glo@type{\glsdefaulttype}%
```

```
5662 \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%
```

```
5663 \def\glossarytoctitle{\glossarytitle}%
```

```
5664 \let\org@glossarytitle\glossarytitle
```

```
5665 \def\@glossarystyle{%
```

```
5666 \ifx\@glossary@default@style\relax
```

```
5667 \GlossariesWarning{No default glossary style provided \MessageBreak  
5668 for the glossary ‘\@glo@type’. \MessageBreak
```

```
5669 Using deprecated fallback. \MessageBreak
```

```
5670 To fix this set the style with \MessageBreak
```

```
5671 \string\setglossarystyle\space or use the \MessageBreak
```

```
5672 style key=value option}%
```

```
5673 \fi
```

```
5674 }%
```

```
5675 \def\gls@dotocitle{\glssettocitle{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5676 \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5677 \bgroup
```

Activate or deactivate sort key:

```
5678 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5679 \setkeys{printgloss}{#1}%
```

Does the glossary exist?

```
5680 \ifglossaryexists{\@glo@type}%
```

```
5681 {%
```

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)

```
5682 \ifx\glossarytitle\org@glossarytitle
```

```
5683 \else
```

```
5684 \expandafter\let\csname @glotype@\@glo@type @title\endcsname
```

```
5685 \glossarytitle
```

```
5686 \fi
```

Allow a high-level user command to indicate the current glossary

```
5687 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5688 \let\org@glossaryentrynumbers@glossaryentrynumbers
```

```
5689 \let\glsnonextpages@glsnonextpages
```

Enable individual number list to be activated:

```
5690 \let\glsnextpages@glsnextpages
```

Enable suppression of description terminators.

```
5691 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5692 \gls@dotoc@title
```

Set the glossary style

```
5693 \@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):

```
5694 \let\gls@org@glossaryentryfield@glossentry
5695 \let\gls@org@glossarysubentryfield@subglossentry
5696 \renewcommand{\glossentry}[1]{%
5697   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5698   \gls@org@glossaryentryfield{##1}%
5699 }%
5700 \renewcommand{\subglossentry}[2]{%
5701   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5702   \gls@org@glossarysubentryfield{##1}{##2}%
5703 }%
```

```
5704 \@gls@preglossaryhook
```

Now do the handler macro that deals with the actual glossary:

```
5705 #2%
5706 }%
5707 {\GlossariesWarning{Glossary ‘\@glo@type’ doesn’t exist}}%
```

End the current scope

```
5708 \egroup
```

Reset `\glossaryentrynumbers`

```
5709 \global\let@glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5710 \global\let\warn@noprintglossary\relax
5711 }
```

`@print@glossary` Internal workings of `\printglossary` dealing with reading the external file.

```
5712 \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.)

5713 \makeatletter

Input the glossary file, if it exists.

5714 \@input@{\jobname.\csname @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.

5715 \IfFileExists{\jobname.\csname @glo@type @in\endcsname}%

5716 {}%

5717 {\null}%

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

5718 \ifglxindy

5719 \ifcsundef{@xdy@\glo@type @language}%

5720 {%

5721 \edef\@do@auxoutstuff{%

5722 \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.

5723 \noexpand\immediate\noexpand\write\@auxout{%

5724 \string\providecommand\string\@xdy@language[2]{}}%

5725 \noexpand\immediate\noexpand\write\@auxout{%

5726 \string\@xdy@language{\@glo@type}{\@xdy@main@language}}%

5727 }%

5728 }%

5729 }%

5730 {%

5731 \edef\@do@auxoutstuff{%

5732 \noexpand\AtEndDocument{%

5733 \noexpand\immediate\noexpand\write\@auxout{%

5734 \string\providecommand\string\@xdy@language[2]{}}%

5735 \noexpand\immediate\noexpand\write\@auxout{%

5736 \string\@xdy@language{\@glo@type}{\csname @xdy@\glo@type
@language\endcsname}}%

5737 }%

5738 }%

5739 }%

5740 }%

5741 \@do@auxoutstuff

5742 \edef\@do@auxoutstuff{%

5743 \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.

5744 \noexpand\immediate\noexpand\write\@auxout{%

5745 \string\providecommand\string\@gls@codepage[2]{}}%

```

5746         \noexpand\immediate\noexpand\write\@auxout{%
5747         \string\@gls@codepage{\@glo@type}{\@gls@codepage}}}%
5748     }%
5749 }%
5750 \do@auxoutstuff
5751 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

5752 \renewcommand*{\@warn@nomakeglossaries}{%
5753 \GlossariesWarningNoLine{\string\makeglossaries\space
5754 hasn't been used,^^Jthe glossaries will not be updated}}%
5755 }%
5756 }

```

The sort macros all have the syntax:

```
\@glo@sortmacro@<order>{<type>}
```

where <order> is the sort order as specified by the sort key and <type> is the glossary type. (The referenced entry list is stored in \@glsref@<type>. The actual sorting is done by \@glo@sortentries{<handler>}{<type>}.

glo@sortentries

```

5757 \newcommand*{\@glo@sortentries}[2]{%
5758 \glosortentrieswarning
5759 \def\@glo@sortinglist{}%
5760 \def\@glo@sortinghandler{#1}%
5761 \edef\@glo@type{#2}%
5762 \forlistcsloop{\@glo@do@sortentries}{@glsref@#2}%
5763 \csdef{@glsref@#2}{}%
5764 \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5765 \xifinlistcs{\@this@label}{@glsref@#2}%
5766 {}%
5767 {%
5768 \listcsxadd{@glsref@#2}{\@this@label}%
5769 }%
5770 \ifcsdef{@glo@sortingchildren@ \@this@label}%
5771 {%
5772 \@glo@addchildren{#2}{\@this@label}%
5773 }%
5774 {}%
5775 }%
5776 }

```

@glo@addchildren

```
\@glo@addchildren{<type>}{<parent>}
```

```

5777 \newcommand*{\@glo@addchildren}[2]{%

```

Scope to allow nesting.

```
5778 \bgroup
5779 \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
5780 \for\@this@childlabel:=\@glo@childlist\do
5781 {%
```

Check this label hasn't already been added.

```
5782 \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
5783 {%
5784 {%
5785 \listcsxadd{\@glsref@#1}{\@this@childlabel}%
5786 }%
```

Does this child have children?

```
5787 \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
5788 {%
5789 \@glo@addchildren{#1}{\@this@childlabel}%
5790 }%
5791 {%
5792 }%
5793 }%
5794 \egroup
5795 }
```

@do@sortentries

```
5796 \newcommand*{\@glo@do@sortentries}[1]{%
5797 \ifglshasparent{#1}%
5798 {%
```

This entry has a parent, so add it to the child list

```
5799 \edef\@glo@parent{\csuse{glo@glsdetoklabel{#1}@parent}}%
5800 \ifcsundef{\@glo@sortingchildren@\@glo@parent}%
5801 {%
5802 \csdef{\@glo@sortingchildren@\@glo@parent}{}%
5803 }%
5804 {}%
5805 \expandafter\@glo@sortedinsert
5806 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
5807 \xifinlistcs{\@glo@parent}{\@glsref@\@glo@type}%
5808 {%
```

Yes, it has so do nothing.

```
5809 }%
5810 {%
```

No, it hasn't so add it now.

```
5811 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5812 }%
5813 }%
```

```

5814  {%
5815    \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5816  }%
5817 }

```

`\glo@sortedinsert` `\@glo@sortedinsert{<list>}{<entry label>}`

Insert into list.

```

5818 \newcommand*{\@glo@sortedinsert}[2]{%
5819   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5820 }%

```

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`

```

5821 \newcommand*{\@glo@sorthandler@word}[2]{%
5822   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5823   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5824   \edef\glo@do@compare{%
5825     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5826     {\expandonce\@gls@sort@B}%
5827     {\expandonce\@gls@sort@A}%
5828   }%
5829   \glo@do@compare
5830 }

```

`\sorthandler@letter`

```

5831 \newcommand*{\@glo@sorthandler@letter}[2]{%
5832   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5833   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5834   \edef\glo@do@compare{%
5835     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5836     {\expandonce\@gls@sort@B}%
5837     {\expandonce\@gls@sort@A}%
5838   }%
5839   \glo@do@compare
5840 }

```

`\sorthandler@case` Case-sensitive sort.

```

5841 \newcommand*{\@glo@sorthandler@case}[2]{%
5842   \letcs\@gls@sort@A{glo@glstdetoklabel{#1}@sort}%
5843   \letcs\@gls@sort@B{glo@glstdetoklabel{#2}@sort}%
5844   \edef\glo@do@compare{%
5845     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5846     {\expandonce\@gls@sort@B}%

```

```

5847     {\expandonce\@gls@sort@A}%
5848 }%
5849 \glo@do@compare
5850 }

```

thandler@nocase Case-insensitive sort.

```

5851 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5852   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5853   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5854   \edef\glo@do@compare{%
5855     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5856     {\expandonce\@gls@sort@B}%
5857     {\expandonce\@gls@sort@A}%
5858   }%
5859   \glo@do@compare
5860 }

```

@sortmacro@word Sort macro for ‘word’

```

5861 \newcommand*{\@glo@sortmacro@word}[1]{%
5862   \ifdefstring{\@glo@default@sorttype}{standard}%
5863   {%
5864     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5865   }%
5866   {%
5867     \PackageError{glossaries}{Conflicting sort options:^^J
5868       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5869       \string\printnoidxglossary[sort=word]}{ }%
5870   }%
5871 }

```

ortmacro@letter Sort macro for ‘letter’

```

5872 \newcommand*{\@glo@sortmacro@letter}[1]{%
5873   \ifdefstring{\@glo@default@sorttype}{standard}%
5874   {%
5875     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5876   }%
5877   {%
5878     \PackageError{glossaries}{Conflicting sort options:^^J
5879       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5880       \string\printnoidxglossary[sort=letter]}{ }%
5881   }%
5882 }

```

tmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5883 \newcommand*{\@glo@sortmacro@standard}[1]{%
5884   \ifdefstring{\@glo@default@sorttype}{standard}%
5885   {%
5886     \ifcsdef{\@glo@sorthandler@\glsorder}%
5887     {%

```

```

5888     \@glo@sortentries{\csuse{@glo@sorthandler@\glsorder}}{#1}%
5889 }%
5890 {%
5891     \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5892 }%
5893 }%
5894 {%
5895     \PackageError{glossaries}{Conflicting sort options:^^J
5896     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5897     \string\printnoidxglossary[sort=standard]}{}%
5898 }%
5899 }

```

@sortmacro@case Sort macro for ‘case’

```

5900 \newcommand*{\@glo@sortmacro@case}[1]{%
5901   \ifdefstring{\@glo@default@sorttype}{standard}%
5902   {%
5903     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5904   }%
5905   {%
5906     \PackageError{glossaries}{Conflicting sort options:^^J
5907     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5908     \string\printnoidxglossary[sort=case]}{}%
5909   }%
5910 }

```

ortmacro@nocase Sort macro for ‘nocase’

```

5911 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5912   \ifdefstring{\@glo@default@sorttype}{standard}%
5913   {%
5914     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5915   }%
5916   {%
5917     \PackageError{glossaries}{Conflicting sort options:^^J
5918     \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5919     \string\printnoidxglossary[sort=nocase]}{}%
5920   }%
5921 }

```

o@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@<type>.

```

5922 \newcommand*{\@glo@sortmacro@def}[1]{%
5923   \def\@glo@sortinglist{}%
5924   \for@gl@sentries[#1]{\@gls@thislabel}%
5925   {%
5926     \xifinlistcs{\@gls@thislabel}{\@gls@ref@#1}%
5927     {%
5928       \listead{\@glo@sortinglist}{\@gls@thislabel}%
5929     }%
5930   }%

```

Hasn't been referenced.

```
5931 }%
5932 }%
5933 \cslet{@glsref@#1}{\@glo@sortinglist}%
5934 }
```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5935 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5936 \ifinlistcs{#1}{@glsref@\@glo@type}%
5937 }%
5938 {%
5939 \listcsadd{@glsref@\@glo@type}{#1}%
5940 }%
5941 \ifcsdef{@glo@sortingchildren@#1}%
5942 {%
5943 \@glo@addchildren{\@glo@type}{#1}%
5944 }%
5945 }%
5946 }
```

o@sortmacro@use Sort macro for 'use'. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
5947 \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.

```
5948 \newcommand*{\@print@noidx@glossary}{%
5949 \ifcsdef{@glsref@\@glo@type}%
5950 {%
```

Sort the entries:

```
5951 \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
5952 {%
5953 \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5954 }%
5955 {%
5956 \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{ }%
5957 }
```

Do the glossary heading and preamble

```
5958 \glossarysection[\glossarytoctitle]{\glossarytitle}%
5959 \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.

```
5960 \def\@gls@currentlettergroup{}%
5961 \begin{theglossary}%
5962 \glossaryheader
5963 \glsresetentrylist
```

Iterate through the entries.

```
5964 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}}%
```

Finally end the glossary and do the postamble:

```
5965 \end{theglossary}%
5966 \glossarypostamble
5967 }%
5968 {%
5969 \@gls@noref@warn{\@glo@type}%
5970 }%
5971 }
```

\glo@grabfirst

```
5972 \def\glo@grabfirst#1#2\@nil{%
5973 \def\@gls@firsttok{#1}%
5974 \ifdefempty\@gls@firsttok
5975 {%
5976 \def\@glo@thislettergrp{0}%
5977 }%
5978 {%
```

Sanitize it:

```
5979 \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
5980 \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
5981 }%
5982 }
```

\@glo@grabfirst

```
5983 \def\@glo@grabfirst#1#2\@nil{%
5984 \ifdefempty\@glo@thislettergrp
5985 {%
5986 \def\@glo@thislettergrp{glssymbols}%
5987 }%
5988 {%
5989 \count@=\uccode'#1\relax
5990 \ifnum\count@=0\relax
5991 \def\@glo@thislettergrp{glssymbols}%
5992 \else
5993 \ifdefstring\@glo@sorttype{case}%
5994 {%
5995 \count@='#1\relax
```



```

5996     }%
5997     {%
5998     }%
5999     \edef\@glo@thislettergrp{\the\count@}%
6000     \fi
6001     }%
6002 }

```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label.
 This only allows one sublevel.

```

6003 \newcommand{\@gls@noidx@do}[1]{%

```

 Get this entry's location list

```

6004    \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%

```

 Does this entry have a parent?

```

6005    \ifglshasparent{#1}%
6006    {%

```

 Has a parent.

```

6007      \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
6008      \ifdefvoid{\@gls@loclist}
6009      {%
6010        \subglossentry{\gls@level}{#1}{}%
6011      }%
6012      {%
6013        \subglossentry{\gls@level}{#1}%
6014        {%
6015          \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
6016        }%
6017      }%
6018    }%
6019    {%

```

 Doesn't have a parent Get this entry's sort key

```

6020    \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%

```

 Fetch the first letter:

```

6021    \expandafter\glo@grabfirst\@gls@sort{}\@nil
6022    \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
6023    {%
6024    {%

```

 Do the group header:

```

6025      \ifdefempty{\@gls@currentlettergroup}{}%
6026      {%

```

 The group skip may start a new scope, so make a global assignment.

```

6027        \global\let\@glo@thislettergrp\@glo@thislettergrp
6028        \glsgroupskip
6029      }%
6030      \glsgroupheading{\@glo@thislettergrp}%
6031    }%

```

```
6032 \global\let\@gls@currentlettergroup\@glo@thislettergrp
```

Do this entry:

```
6033 \ifdefvoid{\@gls@loclist}
6034 {%
6035   \glossentry{#1}{}%
6036 }%
6037 {%
6038   \glossentry{#1}%
6039   {%
6040     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
6041   }%
6042 }%
6043 }%
6044 }
```

```
\glsnoidxloclist \glsnoidxloclist{<list cs>}
```

Display location list.

```
6045 \newcommand*{\glsnoidxloclist}[1]{%
6046   \def\@gls@noidxloclist@sep{}%
6047   \def\@gls@noidxloclist@prev{}%
6048   \forlistloop{\glsnoidxloclisthandler}{#1}%
6049 }
```

`xloclisthandler` Handler for location list iterator.

```
6050 \newcommand*{\glsnoidxloclisthandler}[1]{%
6051   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6052   {%
6053     }%
6054     {%
6055       \@gls@noidxloclist@sep
6056       #1%
6057       \def\@gls@noidxloclist@sep{\delimN}%
6058       \def\@gls@noidxloclist@prev{#1}%
6059     }%
6060 }
```

`yloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```
6061 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
6062   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
6063   {%
6064     }%
6065     {%
6066       \def\@gls@noidxdisplayloclist@sep{\delimN}%
6067       \def\@gls@noidxdisplayloclist@prev{#1}%
6068     }%
6069 }
```

```

6066 \gls@noidxloclist@sep
6067 \gls@noidxloclist@prev
6068 \def\gls@noidxloclist@prev{#1}%
6069 }%
6070 }

```

\glsnoidxdisplayloc

```
\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}
```

Display a location in the location list.

```

6071 \newcommand*\glsnoidxdisplayloc[4]{%
6072 \setentrycounter[#1]{#2}%
6073 \csuse{#3}{#4}%
6074 }

```

\gls@reference

```
\gls@reference{<type>}{<label>}{<loc>}
```

Identifies that a reference has been used (for use in the aux file). All entries must be defined in the preamble.

```
6075 \newcommand*\gls@reference}[3]{%
```

Add to label list

```

6076 \glsdoifexistsorwarn{#2}%
6077 {%
6078 \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}}{}%
6079 \ifinlistcs{#2}{@glsref@#1}%
6080 }%
6081 {\listcsgadd{@glsref@#1}{#2}}%

```

Add to location list

```

6082 \ifcsundef{glo@glstdetoklabel{#2}@loclist}%
6083 {\csgdef{glo@glstdetoklabel{#2}@loclist}{}}%
6084 }%
6085 \listcsgadd{glo@glstdetoklabel{#2}@loclist}{#3}%
6086 }%
6087 }

```

The keys that can be used in the optional argument to \printglossary or \printnoidxglossary are as follows: The type key sets the glossary type.

```
6088 \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.

```

6089 \define@key{printgloss}{title}{%
6090 \def\glossarytitle{#1}%
6091 \let\gls@dotocitle\relax
6092 }

```

The toctitle sets the text used for the relevant entry in the table of contents.

```
6093 \define@key{printgloss}{toctitle}{%
6094   \def\glossarytoctitle{#1}%
6095   \let\gls@dotoc\toc\relax
6096 }
```

The style key sets the glossary style (but only for the given glossary).

```
6097 \define@key{printgloss}{style}{%
6098   \ifcsundef{@glsstyle@#1}%
6099   {%
6100     \PackageError{glossaries}%
6101     {Glossary style ‘#1’ undefined}{}%
6102   }%
6103   {%
6104     \def\@glossarystyle{\setglossentrycompatibility
6105       \csname @glsstyle@#1\endcsname}%
6106   }%
6107 }
```

The numberedsection key determines if this glossary should be in a numbered section.

```
6108 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
6109   false,nolabel,autolabel,nameref}[nolabel]{%
6110     \ifcase\nr\relax
6111       \renewcommand*{\@glossarysecstar}{*}%
6112       \renewcommand*{\@glossaryseclabel}{}%
6113     \or
6114       \renewcommand*{\@glossarysecstar}{}%
6115       \renewcommand*{\@glossaryseclabel}{}%
6116     \or
6117       \renewcommand*{\@glossarysecstar}{}%
6118       \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
6119     \or
6120       \renewcommand*{\@glossarysecstar}{*}%
6121       \renewcommand*{\@glossaryseclabel}{%
6122         \protected@edef\@currentlabelname{\glossarytoctitle}%
6123         \label{\glsautoprefix\@glo@type}}%
6124     \fi
6125 }
```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.

```
6126 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
6127   \csuse{glsnogroupskip#1}%
6128 }
```

The nopostdot key has the same effect as the package option of the same name.

```
6129 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
6130   \csuse{glsnopostdot#1}%
6131 }
```

nterLabelPrefix Make it easier to redefine the label prefix.

```
6132 \newcommand*{\GlsEntryCounterLabelPrefix}{glentry-}
```

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.

```
6133 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
6134   \csuse{glentrycounter#1}%
6135   \@gls@define@glossaryentrycounter
6136 }
```

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.

```
6137 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6138   \csuse{glssubentrycounter#1}%
6139   \@gls@define@glossarysubentrycounter
6140 }
```

The nonumberlist key determines if this glossary should have a number list.

```
6141 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
6142   \ifglslnonumberlist
6143   \def\glossaryentrynumbers##1{%
6144     \else
6145     \def\glossaryentrynumbers##1{##1}%
6146   \fi}
```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```
6147 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}
```

@assign@sortkey Issue error if used with \printglossary

```
6148 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6149   \PackageError{glossaries}{‘sort’ key not permitted with
6150   \string\printglossary}%
6151   {The ‘sort’ key may only be used with \string\printnoidxglossary}%
6152 }
```

@assign@sortkey For use with \printnoidxglossary

```
6153 \newcommand*{\@glo@assign@sortkey}[1]{%
6154   \def\@glo@sorttype{#1}%
6155 }
```

@glslnonextpages 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 \glslnonextpages is place in the entry's description and 3 column tabular style glossary is used.) \org@glossaryentrynumbers needs to be set at the start of each glossary, in the event that \glossaryentrynumber is re-defined.

```

6156 \newcommand*{\@glsnonextpages}{%
6157   \gdef\glossaryentrynumbers##1{%
6158     \glsresetentrylist
6159   }%
6160 }

```

`\@glsnextpages` Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6161 \newcommand*{\@glsnextpages}{%
6162   \gdef\glossaryentrynumbers##1{%
6163     ##1\glsresetentrylist}}

```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```

6164 \newcommand*{\glsresetentrylist}{%
6165   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```

6166 \newcommand*{\glsnonextpages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

6167 \newcommand*{\glsnextpages}{}

```

Process `entrycounter` and then `subentrycounter` options (this ensures the sub-counter can pick up the main counter as the master if required):

```

6168 \@gls@define@glossaryentrycounter
6169 \@gls@define@glossarysubentrycounter

```

`\glsresetsubentrycounter` Resets the `glossarysubentry` counter.

```

6170 \newcommand*{\glsresetsubentrycounter}{%
6171   \ifglssubentrycounter
6172     \setcounter{glossarysubentry}{0}%
6173   \fi
6174 }

```

`\glsresetentrycounter` Resets the `glossaryentry` counter.

```

6175 \newcommand*{\glsresetentrycounter}{%
6176   \ifglsentrycounter
6177     \setcounter{glossaryentry}{0}%
6178   \fi
6179 }

```

`\glsstepentry` Advance the `glossaryentry` counter if in use. The argument is the label associated with the entry.

```

6180 \newcommand*{\glsstepentry}[1]{%

```

```

6181 \ifglentrycounter
6182   \refstepcounter{glossaryentry}%
6183   \label{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6184 \fi
6185 }

```

glsstepsubentry Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```

6186 \newcommand*{\glsstepsubentry}[1]{%
6187   \ifglssubentrycounter
6188     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6189     \refstepcounter{glossarysubentry}%
6190     \label{\GlsEntryCounterLabelPrefix\currentglssubentry}%
6191   \fi
6192 }

```

\glsrefentry Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

6193 \newcommand*{\glsrefentry}[1]{%
6194   \ifglentrycounter
6195     \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6196   \else
6197     \ifglssubentrycounter
6198       \ref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
6199     \else
6200       \gls{#1}%
6201     \fi
6202   \fi
6203 }

```

trycounterlabel Defines how to display the glossaryentry counter.

```

6204 \newcommand*{\glentrycounterlabel}{%
6205   \ifglentrycounter
6206     \theglossaryentry.\space
6207   \fi
6208 }

```

trycounterlabel Defines how to display the glossarysubentry counter.

```

6209 \newcommand*{\glssubentrycounterlabel}{%
6210   \ifglssubentrycounter
6211     \theglossarysubentry)\space
6212   \fi
6213 }

```

\glentryitem Step and display glossaryentry counter, if appropriate.

```

6214 \newcommand*{\glentryitem}[1]{%
6215   \ifglentrycounter
6216     \glsstepentry{#1}\glentrycounterlabel
6217   \else

```

```

6218 \glsresetsubentrycounter
6219 \fi
6220 }

```

`glsesubentryitem` Step and display glossarysubentry counter, if appropriate.

```

6221 \newcommand*{\glsesubentryitem}[1]{%
6222 \ifglsesubentrycounter
6223 \glssubentry{#1}\glsesubentrycounterlabel
6224 \fi
6225 }

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

6226 \ifcsundef{theglossary}%
6227 {%
6228 \newenvironment{theglossary}{}{}%
6229 }%
6230 {%
6231 \@gls@warnontheglossdefined
6232 \renewenvironment{theglossary}{}{}%
6233 }

```

The glossary header is given by `\glossaryheader`. This forms part of the glossary style, and must indicate what should appear immediately after the start of the `theglossary` environment. (For example, if the glossary uses a tabular-like environment, it may be used to set the header row.) Note that if you don't want a header row, the glossary style must redefine `\glossaryheader` to do nothing.

`\glossaryheader`

```

6234 \newcommand*{\glossaryheader}{}

```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

6235 \newcommand*{\glstarget}[2]{\@glstarget{\gloslinkprefix#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>}
```

```

6236 \providecommand*{\compatibleglossentry}[2]{%
6237 \toks@{#2}%
6238 \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%

```



```

6239     {\noexpand\glsnamefont
6240       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6241     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6242     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6243     {\the\toks@}%
6244   }%
6245   \@do@glossentry
6246 }

```

\glossentryname

```

6247 \newcommand*{\glossentryname}[1]{%
6248   \glsdoifexistsorwarn{#1}%
6249   {%
6250     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6251     \expandafter\glsnamefont\expandafter{\glo@name}%
6252   }%
6253 }

```

\Glossentryname

```

6254 \newcommand*{\Glossentryname}[1]{%
6255   \glsdoifexistsorwarn{#1}%
6256   {%
6257     \glsnamefont{\Glsentryname{#1}}%
6258   }%
6259 }

```

\glossentrydesc

```

6260 \newcommand*{\glossentrydesc}[1]{%
6261   \glsdoifexistsorwarn{#1}%
6262   {%
6263     \glentrydesc{#1}%
6264   }%
6265 }

```

\Glossentrydesc

```

6266 \newcommand*{\Glossentrydesc}[1]{%
6267   \glsdoifexistsorwarn{#1}%
6268   {%
6269     \Glsentrydesc{#1}%
6270   }%
6271 }

```

\glossentrysymbol

```

6272 \newcommand*{\glossentrysymbol}[1]{%
6273   \glsdoifexistsorwarn{#1}%
6274   {%
6275     \glentrysymbol{#1}%
6276   }%
6277 }

```

lossentrysymbol

```
6278 \newcommand*{\Glossentrysymbol}[1]{%
6279   \glsdoifexistsorwarn{#1}%
6280   {%
6281     \Glsentrysymbol{#1}%
6282   }%
6283 }
```

blesubglossentry

`\subglossentry{<level>}{<label>}{<page-list>}`

```
6284 \providecommand*{\compatiblesubglossentry}[3]{%
6285   \toks@{#3}%
6286   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6287     {#2}%
6288     {\noexpand\glsnamefont
6289       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6290   {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6291   {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6292   {\the\toks@}%
6293 }%
6294 \@do@subglossentry
6295 }
```

rycompatibility

```
6296 \newcommand*{\setglossentrycompatibility}{%
6297   \let\glossentry\compatibleglossentry
6298   \let\subglossentry\compatiblesubglossentry
6299 }
6300 \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.

```
6301 \newcommand{\glossaryentryfield}[5]{%
6302   \GlossariesWarning
6303   {Deprecated use of \string\glossaryentryfield.^^J
6304     I recommend you change to \string\glossentry.^^J
6305     If you've just upgraded, try removing your gls auxiliary
6306     files^^J and recompile}%
6307   \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.)

```
6308 \newcommand*{\glossarysubentryfield}[6]{%
6309   \GlossariesWarning
6310   {Deprecated use of \string\glossarysubentryfield.^^J
6311    I recommend you change to \string\subglossentry.^^J
6312    If you've just upgraded, try removing your gls auxiliary
6313    files^^J and recompile}%
6314   \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`

```
6315 \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`

```
6316 \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.

`\sgetgrouptitle`

```
6317 \newcommand*{\glsgetgrouptitle}[1]{%
6318   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6319   \@gls@grptitle
6320 }
```

`\s@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
6321 \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.

```
6322 \dtl@ifsingle{#1}%
6323 {%
6324   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6325 }%
6326 {%
6327   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6328               or test{\ifstrequal{#1}{glsnumbers}}}%
6329   {%
6330     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6331   }%
6332   {%
6333     \def#2{#1}%
6334   }%
6335 }%
6336 }
```

`\x@getgrouptitle` Version for the no-indexing app option:

```
6337 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6338   \DTLifint{#1}%
6339   {\edef#2{\char#1\relax}}%
6340   {%
6341     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6342   }%
6343 }
```

`\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`.

`\sgetgrouplabel`

```
6344 \newcommand*{\glsgetgrouplabel}[1]{%
```

```

6345 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6346 \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`

```

6347 \newcommand*{\setentrycounter}[2][]{%
6348   \def\@glo@counterprefix{#1}%
6349   \ifx\@glo@counterprefix\empty
6350     \def\@glo@counterprefix{.}%
6351   \else
6352     \def\@glo@counterprefix{.#1.}%
6353   \fi
6354   \def\glsentrycounter{#2}%
6355 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

6356 \newcommand*{\setglossarystyle}[1]{%
6357   \ifcsundef{@glsstyle@#1}%
6358   {%
6359     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6360   }%
6361   {%
6362     \csname @glsstyle@#1\endcsname
6363   }%

```

Set the default style if it's not already set.

```

6364   \ifx\@glossary@default@style\relax
6365     \protected@edef\@glossary@default@style{#1}%
6366   \fi
6367 }

```

`\glossarystyle`

```

6368 \newcommand*{\glossarystyle}[1]{%
6369   \ifcsundef{@glsstyle@#1}%
6370   {%
6371     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6372   }%
6373   {%
6374     \GlossariesWarning
6375     {Deprecated command \string\glossarystyle.^~J
6376     I recommend you switch to \string\setglossarystyle\space unless
6377     you want to maintain backward compatibility}%
6378     \setglossentrycompatibility
6379     \csname @glsstyle@#1\endcsname

```

```

6380 \ifcsdef{@glscompstyle@#1}%
6381 {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6382 {}%
6383 }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6384 \ifx\@glossary@default@style\relax
6385 \protected@edef\@glossary@default@style{#1}%
6386 \fi
6387 }

```

`\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.

```

6388 \newcommand{\newglossarystyle}[2]{%
6389 \ifcsundef{@glsstyle@#1}%
6390 {%
6391 \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6392 }%
6393 {%
6394 \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6395 }%
6396 }

```

`\newglossarystyle` Code for this macro supplied by Marco Daniel.

```

6397 \newcommand{\renewglossarystyle}[2]{%
6398 \ifcsundef{@glsstyle@#1}%
6399 {%
6400 \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6401 }%
6402 {%
6403 \csdef{@glsstyle@#1}{#2}%
6404 }%
6405 }

```

Glossary entries are encoded so that the second argument to `\glossaryentryfield` is always specified as `\glsnamefont{<name>}`. This allows the user to change the font used to display the name term without having to redefine `\glossaryentryfield`. The default uses the surrounding font, so in the list type styles (which place the name in the optional argument to `\item`) the name will appear in bold.

`\glsnamefont`

```
6406 \newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the format key in commands like `\glslink`. The default format is given by `\glshypernumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn't have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the package.

`\glshypernumber`

```
6407 \ifcsundef{hyperlink}%
6408 {%
6409   \def\glshypernumber#1{#1}%
6410 }%
6411 {%
6412   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6413 }
```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6414 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6415   \ifx\#1\%
6416     \else
6417       \@delimR#1\delimR\delimR\%
6418     \fi
6419   \ifx\#2\%
6420     \else
6421       #2%
6422     \fi
6423   \ifx\#3\%
6424     \else
6425       \@glshypernumber#3\@nil
6426     \fi
6427 }
```

`\@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`

```
6428 \def\@delimR#1\delimR #2\delimR #3\%
6429 \ifx\#2\%
6430   \@delimN{#1}%
6431 \else
6432   #2\delimR #3\%
6433 \fi
```

```

6431 \else
6432   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6433 \fi}

```

\@delimN displays a list of individual numbers, instead of a range:

\@delimN

```

6434 \def\@delimN#1{\@delimN#1\delimN \delimN\}
6435 \def\@delimN#1\delimN #2\delimN#3\{\{%
6436 \ifx\#3\}%
6437   \@gls@numberlink{#1}%
6438 \else
6439   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6440 \fi
6441 }

```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

6442 \def\@gls@numberlink#1{%
6443 \begingroup
6444 \toks@={}%
6445 \@gls@removespaces#1 \@nil
6446 \endgroup}

6447 \def\@gls@removespaces#1 #2\@nil{%
6448 \toks@=\expandafter{\the\toks@#1}%
6449 \ifx\#2\}%
6450 \edef\x{\the\toks@}%
6451 \ifx\x\empty
6452 \else

6453   \hyperlink{\glstrycounter\@glo@counterprefix\the\toks@}%
6454   {\the\toks@}%
6455 \fi
6456 \else
6457   \@gls@ReturnAfterFi{%
6458     \@gls@removespaces#2\@nil
6459   }%
6460 \fi
6461 }
6462 \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

```

6463 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}

```

\hypersf

```

6464 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}

```



```

\hypertt
6465 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
6466 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}

\hypermd
6467 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}

\hyperit
6468 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
6469 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
6470 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
6471 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
6472 \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.

```

6473 \newcommand{\oldacronym}[4][\gls@label]{%
6474   \def\gls@label{#2}%
6475   \newacronym[#4]{#1}{#2}{#3}%
6476   \ifcsundef{xspace}%

```

```

6477 {%
6478   \expandafter\edef\csname#1\endcsname{%
6479     \noexpand@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6480   }%
6481 }%
6482 {%
6483   \expandafter\edef\csname#1\endcsname{%
6484     \noexpand@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6485       \noexpand\gls{#1}\noexpand\xspace}%
6486     }%
6487   }%
6488 }

```

`\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`

```

6489 \newcommand{\newacronym}[4] [] {}

```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix` Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, `ABCS` looks as though the “s” is part of the acronym, but `ABCS` looks as though the “s” is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is need to cancel it out.

```

6490 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}

```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```

6491 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}

```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```

6492 \newcommand*{\glsshortkey}{short}

```

`\sshortpluralkey`

```

6493 \newcommand*{\glsshortpluralkey}{shortplural}

```

```

\glslongkey
6494 \newcommand*{\glslongkey}{long}

lslongpluralkey
6495 \newcommand*{\glslongpluralkey}{longplural}

\acrfull  Full form of the acronym.
6496 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\@ns@acrfull}

6497 \newcommand*\@ns@acrfull[2][{}]{%
6498   \new@ifnextchar[{\@acrfull{#1}{#2}}{%
6499     {\@acrfull{#1}{#2}[]}%
6500 }

\@acrfull  Low-level macro:
6501 \def\@acrfull#1#2[#3]{%
  Make it easier for acronym styles to change this:
6502   \acrfullfmt{#1}{#2}{#3}%
6503 }

  Using \acrlinkfullformat and \acrfullformat is now deprecated as it can cause com-
  plications with the first letter upper case variants, but the package needs to provide backward
  compatibility support.

\acrfullfmt  No case change full format.
6504 \newcommand*{\acrfullfmt}[3]{%
6505   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6506 }

acrlinkfullformat  Format for full links like \acrfull. Syntax: \acrlinkfullformat{<long cs>}{<short cs>}{
  <options>}{<label>}{<insert>}
6507 \newcommand{\acrlinkfullformat}[5]{%
6508   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[]}%
6509 }

\acrfullformat  Default full form is <long> (<short>).
6510 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}

\glsspace  Robust space to ensure it's written to the .glsdefs file.
6511 \newrobustcmd{\glsspace}{\space}

  Default format for full acronym

\Acrfull
6512 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\@ns@Acrfull}

```

```

6513 \newcommand*\ns@Acrfull[2] [] {%
6514   \new@ifnextchar[{\@Acrfull{#1}{#2}}}%
6515   {\@Acrfull{#1}{#2} []}%
6516 }

```

Low-level macro:

```

6517 \def\@Acrfull#1#2[#3] {%

```

Make it easier for acronym styles to change this:

```

6518   \Acrfullfmt{#1}{#2}{#3}%
6519 }

```

`\Acrfullfmt` First letter upper case full format.

```

6520 \newcommand*\@Acrfullfmt[3] {%
6521   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6522 }

```

`\ACRfull`

```

6523 \newrobustcmd*\@ACRfull{\@gls@hyp@opt\ns@ACRfull}

```

```

6524 \newcommand*\ns@ACRfull[2] [] {%
6525   \new@ifnextchar[{\@ACRfull{#1}{#2}}}%
6526   {\@ACRfull{#1}{#2} []}%
6527 }

```

Low-level macro:

```

6528 \def\@ACRfull#1#2[#3] {%

```

Make it easier for acronym styles to change this:

```

6529   \ACRfullfmt{#1}{#2}{#3}%
6530 }

```

`\ACRfullfmt` All upper case full format.

```

6531 \newcommand*\@ACRfullfmt[3] {%
6532   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
6533 }

```

Plural:

`\acrfullpl`

```

6534 \newrobustcmd*\@acrfullpl{\@gls@hyp@opt\ns@acrfullpl}

```

```

6535 \newcommand*\ns@acrfullpl[2] [] {%
6536   \new@ifnextchar[{\@acrfullpl{#1}{#2}}}%
6537   {\@acrfullpl{#1}{#2} []}%
6538 }

```

Low-level macro:

```

6539 \def\@acrfullpl#1#2[#3] {%

```

Make it easier for acronym styles to change this:

```
6540 \acrfullplfmt{#1}{#2}{#3}%  
6541 }
```

\acrfullplfmt No case change plural full format.

```
6542 \newcommand*{\acrfullplfmt}[3]{%  
6543 \acrlinkfullformat{\@acrlongpl}{\acrshortpl}{#1}{#2}{#3}%  
6544 }
```

\Acrfullpl

```
6545 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}  
  
6546 \newcommand*\ns@Acrfullpl[2][ ]{%  
6547 \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6548 {\@Acrfullpl{#1}{#2}[ ]}%  
6549 }
```

Low-level macro:

```
6550 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6551 \Acrfullplfmt{#1}{#2}{#3}%  
6552 }
```

\Acrfullplfmt First letter upper case plural full format.

```
6553 \newcommand*{\Acrfullplfmt}[3]{%  
6554 \acrlinkfullformat{\@Acrlongpl}{\acrshortpl}{#1}{#2}{#3}%  
6555 }
```

\ACRfullpl

```
6556 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}  
  
6557 \newcommand*\ns@ACRfullpl[2][ ]{%  
6558 \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%  
6559 {\@ACRfullpl{#1}{#2}[ ]}%  
6560 }
```

Low-level macro:

```
6561 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6562 \ACRfullplfmt{#1}{#2}{#3}%  
6563 }
```

\ACRfullplfmt All upper case plural full format.

```
6564 \newcommand*{\ACRfullplfmt}[3]{%  
6565 \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6566 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:
6567 `\newcommand{\acronymfont}[1]{#1}`

`\firstacronymfont` This is only used with the additional acronym styles:
6568 `\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.
6569 `\newcommand*{\acrnameformat}[2]{\acronymfont{#1}}`

Define some tokens used by `\newacronym`:

`\glskeylisttok`
6570 `\newtoks\glskeylisttok`

`\glslabeltok`
6571 `\newtoks\glslabeltok`

`\glsshorttok`
6572 `\newtoks\glsshorttok`

`\glslongtok`
6573 `\newtoks\glslongtok`

`\newacronymhook` Provide a hook for `\newacronym`:
6574 `\newcommand*{\newacronymhook}{}`

`\genericNewAcronym` New improved version of setting the acronym style.
6575 `\newcommand*{\SetGenericNewAcronym}{%`
Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`
6576 `\let\@Gls@entryname\@Gls@acrentryname`
Change the way acronyms are defined:
6577 `\renewcommand{\newacronym}[4][\%`
6578 `\ifdefempty{\@glsacronymlists}%`
6579 `{%`
6580 `\def\@glo@type{\acronymtype}%`
6581 `\setkeys{glossentry}{##1}%`
6582 `\DeclareAcronymList{\@glo@type}%`
6583 `}%`
6584 `{}%`
6585 `\glskeylisttok{##1}%`
6586 `\glslabeltok{##2}%`
6587 `\glsshorttok{##3}%`
6588 `\glslongtok{##4}%`

```

6589 \newacronymhook
6590 \protected@edef\@do@newglossaryentry{%
6591 \noexpand\newglossaryentry{\the\glslabeltok}%
6592 {%
6593 type=\acronymtype,%
6594 name={\expandonce{\acronymentry{##2}}},%
6595 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6596 text={\the\glsshorttok},%
6597 short={\the\glsshorttok},%
6598 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6599 long={\the\glslongtok},%
6600 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6601 \GenericAcronymFields,%
6602 \the\glskeylisttok
6603 }%
6604 }%
6605 \@do@newglossaryentry
6606 }%

```

Make sure that \acrfull etc reflects the new style:

```

6607 \renewcommand*{\acrfullfmt}[3]{%
6608 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6609 \renewcommand*{\Acrfullfmt}[3]{%
6610 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6611 \renewcommand*{\ACRfullfmt}[3]{%
6612 \glslink[##1]{##2}{%
6613 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
6614 \renewcommand*{\acrfullplfmt}[3]{%
6615 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6616 \renewcommand*{\Acrfullplfmt}[3]{%
6617 \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6618 \renewcommand*{\ACRfullplfmt}[3]{%
6619 \glslink[##1]{##2}{%
6620 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6621 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6622 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6623 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6624 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6625 }

```

\icAcronymFields Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

6626 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

\acronymentry \acronymentry{<label>}

Display style for the name field in the list of acronyms.

```

6627 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```
6628 \newcommand*{\acronymsort}[2]{#1}
```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```
6629 \newcommand*{\setacronymstyle}[1]{%
6630   \ifcsundef{@glsacr@dispstyle@#1}
6631   {%
6632     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6633   }%
6634   {%
6635     \ifdefempty{\@glsacronymlists}%
6636     {%
6637       \DeclareAcronymList{\acronymtype}%
6638     }%
6639   }%
6640   \SetGenericNewAcronym
6641   \GlsUseAcrStyleDefs{#1}%
6642   \@for\@gls@type:=\@glsacronymlists\do{%
6643     \defglentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6644   }%
6645 }%
6646 }
```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```
6647 \newcommand*{\newacronymstyle}[3]{%
6648   \ifcsdef{@glsacr@dispstyle@#1}%
6649   {%
6650     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6651   }%
6652   {%
6653     \csdef{@glsacr@dispstyle@#1}{#2}%
6654     \csdef{@glsacr@styledefs@#1}{#3}%
6655   }%
6656 }
```

`\renewacronymstyle` Redefines the given acronym style.

```
6657 \newcommand*{\renewacronymstyle}[3]{%
6658   \ifcsdef{@glsacr@dispstyle@#1}%
6659   {%
```



```

6660 \csdef{@glsacr@dispstyle@#1}{#2}%
6661 \csdef{@glsacr@styledefs@#1}{#3}%
6662 }%
6663 {%
6664 \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6665 }%
6666 }

```

rEntryDispStyle

```

6667 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

UseAcrStyleDefs

```

6668 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short *<long>* (*<short>*) acronym style.

```

6669 \newacronymstyle{long-short}%
6670 {%

```

Check for long form in case this is a mixed glossary.

```

6671 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6672 }%
6673 {%
6674 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6675 \renewcommand*{\genacrfullformat}[2]{%
6676 \glentrylong{##1}##2\space
6677 (\protect\firstacronymfont{\glentryshort{##1}})%
6678 }%
6679 \renewcommand*{\Genacrfullformat}[2]{%
6680 \Glsentrylong{##1}##2\space
6681 (\protect\firstacronymfont{\glentryshort{##1}})%
6682 }%
6683 \renewcommand*{\genplacrfullformat}[2]{%
6684 \glentrylongpl{##1}##2\space
6685 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6686 }%
6687 \renewcommand*{\Genplacrfullformat}[2]{%
6688 \Glsentrylongpl{##1}##2\space
6689 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6690 }%
6691 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6692 \renewcommand*{\acronymsort}[2]{##1}%
6693 \renewcommand*{\acronymfont}[1]{##1}%
6694 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6695 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6696 }

```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```

6697 \newacronymstyle{long-sp-short}%
6698 {%
    Check for long form in case this is a mixed glossary.
6699 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6700 }%
6701 {%
6702 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6703 \renewcommand*{\genacrfullformat}[2]{%
6704 \glentrylong{##1}##2\glsacspace{##1}%
6705 (\protect\firstacronymfont{\glentryshort{##1}})%
6706 }%
6707 \renewcommand*{\Genacrfullformat}[2]{%
6708 \Glsentrylong{##1}##2\glsacspace{##1}%
6709 (\protect\firstacronymfont{\glentryshort{##1}})%
6710 }%
6711 \renewcommand*{\genplacrfullformat}[2]{%
6712 \glentrylongpl{##1}##2\glsacspace{##1}%
6713 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6714 }%
6715 \renewcommand*{\Genplacrfullformat}[2]{%
6716 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6717 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6718 }%
6719 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6720 \renewcommand*{\acronymsort}[2]{##1}%
6721 \renewcommand*{\acronymfont}[1]{##1}%
6722 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6723 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6724 }

```

`\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.

```

6725 \newcommand*{\glsacspace}[1]{%
6726 \settowidth{\dimen@}{(\firstacronymfont{\glentryshort{##1}})}%
6727 \ifdim\dimen@<3em~\else\space\fi
6728 }

```

`short-long` (*short*) (*long*) acronym style.

```

6729 \newacronymstyle{short-long}%
6730 {%
    Check for long form in case this is a mixed glossary.
6731 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6732 }%
6733 {%
6734 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6735 \renewcommand*{\genacrfullformat}[2]{%
6736 \protect\firstacronymfont{\glentryshort{##1}}##2\space
6737 (\glentrylong{##1})%

```

```

6738 }%
6739 \renewcommand*{\Genacrfullformat}[2]{%
6740   \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6741   (\glsentrylong{##1})%
6742 }%
6743 \renewcommand*{\genplacrfullformat}[2]{%
6744   \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6745   (\glsentrylongpl{##1})%
6746 }%
6747 \renewcommand*{\Genplacrfullformat}[2]{%
6748   \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6749   (\glsentrylongpl{##1})%
6750 }%

6751 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6752 \renewcommand*{\acronymsort}[2]{##1}%
6753 \renewcommand*{\acronymfont}[1]{##1}%
6754 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6755 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6756 }

```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```

6757 \newacronymstyle{long-sc-short}%
6758 {%
6759   \GlsUseAcrEntryDisplayStyle{long-short}%
6760 }%
6761 {%
6762   \GlsUseAcrStyleDefs{long-short}%
6763   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6764   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6765 }

```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6766 \newacronymstyle{long-sm-short}%
6767 {%
6768   \GlsUseAcrEntryDisplayStyle{long-short}%
6769 }%
6770 {%
6771   \GlsUseAcrStyleDefs{long-short}%
6772   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6773   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6774 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6775 \newacronymstyle{sc-short-long}%
6776 {%
6777   \GlsUseAcrEntryDisplayStyle{short-long}%
6778 }%
6779 {%

```

```

6780 \GlsUseAcrStyleDefs{short-long}%
6781 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6782 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6783 }

```

sm-short-long *<short>* (*\textsmaller{<long>}*) acronym style.

```

6784 \newacronymstyle{sm-short-long}%
6785 {%
6786 \GlsUseAcrEntryDisplayStyle{short-long}%
6787 }%
6788 {%
6789 \GlsUseAcrStyleDefs{short-long}%
6790 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6791 \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6792 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6793 \newacronymstyle{long-short-desc}%
6794 {%
6795 \GlsUseAcrEntryDisplayStyle{long-short}%
6796 }%
6797 {%
6798 \GlsUseAcrStyleDefs{long-short}%
6799 \renewcommand*{\GenericAcronymFields}{}%
6800 \renewcommand*{\acronymsort}[2]{##2}%
6801 \renewcommand*{\acronymentry}[1]{%
6802 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6803 }

```

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`.

```

6804 \newacronymstyle{long-sp-short-desc}%
6805 {%
6806 \GlsUseAcrEntryDisplayStyle{long-sp-short}%
6807 }%
6808 {%
6809 \GlsUseAcrStyleDefs{long-sp-short}%
6810 \renewcommand*{\GenericAcronymFields}{}%
6811 \renewcommand*{\acronymsort}[2]{##2}%
6812 \renewcommand*{\acronymentry}[1]{%
6813 \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6814 }

```

g-sc-short-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6815 \newacronymstyle{long-sc-short-desc}%
6816 {%

```

```

6817 \GlsUseAcrEntryDispStyle{long-sc-short}%
6818 }%
6819 {%
6820 \GlsUseAcrStyleDefs{long-sc-short}%
6821 \renewcommand*{\GenericAcronymFields}{}%
6822 \renewcommand*{\acronymsort}[2]{##2}%
6823 \renewcommand*{\acronymentry}[1]{%
6824     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6825 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6826 \newacronymstyle{long-sm-short-desc}%
6827 {%
6828 \GlsUseAcrEntryDispStyle{long-sm-short}%
6829 }%
6830 {%
6831 \GlsUseAcrStyleDefs{long-sm-short}%
6832 \renewcommand*{\GenericAcronymFields}{}%
6833 \renewcommand*{\acronymsort}[2]{##2}%
6834 \renewcommand*{\acronymentry}[1]{%
6835     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6836 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

6837 \newacronymstyle{short-long-desc}%
6838 {%
6839 \GlsUseAcrEntryDispStyle{short-long}%
6840 }%
6841 {%
6842 \GlsUseAcrStyleDefs{short-long}%
6843 \renewcommand*{\GenericAcronymFields}{}%
6844 \renewcommand*{\acronymsort}[2]{##2}%
6845 \renewcommand*{\acronymentry}[1]{%
6846     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6847 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6848 \newacronymstyle{sc-short-long-desc}%
6849 {%
6850 \GlsUseAcrEntryDispStyle{sc-short-long}%
6851 }%
6852 {%
6853 \GlsUseAcrStyleDefs{sc-short-long}%
6854 \renewcommand*{\GenericAcronymFields}{}%
6855 \renewcommand*{\acronymsort}[2]{##2}%
6856 \renewcommand*{\acronymentry}[1]{%

```

```

6857 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6858 }

```

short-long-desc <long> (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6859 \newacronymstyle{sm-short-long-desc}%
6860 {%
6861 \GlsUseAcrEntryDispStyle{sm-short-long}%
6862 }%
6863 {%
6864 \GlsUseAcrStyleDefs{sm-short-long}%
6865 \renewcommand*{\GenericAcronymFields}{}%
6866 \renewcommand*{\acronymsort}[2]{##2}%
6867 \renewcommand*{\acronymentry}[1]{%
6868 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6869 }

```

dua <long> only acronym style.

```

6870 \newacronymstyle{dua}%
6871 {%

```

Check for long form in case this is a mixed glossary.

```

6872 \ifdefempty\glscustomtext
6873 {%
6874 \ifglshaslong{\glslabel}%
6875 {%
6876 \glsifplural
6877 {%

```

Plural form:

```

6878 \glscapscase
6879 {%

```

Plural form, don't adjust case:

```

6880 \glsentrylongpl{\glslabel}\glsinsert
6881 }%
6882 {%

```

Plural form, make first letter upper case:

```

6883 \Glsentrylongpl{\glslabel}\glsinsert
6884 }%
6885 {%

```

Plural form, all caps:

```

6886 \mfirstucMakeUppercase
6887 {\glsentrylongpl{\glslabel}\glsinsert}%
6888 }%
6889 }%
6890 {%

```

Singular form

```
6891      \glscapscase
6892      {%
```

Singular form, don't adjust case:

```
6893      \glentrylong{\glslabel}\glsinsert
6894      }%
6895      {%
```

Subsequent singular form, make first letter upper case:

```
6896      \Glsentrylong{\glslabel}\glsinsert
6897      }%
6898      {%
```

Subsequent singular form, all caps:

```
6899      \mfirstucMakeUppercase
6900      {\glentrylong{\glslabel}\glsinsert}%
6901      }%
6902      }%
6903      }%
6904      {%
```

Not an acronym:

```
6905      \glsgenentryfmt
6906      }%
6907      }%
6908      {\glscustomtext\glsinsert}%
6909      }%
6910      {%
6911      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6912      \renewcommand*{\acrfullfmt}[3]{%
6913          \glslink[##1]{##2}{\glentrylong{##2}##3\space
6914              (\acronymfont{\glentryshort{##2}})}}%
6915      \renewcommand*{\Acrfullfmt}[3]{%
6916          \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6917              (\acronymfont{\glentryshort{##2}})}}%
6918      \renewcommand*{\ACRfullfmt}[3]{%
6919          \glslink[##1]{##2}{%
6920              \mfirstucMakeUppercase{\glentrylong{##2}##3\space
6921                  (\acronymfont{\glentryshort{##2}})}}}%

6922      \renewcommand*{\acrfullplfmt}[3]{%
6923          \glslink[##1]{##2}{\glentrylongpl{##2}##3\space
6924              (\acronymfont{\glentryshortpl{##2}})}}%

6925      \renewcommand*{\Acrfullplfmt}[3]{%
6926          \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6927              (\acronymfont{\glentryshortpl{##2}})}}%
6928      \renewcommand*{\ACRfullplfmt}[3]{%
6929          \glslink[##1]{##2}{%
```

```

6930      \mfirstucMakeUppercase{\glentrylongpl{##2}##3\space
6931      (\acronymfont{\glentryshortpl{##2}})}}}%
6932 \renewcommand*{\glentryfull}[1]{%
6933   \glentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6934 }%
6935 \renewcommand*{\Glsentryfull}[1]{%
6936   \Glsentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6937 }%
6938 \renewcommand*{\glentryfullpl}[1]{%
6939   \glentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6940 }%
6941 \renewcommand*{\Glsentryfullpl}[1]{%
6942   \Glsentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6943 }%
6944 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6945 \renewcommand*{\acronymsort}[2]{##1}%
6946 \renewcommand*{\acronymfont}[1]{##1}%
6947 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6948 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6949 \newacronymstyle{dua-desc}%
6950 {%
6951   \GlsUseAcrEntryDispStyle{dua}%
6952 }%
6953 {%
6954   \GlsUseAcrStyleDefs{dua}%
6955   \renewcommand*{\GenericAcronymFields}{}%
6956   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
6957   \renewcommand*{\acronymsort}[2]{##2}%
6958 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6959 \newacronymstyle{footnote}%
6960 {%
6961   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6962 }%
6963 {%
6964   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6965   \glshyperfirstfalse
6966   \renewcommand*{\genacrfullformat}[2]{%
6967     \protect\firstacronymfont{\glentryshort{##1}}##2%
6968     \protect\footnote{\glentrylong{##1}}%
6969   }%
6970   \renewcommand*{\Genacrfullformat}[2]{%

```



```

6971 \firstacronymfont{\Glsentryshort{##1}}##2%
6972 \protect\footnote{\glsentrylong{##1}}%
6973 }%
6974 \renewcommand*{\genplacrfullformat}[2]{%
6975 \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
6976 \protect\footnote{\glsentrylongpl{##1}}%
6977 }%
6978 \renewcommand*{\Genplacrfullformat}[2]{%
6979 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6980 \protect\footnote{\glsentrylongpl{##1}}%
6981 }%
6982 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6983 \renewcommand*{\acronymsort}[2]{##1}%
6984 \renewcommand*{\acronymfont}[1]{##1}%
6985 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6986 \renewcommand*{\acrfullfmt}[3]{%
6987 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
6988 (\glsentrylong{##2})}%
6989 \renewcommand*{\Acrfullfmt}[3]{%
6990 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6991 (\glsentrylong{##2})}%
6992 \renewcommand*{\ACRfullfmt}[3]{%
6993 \glslink[##1]{##2}{%
6994 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
6995 (\glsentrylong{##2})}}}%
6996 \renewcommand*{\acrfullplfmt}[3]{%
6997 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
6998 (\glsentrylongpl{##2})}%
6999 \renewcommand*{\Acrfullplfmt}[3]{%
7000 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
7001 (\glsentrylongpl{##2})}%
7002 \renewcommand*{\ACRfullplfmt}[3]{%
7003 \glslink[##1]{##2}{%
7004 \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
7005 (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

7006 \renewcommand*{\glsentryfull}[1]{%
7007 \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
7008 \renewcommand*{\Glsentryfull}[1]{%
7009 \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
7010 \renewcommand*{\glsentryfullpl}[1]{%
7011 \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7012 \renewcommand*{\Glsentryfullpl}[1]{%
7013 \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
7014 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

7015 \newacronymstyle{footnote-sc}%
7016 {%
7017   \GlsUseAcrEntryDisplayStyle{footnote}%
7018 }%
7019 {%
7020   \GlsUseAcrStyleDefs{footnote}%
7021   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
7022   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
7023   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7024 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

7025 \newacronymstyle{footnote-sm}%
7026 {%
7027   \GlsUseAcrEntryDisplayStyle{footnote}%
7028 }%
7029 {%
7030   \GlsUseAcrStyleDefs{footnote}%
7031   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
7032   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
7033   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
7034 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7035 \newacronymstyle{footnote-desc}%
7036 {%
7037   \GlsUseAcrEntryDisplayStyle{footnote}%
7038 }%
7039 {%
7040   \GlsUseAcrStyleDefs{footnote}%
7041   \renewcommand*{\GenericAcronymFields}{}%
7042   \renewcommand*{\acronymsort}[2]{##2}%
7043   \renewcommand*{\acronymentry}[1]{%
7044     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7045 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7046 \newacronymstyle{footnote-sc-desc}%
7047 {%
7048   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
7049 }%
7050 {%
7051   \GlsUseAcrStyleDefs{footnote-sc}%
7052   \renewcommand*{\GenericAcronymFields}{}%
7053   \renewcommand*{\acronymsort}[2]{##2}%
7054   \renewcommand*{\acronymentry}[1]{%
7055     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

```

7056 }

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

7057 \newacronymstyle{footnote-sm-desc}%

7058 {%

7059 \GlsUseAcrEntryDispStyle{footnote-sm}%

7060 }%

7061 {%

7062 \GlsUseAcrStyleDefs{footnote-sm}%

7063 \renewcommand*{\GenericAcronymFields}{}%

7064 \renewcommand*{\acronymsort}[2]{##2}%

7065 \renewcommand*{\acronymentry}[1]{%

7066 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

7067 }

AcronymSynonyms

7068 \newcommand*{\DefineAcronymSynonyms}{%

Short form

\acs

7069 \let\acs\acrshort

First letter uppercase short form

\Acs

7070 \let\Acs\Acrshort

Plural short form

\acsp

7071 \let\acsp\acrshortpl

First letter uppercase plural short form

\Acsp

7072 \let\Acsp\Acrshortpl

Long form

\acl

7073 \let\acl\aclong

Plural long form

\aclp

7074 \let\aclp\aclongpl

First letter upper case long form

`\Acl`
 7075 `\let\Acl\Aclong`
 First letter upper case plural long form

`\Aclp`
 7076 `\let\Aclp\Aclongpl`
 Full form

`\acf`
 7077 `\let\acf\acrfull`
 Plural full form

`\acfp`
 7078 `\let\acfp\acrfullpl`
 First letter upper case full form

`\Acf`
 7079 `\let\Acf\Acrfull`
 First letter upper case plural full form

`\Acfp`
 7080 `\let\Acfp\Acrfullpl`
 Standard form

`\ac`
 7081 `\let\ac\gls`
 First upper case standard form

`\Ac`
 7082 `\let\Ac\Gls`
 Standard plural form

`\acp`
 7083 `\let\acp\glspl`
 Standard first letter upper case plural form

`\Acp`
 7084 `\let\Acp\Glspl`
 7085 `}`
 Define synonyms if required

7086 `\ifglsacrshortcuts`
 7087 `\DefineAcronymSynonyms`
 7088 `\fi`

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\setAcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
7089 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
7090   \def\glsentryfmt[#1]{\glsentryfmt}%
7091 }
```

`\setNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
7092 \newcommand*{\DefaultNewAcronymDef}{%
7093   \edef\@do@newglossaryentry{%
7094     \noexpand\newglossaryentry{\the\glslabeltok}%
7095     {%
7096       type=\acronymtype,%
7097       name={\the\glsshorttok},%
7098       sort={\the\glsshorttok},%
7099       text={\the\glsshorttok},%
7100       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7101       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7102       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7103                     {\noexpand\expandonce\noexpand\@glo@shortpl}},%
7104       short={\the\glsshorttok},%
7105       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7106       long={\the\glslongtok},%
7107       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7108       description={\the\glslongtok},%
7109       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
7110       \the\glskeylisttok
7111     }%
7112   }%
7113   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7114   \let\@org@gls@assign@plural\gls@assign@plural
7115   \let\@org@gls@assign@descplural\gls@assign@descplural
7116   \def\gls@assign@firstpl##1##2{%
7117     \@@gls@expand@field{##1}{firstpl}{##2}%
7118   }%
7119   \def\gls@assign@plural##1##2{%
7120     \@@gls@expand@field{##1}{plural}{##2}%
7121   }%
7122   \def\gls@assign@descplural##1##2{%
7123     \@@gls@expand@field{##1}{descplural}{##2}%
7124   }%
7125   \@do@newglossaryentry
7126   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7127   \let\gls@assign@plural\@org@gls@assign@plural
7128   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7129 }
```

ultAcronymStyle Set up the default acronym style:

```
7130 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```
7131 \@for\@gls@type:=\@glsacronymlists\do{%
```

```
7132 \SetDefaultAcronymDisplayStyle{\@gls@type}%
```

```
7133 }%
```

Set up the definition of \newacronym:

```
7134 \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).

```
7135 \ifx\@glsacronymlists\empty
```

```
7136 \def\@glo@type{\acronymtype}%
```

```
7137 \setkeys{glossentry}{##1}%
```

```
7138 \DeclareAcronymList{\@glo@type}%
```

```
7139 \SetDefaultAcronymDisplayStyle{\@glo@type}%
```

```
7140 \fi
```

```
7141 \glskeylisttok{##1}%
```

```
7142 \glslabeltok{##2}%
```

```
7143 \glsshorttok{##3}%
```

```
7144 \glslongtok{##4}%
```

```
7145 \newacronymhook
```

```
7146 \DefaultNewAcronymDef
```

```
7147 }%
```

```
7148 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
```

```
7149 }
```

\acrfootnote Used by the footnote acronym styles.

```
7150 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

acrlinkfootnote

```
7151 \newcommand*{\acrlinkfootnote}[3]{%
```

```
7152 \footnote{\glslink[#1]{#2}{#3}}%
```

```
7153 }
```

acrnolinkfootnote

```
7154 \newcommand*{\acrnolinkfootnote}[3]{%
```

```
7155 \footnote{#3}%
```

```
7156 }
```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
7157 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
```

```
7158 \defglsentryfmt[#1]{%
```

```
7159 \ifdefempty\glscustomtext
```

```
7160 {%
```

```
7161 \ifglsused{\glslabel}%
```

```

7162      {%
7163      \acronymfont{\glsgenentryfmt}%
7164      }%
7165      {%
7166      \firstacronymfont{\glsgenentryfmt}%
7167      \ifglshassymbol{\glslabel}%
7168      {%
7169      \expandafter\protect\expandafter\acrfootnote\expandafter
7170      {\@gls@link@opts}{\@gls@link@label}%
7171      {%
7172      \glsifplural
7173      {\glsentrysymbolplural{\glslabel}}%
7174      {\glsentrysymbol{\glslabel}}%
7175      }%
7176      }%
7177      }%
7178      }%
7179      {\glscustomtext\glsinsert}%
7180      }%
7181      }

```

teNewAcronymDef

```

7182 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7183 \edef\@do@newglossaryentry{%
7184 \noexpand\newglossaryentry{\the\glslabeltok}%
7185 {%
7186 type=\acronymtype,%
7187 name={\noexpand\acronymfont{\the\glsshorttok}},%
7188 sort={\the\glsshorttok},%
7189 first={\the\glsshorttok},%
7190 firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7191 text={\the\glsshorttok},%
7192 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7193 short={\the\glsshorttok},%
7194 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7195 long={\the\glslongtok},%
7196 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7197 symbol={\the\glslongtok},%
7198 symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7199 \the\glskeylisttok
7200 }%
7201 }%
7202 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7203 \let\@org@gls@assign@plural\gls@assign@plural
7204 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7205 \def\gls@assign@firstpl##1##2{%
7206 \@@gls@expand@field{##1}{firstpl}{##2}%
7207 }%
7208 \def\gls@assign@plural##1##2{%

```

```

7209 \@@gls@expand@field{##1}{plural}{##2}%
7210 }%
7211 \def\gls@assign@symbolplural##1##2{%
7212 \@@gls@expand@field{##1}{symbolplural}{##2}%
7213 }%
7214 \do@newglossaryentry
7215 \let\gls@assign@plural\@org@gls@assign@plural
7216 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7217 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7218 }

```

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.

```

7219 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
7220 \renewcommand{\newacronym}[4][\]{%
7221 \ifx\@glsacronymlists\@empty
7222 \def\@glo@type{\acronymtype}%
7223 \setkeys{glossentry}{##1}%
7224 \DeclareAcronymList{\@glo@type}%
7225 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7226 \fi
7227 \glskeylisttok{##1}%
7228 \glslabeltok{##2}%
7229 \glsshorttok{##3}%
7230 \glslongtok{##4}%
7231 \newacronymhook
7232 \DescriptionFootnoteNewAcronymDef
7233 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7234 \@for\@gls@type:=\@glsacronymlists\do{%
7235 \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7236 }%

```

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.

```

7237 \ifglsacrsmallcaps
7238 \renewcommand*\acronymfont[1]{\textsc{##1}}%
7239 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7240 \else
7241 \ifglsacrsmaller
7242 \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
7243 \fi
7244 \fi

```

Check for package option clash


```

7245 \ifglsacrdue
7246   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7247     can’t both be set}{}%
7248 \fi
7249 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

7250 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7251   \defglsentryfmt[#1]{\glsentryfmt}%
7252 }

```

UANewAcronymDef

```

7253 \newcommand*{\DescriptionDUANewAcronymDef}{%
7254   \edef\@do@newglossaryentry{%
7255     \noexpand\newglossaryentry{\the\glslabeltok}%
7256     {%
7257       type=\acronymtype,%
7258       name={\the\glslongtok},%
7259       sort={\the\glslongtok},%
7260       text={\the\glslongtok},%
7261       first={\the\glslongtok},%
7262       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7263       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7264       short={\the\glsshorttok},%
7265       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7266       long={\the\glslongtok},%
7267       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7268       symbol={\the\glsshorttok},%
7269       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7270       \the\glskeylisttok
7271     }%
7272   }%
7273   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7274   \let\@org@gls@assign@plural\gls@assign@plural
7275   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7276   \def\gls@assign@firstpl##1##2{%
7277     \@@gls@expand@field{##1}{firstpl}{##2}%
7278   }%
7279   \def\gls@assign@plural##1##2{%
7280     \@@gls@expand@field{##1}{plural}{##2}%
7281   }%
7282   \def\gls@assign@symbolplural##1##2{%
7283     \@@gls@expand@field{##1}{symbolplural}{##2}%
7284   }%
7285   \@do@newglossaryentry
7286   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7287   \let\gls@assign@plural\@org@gls@assign@plural
7288   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7289 }

```

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.

```

7290 \newcommand*{\SetDescriptionDUAACronymStyle}{%
7291   \ifglsmallcaps
7292     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7293       can't both be set}{}%
7294   \else
7295     \ifglsmaller
7296       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7297         can't both be set}{}%
7298     \fi
7299   \fi
7300   \renewcommand{\newacronym}[4][{}]{%
7301     \ifx\@glsacronymlists\@empty
7302       \def\@glo@type{\acronymtype}%
7303       \setkeys{glossentry}{##1}%
7304       \DeclareAcronymList{\@glo@type}%
7305       \SetDescriptionDUAACronymDisplayStyle{\@glo@type}%
7306     \fi
7307     \glskeylisttok{##1}%
7308     \glslabeltok{##2}%
7309     \glsshorttok{##3}%
7310     \glslongtok{##4}%
7311     \newacronymhook
7312     \DescriptionDUANewAcronymDef
7313   }%

```

Set display.

```

7314   \@for\@gls@type:=\@glsacronymlists\do{%
7315     \SetDescriptionDUAACronymDisplayStyle{\@gls@type}%
7316   }%
7317 }%

```

nymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7318 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7319   \defglentryfmt[#1]{%
7320     \ifdefempty\glscustomtext
7321     {%
7322       \ifglused{\glslabel}%
7323     }%

```

Move the inserted text outside of \acronymfont

```

7324       \let\gls@org@insert\glsinsert
7325       \let\glsinsert\@empty
7326       \acronymfont{\glsgenentryfmt}\gls@org@insert
7327     }%

```

```

7328     {%
7329         \glsgenentryfmt
7330         \ifglshassymbol{\glslabel}%
7331         {%
7332             \glsifplural
7333             {%
7334                 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7335             }%
7336             {%
7337                 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7338             }%
7339             \space(\protect\firstacronymfont
7340             {\glscapscase
7341              {\@glo@symbol}
7342              {\@glo@symbol}
7343              {\mfirstucMakeUppercase{\@glo@symbol}}})%
7344         }%
7345     }%
7346 }%
7347 }%
7348 {\glscustomtext\glsinsert}%
7349 }%
7350 }

```

onNewAcronymDef

```

7351 \newcommand*{\DescriptionNewAcronymDef}{%
7352 \edef\@do@newglossaryentry{%
7353 \noexpand\newglossaryentry{\the\glslabeltok}%
7354 {%
7355     type=\acronymtype,%
7356     name={\noexpand
7357         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
7358     sort={\the\glsshorttok},%
7359     first={\the\glslongtok},%
7360     firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7361     text={\the\glsshorttok},%
7362     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7363     short={\the\glsshorttok},%
7364     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7365     long={\the\glslongtok},%
7366     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7367     symbol={\noexpand\@glo@text},%
7368     symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7369     \the\glskeylisttok}%
7370 }%
7371 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7372 \let\@org@gls@assign@plural\gls@assign@plural
7373 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7374 \def\gls@assign@firstpl##1##2{%

```

```

7375 \@@gls@expand@field{##1}{firstpl}{##2}%
7376 }%
7377 \def\gls@assign@plural##1##2{%
7378 \@@gls@expand@field{##1}{plural}{##2}%
7379 }%
7380 \def\gls@assign@symbolplural##1##2{%
7381 \@@gls@expand@field{##1}{symbolplural}{##2}%
7382 }%
7383 \do@newglossaryentry
7384 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7385 \let\gls@assign@plural\@org@gls@assign@plural
7386 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7387 }

```

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.

```

7388 \newcommand*{\SetDescriptionAcronymStyle}{%
7389 \renewcommand{\newacronym}[4][\]{%
7390 \ifx\@glsacronymlists\@empty
7391 \def\@glo@type{\acronymtype}%
7392 \setkeys{glossentry}{##1}%
7393 \DeclareAcronymList{\@glo@type}%
7394 \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7395 \fi
7396 \glskeylisttok{##1}%
7397 \glslabeltok{##2}%
7398 \glsshorttok{##3}%
7399 \glslongtok{##4}%
7400 \newacronymhook
7401 \DescriptionNewAcronymDef
7402 }%

```

Set display.

```

7403 \@for\@gls@type:=\@glsacronymlists\do{%
7404 \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7405 }%

```

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.

```

7406 \ifglsacrsmallcaps
7407 \renewcommand{\acronymfont}[1]{\textsc{##1}}
7408 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7409 \else
7410 \ifglsacrsmaller
7411 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7412 \fi
7413 \fi
7414 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7415 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7416   \defglentryfmt[#1]{%

7417     \ifdefempty\glscustomtext
7418     {%

      Move the inserted text outside of \acronymfont

7419       \let\gls@org@insert\glsinsert
7420       \let\glsinsert\@empty
7421       \ifglused{\glslabel}%
7422       {%
7423         \acronymfont{\glsgenentryfmt}\gls@org@insert
7424       }%
7425       {%
7426         \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7427         \ifglshaslong{\glslabel}%
7428         {%
7429           \expandafter\protect\expandafter\acrfootnote\expandafter
7430           {\@gls@link@opts}{\@gls@link@label}%
7431           {%
7432             \glsifplural
7433             {\glsentrylongpl{\glslabel}}%
7434             {\glsentrylong{\glslabel}}%
7435           }%
7436         }%

7437       }%
7438     }%
7439   }%
7440   {\glscustomtext\glsinsert}%
7441 }%
7442 }
```

teNewAcronymDef

```

7443 \newcommand*{\FootnoteNewAcronymDef}{%
7444   \edef\@do@newglossaryentry{%
7445     \noexpand\newglossaryentry{\the\glslabeltok}%
7446     {%
7447       type=\acronymtype,%
7448       name={\noexpand\acronymfont{\the\glsshorttok}},%
7449       sort={\the\glsshorttok},%
7450       text={\the\glsshorttok},%
7451       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7452       first={\the\glsshorttok},%
7453       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7454       short={\the\glsshorttok},%
7455       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7456       long={\the\glslongtok},%
```

```

7457     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7458     description={\the\glslongtok},%
7459     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7460     \the\glskeylisttok
7461   }%
7462 }%
7463 \let\@org@gls@assign@plural\gls@assign@plural
7464 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7465 \let\@org@gls@assign@descplural\gls@assign@descplural
7466 \def\gls@assign@firstpl##1##2{%
7467   \@gls@expand@field{##1}{firstpl}{##2}%
7468 }%
7469 \def\gls@assign@plural##1##2{%
7470   \@gls@expand@field{##1}{plural}{##2}%
7471 }%
7472 \def\gls@assign@descplural##1##2{%
7473   \@gls@expand@field{##1}{descplural}{##2}%
7474 }%
7475 \do@newglossaryentry
7476 \let\gls@assign@plural\@org@gls@assign@plural
7477 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7478 \let\gls@assign@descplural\@org@gls@assign@descplural
7479 }

```

oteAcronymStyle If footnote package option is specified, set the first use to append the long form (stored in description) as a footnote. Use the description key to store the long form.

```

7480 \newcommand*{\SetFootnoteAcronymStyle}{%
7481   \renewcommand{\newacronym}[4][]{%
7482     \ifx\@glsacronymlists\@empty
7483       \def\@glo@type{\acronymtype}%
7484       \setkeys{glossentry}{##1}%
7485       \DeclareAcronymList{\@glo@type}%
7486       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7487     \fi
7488     \glskeylisttok{##1}%
7489     \glslabeltok{##2}%
7490     \glsshorttok{##3}%
7491     \glslongtok{##4}%
7492     \newacronymhook
7493     \FootnoteNewAcronymDef
7494   }%

```

Set display

```

7495   \@for\@gls@type:=\@glsacronymlists\do{%
7496     \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7497   }%

```

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.

```

7498   \ifglsacrsmallcaps

```

```

7499     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7500     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7501 \else
7502     \ifglsmaller
7503         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7504     \fi
7505 \fi

    Check for option clash
7506 \ifglacrdua
7507     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7508         can’t both be set}{}%
7509 \fi
7510 }%

```

`\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.

```

7511 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7512     \protected@edef\gls@tmp{#1}%
7513     \ifdefempty\gls@tmp
7514     {}%
7515     {%
7516         \ifx\gls@tmp\gls@default@value
7517         \else
7518             \space (#2{#1})%
7519         \fi
7520     }%
7521 }

```

`\setacronymdisplaystyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7522 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7523     \defglentryfmt[#1]{%

7524     \ifdefempty\glscustomtext
7525     {%

```

Move the inserted text outside of `\acronymfont`

```

7526         \let\gls@org@insert\glsinsert
7527         \let\glsinsert\@empty
7528         \ifglused{\glslabel}%
7529         {%
7530             \acronymfont{\glsgenentryfmt}\gls@org@insert
7531         }%
7532         {%
7533             \glsgenentryfmt
7534             \ifglshassymbol{\glslabel}%
7535             {%
7536                 \glsifplural
7537                 {%

```

```

7538         \def\@glo@symbol{\glentrysymbolplural{\glslabel}}%
7539     }%
7540     {%
7541         \def\@glo@symbol{\glentrysymbol{\glslabel}}%
7542     }%
7543     \space
7544     (\glscapscase
7545     {\firstacronymfont{\@glo@symbol}}%
7546     {\firstacronymfont{\@glo@symbol}}%
7547     {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7548 }%
7549 {}%
7550 }%
7551 }%
7552 {\glscustomtext\glsinsert}%
7553 }%
7554 }

```

11NewAcronymDef

```

7555 \newcommand*{\SmallNewAcronymDef}{%
7556     \edef\@do@newglossaryentry{%
7557         \noexpand\newglossaryentry{\the\glslabeltok}%
7558         {%
7559             type=\acronymtype,%
7560             name={\noexpand\acronymfont{\the\glsshorttok}},%
7561             sort={\the\glsshorttok},%
7562             text={\the\glsshorttok},%
7563             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7564             first={\the\glslongtok},%
7565             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7566             short={\the\glsshorttok},%
7567             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7568             long={\the\glslongtok},%
7569             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7570             description={\noexpand\@glo@first},%
7571             descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7572             symbol={\the\glsshorttok},%
7573             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7574             \the\glskeylisttok
7575         }%
7576     }%
7577     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7578     \let\@org@gls@assign@plural\gls@assign@plural
7579     \let\@org@gls@assign@descplural\gls@assign@descplural

```



```

7580 \let\org@gl@s@assign@symbolplural\gl@s@assign@symbolplural
7581 \def\gl@s@assign@firstpl##1##2{%
7582   \@@gl@s@expand@field{##1}{firstpl}{##2}%
7583 }%
7584 \def\gl@s@assign@plural##1##2{%
7585   \@@gl@s@expand@field{##1}{plural}{##2}%
7586 }%
7587 \def\gl@s@assign@descplural##1##2{%
7588   \@@gl@s@expand@field{##1}{descplural}{##2}%
7589 }%
7590 \def\gl@s@assign@symbolplural##1##2{%
7591   \@@gl@s@expand@field{##1}{symbolplural}{##2}%
7592 }%
7593 \do@newglossaryentry
7594 \let\gl@s@assign@firstpl\org@gl@s@assign@firstpl
7595 \let\gl@s@assign@plural\org@gl@s@assign@plural
7596 \let\gl@s@assign@descplural\org@gl@s@assign@descplural
7597 \let\gl@s@assign@symbolplural\org@gl@s@assign@symbolplural
7598 }

```

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.

```

7599 \newcommand*{\SetSmallAcronymStyle}{%
7600   \renewcommand{\newacronym}[4][]{%
7601     \ifx\org@gl@s@acronymlists\@empty
7602       \def\orglo@type{\acronymtype}%
7603       \setkeys{glossentry}{##1}%
7604       \DeclareAcronymList{\orglo@type}%
7605       \SetSmallAcronymDisplayStyle{\orglo@type}%
7606     \fi
7607     \glskeylisttok{##1}%
7608     \glslabeltok{##2}%
7609     \glsshorttok{##3}%
7610     \glslongtok{##4}%
7611     \newacronymhook
7612     \SmallNewAcronymDef
7613   }%

```

Change the display since first only contains long form.

```

7614 \@for\orgls@type:=\org@gl@s@acronymlists\do{%
7615   \SetSmallAcronymDisplayStyle{\orgls@type}%
7616 }%

```

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.

```

7617 \ifgl@acrsmallcaps
7618   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7619   \renewcommand*{\acrpluralsuffix}{\gl@supacrpluralsuffix}%
7620 \else
7621   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}

```

```

7622 \fi
      check for option clash
7623 \ifglsacrdua
7624   \ifglsacrsmallcaps
7625     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7626       can’t both be set}{}%
7627   \else
7628     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7629       can’t both be set}{}%
7630   \fi
7631 \fi
7632 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7633 \newcommand*{\SetDUADisplayStyle}[1]{%
7634   \defglsentryfmt[#1]{\glsentryfmt}%
7635 }

```

UANewAcronymDef

```

7636 \newcommand*{\DUANewAcronymDef}{%
7637   \edef\@do@newglossaryentry{%
7638     \noexpand\newglossaryentry{\the\glslabeltok}%
7639     {%
7640       type=\acronymtype,%
7641       name={\the\glsshorttok},%
7642       text={\the\glslongtok},%
7643       first={\the\glslongtok},%
7644       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7645       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7646       short={\the\glsshorttok},%
7647       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7648       long={\the\glslongtok},%
7649       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7650       description={\the\glslongtok},%
7651       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7652       symbol={\the\glsshorttok},%
7653       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7654       \the\glskeylisttok
7655     }%
7656   }%
7657   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7658   \let\@org@gls@assign@plural\gls@assign@plural
7659   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7660   \let\@org@gls@assign@descplural\gls@assign@descplural
7661   \def\gls@assign@firstpl##1##2{%
7662     \@@gls@expand@field{##1}{firstpl}{##2}%
7663   }%
7664   \def\gls@assign@plural##1##2{%
7665     \@@gls@expand@field{##1}{plural}{##2}%

```

```

7666 }%
7667 \def\gls@assign@symbolplural##1##2{%
7668   \@gls@expand@field{##1}{symbolplural}{##2}%
7669 }%
7670 \def\gls@assign@descplural##1##2{%
7671   \@gls@expand@field{##1}{descplural}{##2}%
7672 }%
7673 \do@newglossaryentry
7674 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7675 \let\gls@assign@plural\@org@gls@assign@plural
7676 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7677 \let\gls@assign@descplural\@org@gls@assign@descplural
7678 }

```

\SetDUASStyle Always expand acronyms.

```

7679 \newcommand*{\SetDUASStyle}{%
7680   \renewcommand{\newacronym}[4][]{%
7681     \ifx\@glsacronymlists\@empty
7682       \def\@glo@type{\acronymtype}%
7683       \setkeys{glossentry}{##1}%
7684       \DeclareAcronymList{\@glo@type}%
7685       \SetDUADisplayStyle{\@glo@type}%
7686     \fi
7687     \glskeylisttok{##1}%
7688     \glslabeltok{##2}%
7689     \glsshorttok{##3}%
7690     \glslongtok{##4}%
7691     \newacronymhook
7692     \DUANewAcronymDef
7693   }%
7694   \Set the display
7695   \@for\@gls@type:=\@glsacronymlists\do{%
7696     \SetDUADisplayStyle{\@gls@type}%
7697   }%
7698 }

```

SetAcronymStyle

```

7698 \newcommand*{\SetAcronymStyle}{%
7699   \SetDefaultAcronymStyle
7700   \ifglsacrdescription
7701     \ifglsacrfootnote
7702       \SetDescriptionFootnoteAcronymStyle
7703     \else
7704       \ifglsacrdua
7705         \SetDescriptionDUAAcronymStyle
7706       \else
7707         \SetDescriptionAcronymStyle
7708       \fi
7709     \fi

```

```

7710 \else
7711   \ifglsacrfootnote
7712     \SetFootnoteAcronymStyle
7713   \else
7714     \ifthenelse{\boolean{glsacrsmalldcaps}\OR
7715       \boolean{glsacrsmalld}}{
7716       {%
7717         \SetSmallAcronymStyle
7718       }%
7719     {%
7720       \ifglsacrdua
7721         \SetDUASStyle
7722       \fi
7723     }%
7724   \fi
7725 \fi
7726 }

```

Set the acronym style according to the package options

```
7727 \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.

`\setacronymstyle` Sets the acronym display style.

```

7728 \newcommand*{\SetCustomDisplayStyle}[1]{%
7729   \defglsentryfmt[#1]{\glsentryfmt}%
7730 }

```

`\setacronymfields`

```

7731 \newcommand*{\CustomAcronymFields}{%
7732   name={\the\glsshorttok},%
7733   description={\the\glslongtok},%
7734   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7735   firstplural={\acrfullformat
7736     {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7737     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

7738   text={\the\glsshorttok},%
7739   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7740 }

```

`\setnewacronymdef`

```

7741 \newcommand*{\CustomNewAcronymDef}{%
7742   \protected@edef\do@newglossaryentry{%
7743     \noexpand\newglossaryentry{\the\glslabeltok}%
7744     {%

```

```

7745     type=\acronymtype,%
7746     short={\the\glsshorttok},%
7747     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7748     long={\the\glslongtok},%
7749     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7750     user1={\the\glsshorttok},%
7751     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7752     user3={\the\glslongtok},%
7753     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7754     \CustomAcronymFields,%
7755     \the\glskeylisttok
7756   }%
7757 }%
7758 \@do@newglossaryentry
7759 }

```

\SetCustomStyle

```

7760 \newcommand*{\SetCustomStyle}{%
7761   \renewcommand{\newacronym}[4][]{%
7762     \ifx\@glsacronymlists\@empty
7763       \def\@gls@type{\acronymtype}%
7764       \setkeys{glossentry}{##1}%
7765       \DeclareAcronymList{\@gls@type}%
7766       \SetCustomDisplayStyle{\@gls@type}%
7767     \fi
7768     \glskeylisttok{##1}%
7769     \glslabeltok{##2}%
7770     \glsshorttok{##3}%
7771     \glslongtok{##4}%
7772     \newacronymhook
7773     \CustomNewAcronymDef
7774   }%
7775   Set the display
7776   \@for\@gls@type:=\@glsacronymlists\do{%
7777     \SetCustomDisplayStyle{\@gls@type}%
7778   }%
7779 }

```

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.

```
7779 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
7780 \@gls@loadlist
```

The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
7781 \@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.

```
7782 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7783 \@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.

```
7784 \ifx\@glossary@default@style\relax
```

```
7785 \else
```

```
7786   \setglossarystyle{\@glossary@default@style}
```

```
7787 \fi
```

1.20 Debugging Commands

```
\showgloparent \showgloparent{\label{}}
```

```
7788 \newcommand*{\showgloparent}[1]{%
```

```
7789   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7790 }
```

```
\showglolevel \showglolevel{\label{}}
```

```
7791 \newcommand*{\showglolevel}[1]{%
```

```
7792   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7793 }
```

```
\showglotext \showglotext{\label{}}
```

```
7794 \newcommand*{\showglotext}[1]{%
```

```
7795   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7796 }
```

```
\showgloplural \showgloplural{\label{}}
```

```

7797 \newcommand*{\showgloplural}[1]{%
7798   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7799 }

```

\showglofirst \showglofirst{<label>}

```

7800 \newcommand*{\showglofirst}[1]{%
7801   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7802 }

```

\showglofirstpl \showglofirstpl{<label>}

```

7803 \newcommand*{\showglofirstpl}[1]{%
7804   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7805 }

```

\showgloftype \showgloftype{<label>}

```

7806 \newcommand*{\showgloftype}[1]{%
7807   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7808 }

```

\showglocounter \showglocounter{<label>}

```

7809 \newcommand*{\showglocounter}[1]{%
7810   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7811 }

```

\showglouser \showglouser{<label>}

```

7812 \newcommand*{\showglouser}[1]{%
7813   \expandafter\show\csname glo@glstetoklabel{#1}@user\endcsname
7814 }

```

\showglouserii \showglouserii{<label>}

```

7815 \newcommand*{\showglouserii}[1]{%
7816   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
7817 }

```

\showglouseriii \showglouseriii{<label>}

```

7818 \newcommand*{\showglouseriii}[1]{%
7819   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
7820 }

```

\showglouseriv \showglouseriv{<label>}

```

7821 \newcommand*{\showglouseriv}[1]{%
7822   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
7823 }

```

\showglouserv \showglouserv{<label>}

```

7824 \newcommand*{\showglouserv}[1]{%
7825   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname
7826 }

```

\showglouservi \showglouservi{<label>}

```

7827 \newcommand*{\showglouservi}[1]{%
7828   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7829 }

```

\showgloname \showgloname{<label>}

```

7830 \newcommand*{\showgloname}[1]{%
7831   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7832 }

```

\showglodesc \showglodesc{<label>}


```

7833 \newcommand*{\showglodesc}[1]{%
7834   \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7835 }

```

showglodescplural `\showglodescplural{<label>}`

```

7836 \newcommand*{\showglodescplural}[1]{%
7837   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7838 }

```

\showglosort `\showglosort{<label>}`

```

7839 \newcommand*{\showglosort}[1]{%
7840   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7841 }

```

\showglosymbol `\showglosymbol{<label>}`

```

7842 \newcommand*{\showglosymbol}[1]{%
7843   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7844 }

```

showglosymbolplural `\showglosymbolplural{<label>}`

```

7845 \newcommand*{\showglosymbolplural}[1]{%
7846   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7847 }

```

\showgloshort `\showgloshort{<label>}`

```

7848 \newcommand*{\showgloshort}[1]{%
7849   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7850 }

```

\showglolong `\showglolong{<label>}`

```

7851 \newcommand*{\showglolong}[1]{%
7852   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7853 }

```

\showgloindex \showgloindex{<label>}

```

7854 \newcommand*{\showgloindex}[1]{%
7855   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7856 }

```

\showgloflag \showgloflag{<label>}

```

7857 \newcommand*{\showgloflag}[1]{%
7858   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7859 }

```

\showgloloclist \showgloloclist{<label>}

```

7860 \newcommand*{\showgloloclist}[1]{%
7861   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7862 }

```

\showglofield \showglofield{<label>}{<field>}

```

7863 \newcommand*{\showglofield}[2]{%
7864   \csshow{glo@\glsdetoklabel{#1}@#2}%
7865 }

```

showacronymlists \showacronymlists

Show list of glossaries that have been flagged as a list of acronyms.

```

7866 \newcommand*{\showacronymlists}{%
7867   \show\@glsacronymlists
7868 }

```

\showglossaries \showglossaries

Show list of defined glossaries.

```

7869 \newcommand*{\showglossaries}{%

```

```
7870 \show\@glo@types
7871 }
```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```
7872 \newcommand*{\showglossaryin}[1]{%
7873 \expandafter\show\csname @glotype@#1@in\endcsname
7874 }
```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```
7875 \newcommand*{\showglossaryout}[1]{%
7876 \expandafter\show\csname @glotype@#1@out\endcsname
7877 }
```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```
7878 \newcommand*{\showglossarytitle}[1]{%
7879 \expandafter\show\csname @glotype@#1@title\endcsname
7880 }
```

`wglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```
7881 \newcommand*{\showglossarycounter}[1]{%
7882 \expandafter\show\csname @glotype@#1@counter\endcsname
7883 }
```

`wglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```
7884 \newcommand*{\showglossaryentries}[1]{%
7885 \expandafter\show\csname glolist@#1\endcsname
7886 }
```

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.

```
7887\csname ifglcompatible-2.07\endcsname
7888 \RequirePackage{glossaries-compatible-207}
7889\fi
```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`\gls{<label>}`” on first use but use “`\an \gls{<label>}`” on subsequent use.

```
7890 \NeedsTeXFormat{LaTeX2e}
```

```
7891 \ProvidesPackage{glossaries-prefix}[2018/05/10 v4.38 (NLCT)]
```

Pass all options to glossaries:

```
7892 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7893 \ProcessOptions
```

Load glossaries:

```
7894 \RequirePackage{glossaries}
```

Add the new keys:

```
7895 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7896 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7897 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7898 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7899 \appto\@gls@keymap{,%
```

```
7900   {prefixfirst}{prefixfirst},%
```

```
7901   {prefixfirstplural}{prefixfirstplural},%
```

```
7902   {prefix}{prefix},%
```

```
7903   {prefixplural}{prefixplural}}%
```

```
7904 }
```

Set the default values:

```
7905 \appto\@newglossaryentryprehook{%
```

```
7906   \def\@glo@entryprefix{}
```

```
7907   \def\@glo@entryprefixplural{}
```

```
7908   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7909   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7910 }
```

Set the assignment code:

```
7911 \appto\@newglossaryentryposthook{%
```

```
7912   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}%
```

```
7913   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7914 \expandafter\gls@assign@field\expandafter
```

```
7915   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}%
```

```
7916   {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7917 \expandafter\gls@assign@field\expandafter
7918   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7919   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7920 }

```

Define commands to access these fields:

entryprefixfirst

```

7921 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7922 \newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7923 \newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7924 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7925 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
7926   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7927   \xmakefirstuc\@glo@text
7928 }

```

entryfirstplural

```

7929 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7930   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7931   \xmakefirstuc\@glo@text
7932 }

```

\Glsentryprefix

```

7933 \newrobustcmd*{\Glsentryprefix}[1]{%
7934   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7935   \xmakefirstuc\@glo@text
7936 }

```

entryprefixplural

```

7937 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7938   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7939   \xmakefirstuc\@glo@text
7940 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7941 \newcommand*{\ifglshasprefix}[3]{%  
7942   \ifcempty{glo@#1@prefix}%  
7943   {#3}%  
7944   {#2}%  
7945 }
```

hasprefixplural

```
7946 \newcommand*{\ifglshasprefixplural}[3]{%  
7947   \ifcempty{glo@#1@prefixplural}%  
7948   {#3}%  
7949   {#2}%  
7950 }
```

shasprefixfirst

```
7951 \newcommand*{\ifglshasprefixfirst}[3]{%  
7952   \ifcempty{glo@#1@prefixfirst}%  
7953   {#3}%  
7954   {#2}%  
7955 }
```

efixfirstplural

```
7956 \newcommand*{\ifglshasprefixfirstplural}[3]{%  
7957   \ifcempty{glo@#1@prefixfirstplural}%  
7958   {#3}%  
7959   {#2}%  
7960 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7961 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7962 \newcommand*{\@pgls}[2][ ]{%  
7963   \new@ifnextchar[%  
7964   {\@pgls@{#1}{#2}}%  
7965   {\@pgls@{#1}{#2}[ ]}%  
7966 }
```

\@pgls@ Read in the final optional argument:

```
7967 \def\@pgls@#1#2[#3]{%  
7968   \glsdoifexists{#2}%  
7969   {%  
7970     \ifglsused{#2}%  
7971     {%  
7972       \glstryprefix{#2}%  
7973     }%  
7974   }
```

```

7974     {%
7975     \glsentryprefixfirst{#2}%
7976     }%
7977     \@gls@{#1}{#2}[#3]%
7978     }%
7979 }

```

Similarly for the plural version:

```

\pglsp1
7980 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7981 \newcommand*{\@pglsp1}[2][ ]{%
7982   \new@ifnextchar[%
7983     {\@pglsp1@{#1}{#2}}%
7984     {\@pglsp1@{#1}{#2}[ ]}%
7985 }

```

\@pglsp1@ Read in the final optional argument:

```

7986 \def\@pglsp1@#1#2[#3]{%
7987   \glsdoifexists{#2}%
7988   {%
7989     \ifglsused{#2}%
7990     {%
7991       \glsentryprefixplural{#2}%
7992     }%
7993     {%
7994       \glsentryprefixfirstplural{#2}%
7995     }%
7996     \@glspl@{#1}{#2}[#3]%
7997   }%
7998 }

```

Now for the first letter upper case versions:

```

\Pgls
7999 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

8000 \newcommand*{\@Pgls}[2][ ]{%
8001   \new@ifnextchar[%
8002     {\@Pgls@{#1}{#2}}%
8003     {\@Pgls@{#1}{#2}[ ]}%
8004 }

```

\@Pgls@ Read in the final optional argument:

```

8005 \def\@Pgls@#1#2[#3]{%

```



```

8006 \glsdoifexists{#2}%
8007 {%
8008   \ifglsused{#2}%
8009   {%
8010     \ifglshasprefix{#2}%
8011     {%
8012       \Glsentryprefix{#2}%
8013       \@gls@{#1}{#2}[#3]%
8014     }%
8015     {\@Gls@{#1}{#2}[#3]}%
8016   }%
8017   {%
8018     \ifglshasprefixfirst{#2}%
8019     {%
8020       \Glsentryprefixfirst{#2}%
8021       \@gls@{#1}{#2}[#3]%
8022     }%
8023     {\@Gls@{#1}{#2}[#3]}%
8024   }%
8025 }%
8026 }

```

Similarly for the plural version:

```

\Pglspl
8027 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

8028 \newcommand*{\@Pglspl}[2] [] {%
8029   \new@ifnextchar[%
8030   {\@Pglspl@{#1}{#2}}%
8031   {\@Pglspl@{#1}{#2} []}%
8032 }

```

\@Pglspl@ Read in the final optional argument:

```

8033 \def\@Pglspl@#1#2[#3] {%
8034   \glsdoifexists{#2}%
8035   {%
8036     \ifglsused{#2}%
8037     {%
8038       \ifglshasprefixplural{#2}%
8039       {%
8040         \Glsentryprefixplural{#2}%
8041         \@glspl@{#1}{#2}[#3]%
8042       }%
8043       {\@Glspl@{#1}{#2}[#3]}%
8044     }%
8045     {%
8046       \ifglshasprefixfirstplural{#2}%

```

```

8047      {%
8048      \Glsentryprefixfirstplural{#2}%
8049      \@glsp1@{#1}{#2}[#3]%
8050      }%
8051      {\@Glspl@{#1}{#2}[#3]}%
8052      }%
8053  }%
8054 }

```

Finally the all upper case versions:

\PGLS

```

8055 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

8056 \newcommand*{\@PGLS}[2][{}]{%
8057   \new@ifnextchar[%
8058   {\@PGLS@{#1}{#2}}%
8059   {\@PGLS@{#1}{#2}[]}%
8060 }

```

\@PGLS@ Read in the final optional argument:

```

8061 \def\@PGLS@#1#2[#3]{%
8062   \glsdoifexists{#2}%
8063   {%
8064     \ifglsused{#2}%
8065     {%
8066       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
8067     }%
8068     {%
8069       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
8070     }%
8071     \@GLS@{#1}{#2}[#3]%
8072   }%
8073 }

```

Plural version:

\PGLSp1

```

8074 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

8075 \newcommand*{\@PGLSp1}[2][{}]{%
8076   \new@ifnextchar[%
8077   {\@PGLSp1@{#1}{#2}}%
8078   {\@PGLSp1@{#1}{#2}[]}%
8079 }

```

\@PGLSp1@ Read in the final optional argument:

```
8080 \def\@PGLSp1@#1#2[#3]{%
8081   \glsdoifexists{#2}%
8082   {%
8083     \ifglsused{#2}%
8084     {%
8085       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
8086     }%
8087     {%
8088       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
8089     }%
8090     \@GLSp1@{#1}{#2}[#3]%
8091   }%
8092 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8093 \ProvidesPackage{glossary-hypernav}[2018/05/10 v4.38 (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`

```
8094 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8095   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8096   \@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.

```
8097 \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`

```
8098 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
8099   \glsnavhypertarget{#1}{#2}{#3}%
8100 }
```

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`

```
8101 \newcommand*{\@glsnavhypertarget}[3]{%}
```

Add this group to the aux file for re-run check.

```
8102 \protected@write\auxout{}\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
8103 \glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8104 \expandafter\let
```

```
8105 \expandafter\@gls@list\csname @gls@hypergroup@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8106 \@for\@gls@elem:=\@gls@list\do{%
```

```
8107 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
8108 \if@endfor
```

```
8109 \else
```

This group was not included in the list, so issue a warning.

```
8110 \GlossariesWarningNoLine{Navigation panel
```

```
8111 for glossary type ‘#1’~Jmissing group ‘#2’}%
```

```
8112 \gdef\gls@hypergroup@rerun{%
```

```
8113 \GlossariesWarningNoLine{Navigation panel
```

```
8114 has changed. Rerun LaTeX}}%
```

```
8115 \fi
```

```
8116 }
```

`\hypergroup@rerun` Give a warning at the end if re-run required

```
8117 \let\gls@hypergroup@rerun\relax
```

```
8118 \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.

```
8119 \newcommand*{\@gls@hypergroup}[2]{%
```

```
8120 \ifundefined{\@gls@hypergroup@list@#1}{%
```

```
8121 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}%
```

```
8122 }{%
```

```
8123 \expandafter\let\expandafter\@gls@tmp
```

```
8124 \csname @gls@hypergroup@list@#1\endcsname
```

```
8125 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
```

```
8126 \@gls@tmp,#2}%
```

```
8127 }%
```

```
8128 }
```

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`

```
8129 \newcommand*{\glsnavigation}{%
8130   \def\@gls@between{}%
8131   \ifcsundef{\@gls@hypergroupplist@\@glo@type}%
8132     {%
8133       \def\@gls@list{}%
8134     }%
8135     {%
8136       \expandafter\let\expandafter\@gls@list
8137         \csname \@gls@hypergroupplist@\@glo@type\endcsname
8138     }%
8139     \@for\@gls@tmp:=\@gls@list\do{%
8140       \@gls@between

8141       \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8142       \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8143       \let\@gls@between\glshypernavsep
8144     }%
8145 }
```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8146 \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`

```
8147 \newcommand*{\glssymbolnav}{%
8148   \glsnavhyperlink{glssymbols}{\@gls@getgrouptitle{glssymbols}}%
8149   \glshypernavsep
8150   \glsnavhyperlink{glsnumbers}{\@gls@getgrouptitle{glsnumbers}}%
8151   \glshypernavsep
8152 }
```

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.

```
8153 \ProvidesPackage{glossary-inline}[2018/05/10 v4.38 (NLCT)]
```

`inline` Define the inline style.

```
8154 \newglossarystyle{inline}{%
      Start of glossary sets up first empty separator between entries. (This is then changed by
      \glossentry)
8155   \renewenvironment{theglossary}%
8156     {%
```

```

8157      \def\gls@inlinesep{}%
8158      \def\gls@inlinesubsep{}%
8159      \def\gls@inlinepostchild{}%
8160      }%
8161      {\glspostinline}%

```

No header:

```

8162 \renewcommand*{\glossaryheader}{}%

```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```

8163 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

8164 \renewcommand{\glossentry}[2]{%
8165   \glsinlinedopostchild
8166   \gls@inlinesep
8167   \glsentryitem{##1}%
8168   \glsinlinenameformat{##1}{%
8169     \glossentryname{##1}%
8170   }%
8171   \ifglshasdescsuppressed{##1}%
8172   {%
8173     \glsinlineemptydescformat
8174     {%
8175       \glossentrysymbol{##1}%
8176     }%
8177     {%
8178       ##2%
8179     }%
8180   }%
8181   {%
8182     \ifglshasdesc{##1}%
8183     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8184     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8185   }%
8186   \ifglshaschildren{##1}%
8187   {%
8188     \glsresetsubentrycounter
8189     \glsinlineparentchildseparator
8190     \def\gls@inlinesubsep{}%
8191     \def\gls@inlinepostchild{\glsinlinepostchild}%
8192   }%
8193   {}%
8194   \def\gls@inlinesep{\glsinlineseparator}%
8195 }%

```

Sub-entries display description:

```

8196 \renewcommand{\subglossentry}[3]{%
8197   \gls@inlinesubsep%
8198   \glsinlinesubnameformat{##2}{%

```

```

8199      \glossentryname{##2}}}%
8200      \glssubentryitem{##2}%
8201      \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8202      \def\gls@inlinesubsep{\glsinlinesubseparator}%
8203  }%

```

Nothing special between groups:

```

8204  \renewcommand*{\glsgroupskip}{}%
8205 }

```

linedopostchild

```

8206 \newcommand*{\glsinlinedopostchild}{%
8207     \gls@inlinepostchild
8208     \def\gls@inlinepostchild{}}%
8209 }

```

inlineseparator Separator to use between entries.

```

8210 \newcommand*{\glsinlineseparator}{;\space}

```

inlinesubseparator Separator to use between sub-entries.

```

8211 \newcommand*{\glsinlinesubseparator}{,\space}

```

parentchildseparator Separator to use between parent and children.

```

8212 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

inlinepostchild Hook to use between child and next entry

```

8213 \newcommand*{\glsinlinepostchild}{}

```

\glspostinline Terminator for inline glossary.

```

8214 \newcommand*{\glspostinline}{\glspostdescription\space}

```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```

8215 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

inlinedescformat Formats the entry's description, symbol and location list:

```

8216 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```

8217 \newcommand*{\glsinlineemptydescformat}[2]{\space#1}

```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```

8218 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{#2}}

```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```

8219 \newcommand*{\glsinlinesubdescformat}[3]{\space#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.

8220 \ProvidesPackage{glossary-list}[2018/05/10 v4.38 (NLCT)]

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8221 \providecommand{\indexspace}{%
8222   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8223 }
```

`tgrouphaderfmt` Provide a way of adjusting the format of the group headings.

```
8224 \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.

```
8225 \newcommand*{\glslistnavigationitem}[1]{\item[#1]}
```

`list` The list glossary style uses the description environment. The group separator `\glsgrpskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the glossaries package.

```
8226 \newglossarystyle{list}{%
```

Use description environment:

```
8227   \renewenvironment{theglossary}%
8228     {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8229   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8230   \renewcommand*{\glsgrpskip}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8231   \renewcommand*{\glossentry}[2]{%
8232     \item[\glsentryitem{##1}%
8233       \glsentrydesc{##1}\glsentryname{##1}]
8234     \glsentrydesc{##1}\glsentryname{##1}\space ##2}%
```

Sub-entries continue on the same line:

```
8235   \renewcommand*{\subglossentry}[3]{%
8236     \glssubentryitem{##2}%
```

```

8237 \glstarget{##2}{\strut}\space
8238 \glossentrydesc{##2}\glspostdescription\space ##3.}%

Add vertical space between groups:

8239 \renewcommand*{\glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
8240 }

```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```

8241 \newglossarystyle{listgroup}{%
Base it on the list style:
8242 \setglossarystyle{list}%

Each group has a heading:
8243 \renewcommand*{\glsgroupheading}[1]{%
8244 \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.

```

8245 \newglossarystyle{listhypergroup}{%
Base it on the list style:
8246 \setglossarystyle{list}%

Add navigation links at the start of the environment.
8247 \renewcommand*{\glossaryheader}{%
8248 \glslistnavigationitem{\glslnavigation}}%

Each group has a heading with a hypertarget:
8249 \renewcommand*{\glsgroupheading}[1]{%
8250 \item[\glslistgroupheaderfmt
8251 {\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.

```

8252 \newglossarystyle{altlist}{%
Base it on the list style:
8253 \setglossarystyle{list}%

Main (level 0) entries start a new item in the list with a line break after the entry name:
8254 \renewcommand*{\glossentry}[2]{%
8255 \item[\glsentryitem{##1}%
8256 \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.

```

8257 \mbox{} \par \nobreak \@afterheading
8258 \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries start a new paragraph:

```
8259 \renewcommand{\subglossentry}[3]{%
8260   \par
8261   \glssubentryitem{##2}%
8262   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
8263 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8264 \newglossarystyle{altlistgroup}{%
  Base it on the altlist style:
8265   \setglossarystyle{altlist}%
  Each group has a heading:
8266   \renewcommand*{\glsgroupheading}[1]{%
8267     \item[\glslistgroupheaderfmt{\glsgrouptitle{##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.

```
8268 \newglossarystyle{altlisthypergroup}{%
  Base it on the altlist style:
8269   \setglossarystyle{altlist}%
  Add navigation links at the start of the environment.
8270   \renewcommand*{\glossaryheader}{%
8271     \glslistnavigationitem{\glslnavigation}}%
  Each group has a heading with a hypertarget:
8272   \renewcommand*{\glsgroupheading}[1]{%
8273     \item[\glslistgroupheaderfmt
8274       {\glslnavhypertarget{##1}{\glsgrouptitle{##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.

```
8275 \newglossarystyle{listdotted}{%
  Base it on the list style:
8276   \setglossarystyle{list}%
  Each main (level 0) entry starts a new item:
8277   \renewcommand*{\glossentry}[2]{%
8278     \item[]\makebox[\glslistdottedwidth][l]{%
8279       \glssentryitem{##1}%
8280       \glstarget{##1}{\glossentryname{##1}}%
8281       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```

8282 \renewcommand*{\subglossentry}[3]{%
8283   \item[\makebox[\glslistdottedwidth][l]{%
8284     \glssubentryitem{##2}%
8285     \glstarget{##2}{\glossentryname{##2}}}%
8286   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8287 }
```

`listdottedwidth`

```

8288 \newlength\glslistdottedwidth
8289 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8290 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
8291 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```

8292 \renewcommand*{\glossentry}[2]{%
8293   \item[\glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
8294 }
```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8295 \ProvidesPackage{glossary-long}[2018/05/10 v4.38 (NLCT)]
```

Requires the package:

```
8296 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. (There's a chance that the user may specify `nolong` and then load later, in which case `\glsdescwidth` may have already been defined by . The same goes for `\glspagelistwidth`.)

```

8297 \@ifundefined{glsdescwidth}{%
8298   \newlength\glsdescwidth
8299   \setlength{\glsdescwidth}{0.6\hsize}
8300 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column.

```

8301 \@ifundefined{glspagelistwidth}{%
8302   \newlength\glspagelistwidth
8303   \setlength{\glspagelistwidth}{0.1\hsize}
8304 }{}
```

long The long glossary style command which uses the longtable environment:

```
8305 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
8306 \renewenvironment{theglossary}{%
8307     {\begin{longtable}\lp{\glstdescwidth}}}%
8308     {\end{longtable}}}%
```

Do nothing at the start of the environment:

```
8309 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8310 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8311 \renewcommand{\glossentry}[2]{%
8312     \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8313     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8314 }%
```

Sub entries displayed on the following row without the name:

```
8315 \renewcommand{\subglossentry}[3]{%
8316     &
8317     \glssubentryitem{##2}%
8318     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8319     ##3\tabularnewline
8320 }%
```

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>)

```
8321 \ifglsgroupskip
8322     \renewcommand*{\glsgroupskip}{}%
8323 \else
8324     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8325 \fi
8326 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
8327 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
8328 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
8329 \renewenvironment{theglossary}{%
8330     \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8331 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8332 }
```

longheader The longheader style is like the long style but with a header:

```
8333 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8334 \setglossarystyle{long}{%
```

Set the table's header:

```
8335 \renewcommand*{\glossaryheader}{%
8336 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8337 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8338 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8339 \setglossarystyle{longborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8340 \renewcommand*{\glossaryheader}{%
8341 \hline\bfseries \entryname & \bfseries
8342 \descriptionname\tabularnewline\hline
8343 \endhead
8344 \hline\endfoot}%
8345 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
8346 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
8347 \renewenvironment{theglossary}{%
8348 {\begin{longtable}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8349 {\end{longtable}}}%

```

No table header:

```
8350 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```
8351 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8352 \renewcommand{\glossentry}[2]{%
8353 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8354 \glossentrydesc{##1} & ##2\tabularnewline
8355 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8356 \renewcommand{\subglossentry}[3]{%
8357 &
8358 \glssubentryitem{##2}%
8359 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8360 ##3\tabularnewline
8361 }%

```

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>)

```
8362 \ifglsgroupskip
8363 \renewcommand*{\glsgroupskip}{}%
8364 \else
8365 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8366 \fi
8367 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
8368 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8369 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8370 \renewenvironment{theglossary}%
8371 {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8372 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8373 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8374 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
8375 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
8376 \setglossarystyle{long3col}%
    Set the table's header:
8377 \renewcommand*{\glossaryheader}{%
8378 \bfseries\entryname&\bfseries\descriptionname&
8379 \bfseries\pagelistname\tabularnewline\endhead}%
8380 }
```

colheaderborder The long3colheaderborder style is like the above but with a border

```
8381 \newglossarystyle{long3colheaderborder}{%
    Base it on the glostylelong3colborder style:
8382 \setglossarystyle{long3colborder}%
    Set the table's header and add horizontal line at table's foot:
8383 \renewcommand*{\glossaryheader}{%
8384 \hline
8385 \bfseries\entryname&\bfseries\descriptionname&
8386 \bfseries\pagelistname\tabularnewline\hline\endhead
8387 \hline\endfoot}%
8388 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8389 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8390 \renewenvironment{theglossary}{%
```

```
8391 {\begin{longtable}{llll}}%
```

```
8392 {\end{longtable}}%
```

No table header:

```
8393 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8394 \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):

```
8395 \renewcommand{\glossentry}[2]{%
```

```
8396 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
8397 \glossentrydesc{##1} &
```

```
8398 \glossentrysymbol{##1} &
```

```
8399 ##2\tabularnewline
```

```
8400 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8401 \renewcommand{\subglossentry}[3]{%
```

```
8402 &
```

```
8403 \glssubentryitem{##2}%
```

```
8404 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
8405 \glossentrysymbol{##2} & ##3\tabularnewline
```

```
8406 }%
```

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>)

```
8407 \ifglsgnogroupskip
```

```
8408 \renewcommand*{\glsgroupskip}{}%
```

```
8409 \else
```

```
8410 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
```

```
8411 \fi
```

```
8412 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8413 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8414 \setglossarystyle{long4col}%
```

Table has a header:

```
8415 \renewcommand*{\glossaryheader}{%
```

```
8416 \bfseries\entryname&\bfseries\descriptionname&
```

```
8417 \bfseries \symbolname&
```



```

8418     \bfseries\pagelistname\tabularnewline\endhead}%
8419 }

```

long4colborder The long4colborder style is like long4col but with a border.

```

8420 \newglossarystyle{long4colborder}{%
    Base it on the glostylelong4col style:
8421     \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8422     \renewenvironment{theglossary}%
8423         {\begin{longtable}{|l|l|l|l|}}%
8424         {\end{longtable}}%
    Add horizontal lines to the head and foot of the table:
8425     \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8426 }

```

colheaderborder The long4colheaderborder style is like the above but with a border.

```

8427 \newglossarystyle{long4colheaderborder}{%
    Base it on the glostylelong4col style:
8428     \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8429     \renewenvironment{theglossary}%
8430         {\begin{longtable}{|l|l|l|l|}}%
8431         {\end{longtable}}%
    Add table header and horizontal line at the table's foot:
8432     \renewcommand*{\glossaryheader}{%
8433         \hline\bfseries\entryname&\bfseries\descriptionname&
8434         \bfseries \symbolname&
8435         \bfseries\pagelistname\tabularnewline\hline\endhead
8436         \hline\endfoot}%
8437 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8438 \newglossarystyle{altlong4col}{%
    Base it on the glostylelong4col style:
8439     \setglossarystyle{long4col}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8440     \renewenvironment{theglossary}%
8441         {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8442         {\end{longtable}}%
8443 }

```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```

8444 \newglossarystyle{altlong4colheader}{%
      Base it on the glostylelong4colheader style:
8445   \setglossarystyle{long4colheader}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8446   \renewenvironment{theglossary}%
8447     {\begin{longtable}{\lp{\glsgdescwidth}\lp{\glspagelistwidth}}}%
8448     {\end{longtable}}}%
8449 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```

8450 \newglossarystyle{altlong4colborder}{%
      Base it on the glostylelong4colborder style:
8451   \setglossarystyle{long4colborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8452   \renewenvironment{theglossary}%
8453     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8454     {\end{longtable}}}%
8455 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```

8456 \newglossarystyle{altlong4colheaderborder}{%
      Base it on the glostylelong4colheaderborder style:
8457   \setglossarystyle{long4colheaderborder}%
      Use a longtable with 4 columns where the second and last columns may have multiple lines
      in each row:
8458   \renewenvironment{theglossary}%
8459     {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
8460     {\end{longtable}}}%
8461 }
```

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>

```

8462 \ProvidesPackage{glossary-longbooktabs}[2018/05/10 v4.38 (NLCT)]
```

Requires `booktabs` package:

```

8463 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8464 \RequirePackage{glossary-long}
8465 \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.

```
8466 \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.

```
8467 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8468 \setglossarystyle{long}{%
```

Add a header with rules.

```
8469 \renewcommand*{\glossaryheader}{%
8470 \toprule \bfseries \entryname & \bfseries
8471 \descriptionname\tabularnewline\midrule\endhead
8472 \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.

```
8473 \ifglsgroupskip
8474 \renewcommand*{\glsgroupskip}{}%
8475 \else
8476 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8477 \fi
8478 }
```

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.

```
8479 \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.

```
8480 \glspatchLToutput
```

Use the long3col style as a base.

```
8481 \setglossarystyle{long3col}{%
```

Add a header with rules.

```
8482 \renewcommand*{\glossaryheader}{%
8483 \toprule \bfseries \entryname &
8484 \bfseries \descriptionname &
8485 \bfseries \pagelistname
8486 \tabularnewline\midrule\endhead
8487 \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.

```
8488 \ifglsgroupskip
8489 \renewcommand*{\glsgroupskip}{}%
8490 \else
8491 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8492 \fi
8493 }
```

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.

```
8494 \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.

```
8495 \glspatchLToutput
```

Use the long4col style as a base.

```
8496 \setglossarystyle{long4col}{%
```

Add a header with rules.

```
8497 \renewcommand*{\glossaryheader}{%
8498 \toprule \bfseries \entryname &
8499 \bfseries \descriptionname &
8500 \bfseries \symbolname &
8501 \bfseries \pagelistname
8502 \tabularnewline\midrule\endhead
8503 \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.

```
8504 \ifglsgroupskip
8505 \renewcommand*{\glsgroupskip}{}%
8506 \else
8507 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8508 \fi
8509 }
```

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.

```
8510 \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.

```
8511 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8512 \setglossarystyle{long4col-booktabs}{%
```

Change the column specifications:

```
8513 \renewenvironment{theglossary}%  
8514   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%  
8515   {\end{longtable}}}%  
8516 }
```

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.

```
8517 \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.

```
8518 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8519 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8520 \renewenvironment{theglossary}%  
8521   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%  
8522   {\end{longtable}}}%  
8523 }
```

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.

```
8524 \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.

```
8525 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8526 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8527 \renewenvironment{theglossary}%  
8528   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%  
8529   >{\raggedright}p{\glspagelistwidth}}}%  
8530   {\end{longtable}}}%  
8531 }
```

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.

```
8532 \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.

```
8533 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8534 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8535 \renewenvironment{theglossary}%  
8536 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8537 >{\raggedright}p{\glspagelistwidth}}}%  
8538 {\end{longtable}}%  
8539 }
```

sLTpenaltycheck

```
8540 \newcommand*{\glslTpenaltycheck}{%  
8541 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8542 }
```

enaltygroupskip

```
8543 \newcommand{\glspenaltygroupskip}{%  
8544 \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8545 \let\@gls@org@LT@output\LT@output  
8546 \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

```
8547 \newcommand*{\glspatchLToutput}{%  
8548 \renewcommand*{\LT@output}{%  
8549 \ifnum\outputpenalty < -\@Mi  
8550 \ifnum\outputpenalty > -\LT@end@pen  
8551 \LT@err{floats and marginpars not allowed in a longtable}\@ehc  
8552 \else  
8553 \setbox\z@\vbox{\unvbox\@cclv}%  
8554 \ifdim \ht\LT@lastfoot>\ht\LT@foot  
8555 \dimen@pagegoal  
8556 \advance\dimen@-\ht\LT@lastfoot  
8557 \ifdim\dimen@<\ht\z@  
8558 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%  
8559 \@makecol  
8560 \@outputpage  
8561 \setbox\z@\vbox{\box\LT@head\glslTpenaltycheck}%  
8562 \fi  
8563 \fi  
8564 \global\@colroom\@colht  
8565 \global\vsize\@colht  
8566 {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%  
8567 \fi  
8568 \else
```

```

8569 \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8570 \@makecol
8571 \@outputpage
8572 \global\ysize\@colroom
8573 \copy\LT@head
8574 \glsLTpenaltycheck
8575 \nobreak
8576 \fi
8577 }%
8578 }

```

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.

```
8579 \ProvidesPackage{glossary-longragged}[2018/05/10 v4.38 (NLCT)]
```

Requires the package:

```
8580 \RequirePackage{array}
```

Requires the package:

```
8581 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8582 \@ifundefined{glsdescwidth}{%
8583 \newlength\glsdescwidth
8584 \setlength{\glsdescwidth}{0.6\hsize}
8585 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8586 \@ifundefined{glspagelistwidth}{%
8587 \newlength\glspagelistwidth
8588 \setlength{\glspagelistwidth}{0.1\hsize}
8589 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8590 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8591 \renewenvironment{theglossary}%
8592 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
8593 {\end{longtable}}%

```

Do nothing at the start of the environment:

```
8594 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8595 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8596 \renewcommand{\glossentry}[2]{%
8597   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8598   \glossentrydesc{##1}\glspostdescription\space ##2%
8599   \tabularnewline
8600 }%
```

Sub entries displayed on the following row without the name:

```
8601 \renewcommand{\subglossentry}[3]{%
8602   &
8603   \glssubentryitem{##2}%
8604   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8605   \glspostdescription\space ##3%
8606   \tabularnewline
8607 }%
```

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>)

```
8608 \ifglsgroupskip
8609 \renewcommand*{\glsgroupskip}{}%
8610 \else
8611 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8612 \fi
8613 }
```

`ongraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8614 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8615 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8616 \renewenvironment{theglossary}{%
8617   \begin{longtable}{|l|>{\raggedright}p{\glsglwidth}|}%
8618   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8619 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8620 }
```

`ongraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8621 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8622 \setglossarystyle{longragged}%
```

Set the table's header:

```
8623 \renewcommand*{\glossaryheader}{%
8624   \bfseries \entryname & \bfseries \descriptionname
```



```

8625 \tabularnewline\endhead}%
8626 }

```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```

8627 \newglossarystyle{longraggedheaderborder}{%
      Base it on the glostylelongraggedborder style:
8628 \setglossarystyle{longraggedborder}%
      Set the table's header and add horizontal line to table's foot:
8629 \renewcommand*{\glossaryheader}{%
8630 \hline\bfseries \entryname & \bfseries \descriptionname
8631 \tabularnewline\hline
8632 \endhead
8633 \hline\endfoot}%
8634 }

```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```

8635 \newglossarystyle{longragged3col}{%
      Use a longtable with 3 columns:
8636 \renewenvironment{theglossary}%
8637 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8638 >{\raggedright}p{\glspagelistwidth}}}%
8639 {\end{longtable}}%

```

No table header:

```

8640 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```

8641 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8642 \renewcommand{\glossentry}[2]{%
8643 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8644 \glossentrydesc{##1} & ##2\tabularnewline
8645 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8646 \renewcommand{\subglossentry}[3]{%
8647 &
8648 \glssubentryitem{##2}%
8649 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8650 ##3\tabularnewline
8651 }%

```

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>)

```

8652 \ifglsnogroupskip
8653 \renewcommand*{\glsgroupskip}{}%

```

```

8654 \else
8655 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8656 \fi
8657 }

```

ragged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```

8658 \newglossarystyle{longragged3colborder}{%
    Base it on the glostylelongragged3col style:
8659 \setglossarystyle{longragged3col}%
    Use a longtable with 3 columns with vertical lines around them:
8660 \renewenvironment{theglossary}%
8661 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%
8662 >{\raggedright}p{\glspagelistwidth}|}%
8663 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8664 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8665 }

```

ragged3colheader The longragged3colheader style is like longragged3col but with a header row:

```

8666 \newglossarystyle{longragged3colheader}{%
    Base it on the glostylelongragged3col style:
8667 \setglossarystyle{longragged3col}%
    Set the table's header:
8668 \renewcommand*{\glossaryheader}{%
8669 \bfseries\entryname&\bfseries\descriptionname&
8670 \bfseries\pagelistname\tabularnewline\endhead}%
8671 }

```

colheaderborder The longragged3colheaderborder style is like the above but with a border

```

8672 \newglossarystyle{longragged3colheaderborder}{%
    Base it on the glostylelongragged3colborder style:
8673 \setglossarystyle{longragged3colborder}%
    Set the table's header and add horizontal line at table's foot:
8674 \renewcommand*{\glossaryheader}{%
8675 \hline
8676 \bfseries\entryname&\bfseries\descriptionname&
8677 \bfseries\pagelistname\tabularnewline\hline\endhead
8678 \hline\endfoot}%
8679 }

```

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.

```

8680 \newglossarystyle{altlongragged4col}{%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8681 \renewenvironment{theglossary}%
8682   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8683     >{\raggedright}p{\glspagelistwidth}}}%
8684   {\end{longtable}}%
```

No table header:

```
8685 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8686 \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):

```
8687 \renewcommand{\glossentry}[2]{%
8688   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8689   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8690   ##2\tabularnewline
8691 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8692 \renewcommand{\subglossentry}[3]{%
8693   &
8694   \glssubentryitem{##2}%
8695   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8696   \glossentrysymbol{##2} & ##3\tabularnewline
8697 }%
```

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>)

```
8698 \ifglsgroupskip
8699   \renewcommand*{\glsgroupskip}{}%
8700 \else
8701   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8702 \fi
8703 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8704 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8705 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8706 \renewenvironment{theglossary}%
8707   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8708     >{\raggedright}p{\glspagelistwidth}}}%
8709   {\end{longtable}}%
```

Table has a header:

```
8710 \renewcommand*{\glossaryheader}{%
8711 \bfseries\entryname&\bfseries\descriptionname&
8712 \bfseries \symbolname&
8713 \bfseries\pagelistname\tabularnewline\endhead}%
8714 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8715 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8716 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8717 \renewenvironment{theglossary}%
8718 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8719 >\raggedright}p{\glspagelistwidth}|}}%
8720 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8721 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8722 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8723 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8724 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8725 \renewenvironment{theglossary}%
8726 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8727 >\raggedright}p{\glspagelistwidth}|}}%
8728 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8729 \renewcommand*{\glossaryheader}{%
8730 \hline\bfseries\entryname&\bfseries\descriptionname&
8731 \bfseries \symbolname&
8732 \bfseries\pagelistname\tabularnewline\hline\endhead
8733 \hline\endfoot}%
8734 }
```

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.

```
8735 \ProvidesPackage{glossary-mcols}[2018/05/10 v4.38 (NLCT)]
```

Required packages:

```
8736 \RequirePackage{multicol}
8737 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8738 \providecommand{\indexspace}{%
8739   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8740 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8741 \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.)

```
8742 \newglossarystyle{mcolindex}{%
8743   \setglossarystyle{index}%
8744   \renewenvironment{theglossary}%
8745     {%
8746       \begin{multicols}{\glsmcols}
8747       \setlength{\parindent}{0pt}%
8748       \setlength{\parskip}{0pt plus 0.3pt}%
8749       \let\item\glstreeitem
8750       \let\subitem\glstreesubitem
8751       \let\subsubitem\glstreesubsubitem
8752     }%
8753     {\end{multicols}}}%
8754 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8755 \newglossarystyle{mcolindexgroup}{%
8756   \setglossarystyle{mcolindex}%
8757   \renewcommand*{\glsgroupheading}[1]{%
8758     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8759 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8760 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8761 \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8762 \renewcommand*{\glossaryheader}{%
8763   \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.

```
8764 \renewcommand*{\glsgroupheading}[1]{%
8765   \item\glstreegroupheaderfmt
8766     {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8767   \indexspace}%
8768 }
```

colindexspannav Similar to **colindexhypergroup**, but puts the navigation line in the optional argument of **multicols**.

```
8769 \newglossarystyle{colindexspannav}{%
8770   \setglossarystyle{index}%
8771   \renewenvironment{theglossary}%
8772     {%
8773       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8774       \setlength{\parindent}{0pt}%
8775       \setlength{\parskip}{0pt plus 0.3pt}%
8776       \let\item\glstreeitem}%
8777     {\end{multicols}}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8778 \renewcommand*{\glsgroupheading}[1]{%
8779   \item\glstreegroupheaderfmt
8780     {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8781   \indexspace}%
8782 }
```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8783 \newglossarystyle{mcoltree}{%
8784   \setglossarystyle{tree}%
8785   \renewenvironment{theglossary}%
8786     {%
8787       \begin{multicols}{\glsmcols}
8788       \setlength{\parindent}{0pt}%
8789       \setlength{\parskip}{0pt plus 0.3pt}%
8790     }%
8791     {\end{multicols}}%
8792 }
```

mcoltreegroup Like the **mcoltree** style but the glossary groups have headings.

```
8793 \newglossarystyle{mcoltreegroup}{%
  Base it on the glostylemcoltree style:
8794   \setglossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
8795 \renewcommand{\glsgroupheading}[1]{\par
8796 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8797 }
```

ltreehypergroup The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8798 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the `glostylemcoltree` style:

```
8799 \setglossarystyle{mcoltree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8800 \renewcommand*{\glossaryheader}{%
8801 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8802 \renewcommand*{\glsgroupheading}[1]{%
8803 \par\noindent
8804 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8805 \indexspace}%
8806 }
```

mcoltreespannav Similar to the `mcoltreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8807 \newglossarystyle{mcoltreespannav}{%
8808 \setglossarystyle{tree}%
8809 \renewenvironment{theglossary}%
8810 {%
8811 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8812 \setlength{\parindent}{0pt}%
8813 \setlength{\parskip}{0pt plus 0.3pt}%
8814 }%
8815 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8816 \renewcommand*{\glsgroupheading}[1]{%
8817 \par\noindent
8818 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8819 \indexspace}%
8820 }
```

mcoltreenoname Multi-column index style. Same as the `treenoname`, but puts the glossary in multiple columns.

```
8821 \newglossarystyle{mcoltreenoname}{%
8822 \setglossarystyle{treenoname}%
8823 \renewenvironment{theglossary}%
8824 {%
```

```

8825     \begin{multicols}{\glsmcols}
8826     \setlength{\parindent}{0pt}%
8827     \setlength{\parskip}{0pt plus 0.3pt}%
8828 }%
8829 {\end{multicols}}%
8830 }

```

treenonamegroup Like the `mcoltreenoname` style but the glossary groups have headings.

```

8831 \newglossarystyle{mcoltreenonamegroup}{%
    Base it on the glostylemcoltreenoname style:
8832 \setglossarystyle{mcoltreenoname}%
    Give each group a heading:
8833 \renewcommand{\glsgroupheading}[1]{\par
8834 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8835 }

```

onamehypergroup The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

8836 \newglossarystyle{mcoltreenonamehypergroup}{%
    Base it on the glostylemcoltreenoname style:
8837 \setglossarystyle{mcoltreenoname}%
    Put navigation links to the groups at the start of the theglossary environment:
8838 \renewcommand*{\glossaryheader}{%
8839 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8840 \renewcommand*{\glsgroupheading}[1]{%
8841 \par\noindent
8842 \glstreegroupheaderfmt{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
8843 \indexspace}%
8844 }

```

eenonamespannav Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8845 \newglossarystyle{mcoltreenonamespannav}{%
8846 \setglossarystyle{treenoname}%
8847 \renewenvironment{theglossary}%
8848 {%
8849 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8850 \setlength{\parindent}{0pt}%
8851 \setlength{\parskip}{0pt plus 0.3pt}%
8852 }%
8853 {\end{multicols}}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8854 \renewcommand*{\glsgroupheading}[1]{%
8855 \par\noindent

```



```

8856 \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgrouptitle{##1}}}\par
8857 \indexspace}%
8858 }

```

mcolalttree Multi-column index style. Same as the alttree, but puts the glossary in multiple columns.

```

8859 \newglossarystyle{mcolalttree}{%
8860 \setglossarystyle{alttree}%
8861 \renewenvironment{theglossary}%
8862 {%
8863 \begin{multicols}{\glsmcols}
8864 \def\@gls@prevlevel{-1}%
8865 \mbox{}\par
8866 }%
8867 {\par\end{multicols}}}%
8868 }

```

colalttreegroup Like the mcolalttree style but the glossary groups have headings.

```

8869 \newglossarystyle{colalttreegroup}{%
      Base it on the glostylemcolalttree style:
8870 \setglossarystyle{mcolalttree}%
      Give each group a heading.
8871 \renewcommand{\glsgroupheading}[1]{\par
8872 \def\@gls@prevlevel{-1}%
8873 \hangindent0pt\relax
8874 \parindent0pt\relax
8875 \glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8876 }

```

treehypergroup The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```

8877 \newglossarystyle{mcolalttreehypergroup}{%
      Base it on the glostylemcolalttree style:
8878 \setglossarystyle{mcolalttree}%
      Put the navigation links in the header
8879 \renewcommand*{\glossaryheader}{%
8880 \par
8881 \def\@gls@prevlevel{-1}%
8882 \hangindent0pt\relax
8883 \parindent0pt\relax
8884 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8885 \renewcommand*{\glsgroupheading}[1]{%
8886 \par
8887 \def\@gls@prevlevel{-1}%
8888 \hangindent0pt\relax

```

```

8889 \parindent0pt\relax
8890 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8891 \indexspace}%
8892 }

```

`lalttreespannav` Similar to the `mcolalttreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8893 \newglossarystyle{mcolalttreespannav}{%
8894 \setglossarystyle{alttree}%
8895 \renewenvironment{theglossary}%
8896 {%
8897 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8898 \def\@gls@prevlevel{-1}%
8899 \mbox{}\par
8900 }%
8901 {\par\end{multicols}}}%

```

Put a hypertarget at the start of each group

```

8902 \renewcommand*{\glsgroupheading}[1]{%
8903 \par
8904 \def\@gls@prevlevel{-1}%
8905 \hangindent0pt\relax
8906 \parindent0pt\relax
8907 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8908 \indexspace}%
8909 }

```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the `supertabular` environment.

```

8910 \ProvidesPackage{glossary-super}[2018/05/10 v4.38 (NLCT)]

```

Requires the package:

```

8911 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if `has` has been loaded.

```

8912 \@ifundefined{glsdescwidth}{%
8913 \newlength{glsdescwidth}
8914 \setlength{glsdescwidth}{0.6\hsize}
8915 }{}

```

`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.

```

8916 \@ifundefined{glspagelistwidth}{%
8917 \newlength{glspagelistwidth}
8918 \setlength{glspagelistwidth}{0.1\hsize}

```

8919 }{}

super The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

8920 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

8921 \renewenvironment{theglossary}{%

8922 {\tablehead{}\tabletail{}}%

8923 \begin{supertabular}{lp{\glsdescwidth}}%

8924 {\end{supertabular}}%

Do nothing at the start of the table:

8925 \renewcommand*{\glossaryheader}{}%

No group headings:

8926 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

8927 \renewcommand{\glossentry}[2]{%

8928 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

8929 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline

8930 }%

Sub entries put in a row (no name, description and page list in second column):

8931 \renewcommand{\subglossentry}[3]{%

8932 &

8933 \glssubentryitem{##2}%

8934 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space

8935 ##3\tabularnewline

8936 }%

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>)

8937 \ifglsnogroupskip

8938 \renewcommand*{\glsgroupskip}{}%

8939 \else

8940 \renewcommand*{\glsgroupskip}{& \tabularnewline}%

8941 \fi

8942 }

superborder The superborder style is like the above, but with horizontal and vertical lines:

8943 \newglossarystyle{superborder}{%

Base it on the glostylesuper style:

8944 \setglossarystyle{super}%

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

8945 \renewenvironment{theglossary}{%

8946 {\tablehead{\hline}\tabletail{\hline}%

```

8947     \begin{supertabular}{|l|p{\glsdescwidth}|}%
8948     {\end{supertabular}}%
8949 }

```

superheader The superheader style is like the super style, but with a header:

```

8950 \newglossarystyle{superheader}{%
      Base it on the glostylesuper style:
8951   \setglossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and no tail:
8952 \renewenvironment{theglossary}%
8953   {\tablehead{\bfseries \entryname &
8954     \bfseries\descriptionname\tabularnewline}%
8955     \tabletail{}}%
8956   \begin{supertabular}{lp{\glsdescwidth}}%
8957   {\end{supertabular}}%
8958 }

```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```

8959 \newglossarystyle{superheaderborder}{%
      Base it on the glostylesuper style:
8960   \setglossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and horizontal
      lines above and below the table:
8961   \renewenvironment{theglossary}%
8962     {\tablehead{\hline\bfseries \entryname &
8963       \bfseries \descriptionname\tabularnewline\hline}%
8964       \tabletail{\hline}}
8965     \begin{supertabular}{|l|p{\glsdescwidth}|}%
8966     {\end{supertabular}}%
8967 }

```

super3col The super3col style is like the super style, but with 3 columns:

```

8968 \newglossarystyle{super3col}{%
      Put the glossary in a supertabular environment with three columns and no head or tail:
8969   \renewenvironment{theglossary}%
8970     {\tablehead{}\tabletail{}}%
8971     \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
8972     {\end{supertabular}}%
      Do nothing at the start of the table:
8973   \renewcommand*{\glossaryheader}{}%
      No group headings:
8974   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8975 \renewcommand{\glossentry}[2]{%
8976   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8977   \glossentrydesc{##1} & ##2\tabularnewline
8978 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

8979 \renewcommand{\subglossentry}[3]{%
8980   &
8981   \glssubentryitem{##2}%
8982   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8983   ##3\tabularnewline
8984 }%

```

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>)

```

8985 \ifglsgroupskip
8986   \renewcommand*{\glsgroupskip}{}%
8987 \else
8988   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8989 \fi
8990 }

```

super3colborder The super3colborder style is like the super3col style, but with a border:

```

8991 \newglossarystyle{super3colborder}{%

```

Base it on the glostylesuper3col style:

```

8992 \setglossarystyle{super3col}%

```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

8993 \renewenvironment{theglossary}%
8994   {\tablehead{\hline}\tabletail{\hline}%
8995   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth|}}%
8996   {\end{supertabular}}%
8997 }

```

super3colheader The super3colheader style is like the super3col style but with a header row:

```

8998 \newglossarystyle{super3colheader}{%

```

Base it on the glostylesuper3col style:

```

8999 \setglossarystyle{super3col}%

```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9000 \renewenvironment{theglossary}%
9001   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9002   \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9003   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
9004   {\end{supertabular}}%
9005 }

```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
9006 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
9007 \setglossarystyle{super3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9008 \renewenvironment{theglossary}{%
9009   {\tablehead{\hline
9010     \bfseries\entryname&\bfseries\descriptionname&
9011     \bfseries\pagelistname\tabularnewline\hline}%
9012   \tabletail{\hline}%
9013   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
9014   {\end{supertabular}}}%
9015 }
```

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.

```
9016 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9017 \renewenvironment{theglossary}{%
9018   {\tablehead{}\tabletail{}}%
9019   \begin{supertabular}{|l|l|l|l|}%
9020   \end{supertabular}}%
```

Do nothing at the start of the table:

```
9021 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9022 \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:

```
9023 \renewcommand{\glossentry}[2]{%
9024   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9025   \glossentrydesc{##1} &
9026   \glossentrysymbol{##1} & ##2\tabularnewline
9027 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9028 \renewcommand{\subglossentry}[3]{%
9029   &
9030   \glssubentryitem{##2}%
9031   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9032   \glossentrysymbol{##2} & ##3\tabularnewline
9033 }%
```

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>)

```
9034 \ifglsgroupskip
9035 \renewcommand*{\glsgroupskip}{}%
9036 \else
9037 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9038 \fi
9039 }
```

super4colheader The super4colheader style is like the super4col but with a header row.

```
9040 \newglossarystyle{super4colheader}{%
    Base it on the glostylesuper4col style:
9041 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns, a header and no tail:
9042 \renewenvironment{theglossary}%
9043 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9044 \bfseries\symbolname &
9045 \bfseries\pagelistname\tabularnewline}%
9046 \tabletail{}}%
9047 \begin{supertabular}{l111}}%
9048 {\end{supertabular}}%
9049 }
```

super4colborder The super4colborder style is like the super4col but with a border.

```
9050 \newglossarystyle{super4colborder}{%
    Base it on the glostylesuper4col style:
9051 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns and a horizontal line in the
    head and tail:
9052 \renewenvironment{theglossary}%
9053 {\tablehead{\hline}\tabletail{\hline}%
9054 \begin{supertabular}{|l11|11|}}%
9055 {\end{supertabular}}%
9056 }
```

colheaderborder The super4colheaderborder style is like the super4col but with a header and border.

```
9057 \newglossarystyle{super4colheaderborder}{%
    Base it on the glostylesuper4col style:
9058 \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:
9059 \renewenvironment{theglossary}%
9060 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
9061 \bfseries\symbolname &
```

```

9062      \bfseries\pagelistname\tabularnewline\hline}%
9063      \tabletail{\hline}%
9064      \begin{supertabular}{|l|l|l|l|}%
9065      {\end{supertabular}}}%
9066 }

```

altsuper4col The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```
9067 \newglossarystyle{altsuper4col}{%
```

Base it on the glostylesuper4col style:

```
9068 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

9069 \renewenvironment{theglossary}%
9070 {\tablehead{}\tabletail{}}%
9071 \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
9072 {\end{supertabular}}}%
9073 }

```

super4colheader The altsuper4colheader style is like the altsuper4col but with a header row.

```
9074 \newglossarystyle{altsuper4colheader}{%
```

Base it on the glostylesuper4colheader style:

```
9075 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

9076 \renewenvironment{theglossary}%
9077 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9078 \bfseries\symbolname &
9079 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9080 \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
9081 {\end{supertabular}}}%
9082 }

```

super4colborder The altsuper4colborder style is like the altsuper4col but with a border.

```
9083 \newglossarystyle{altsuper4colborder}{%
```

Base it on the glostylesuper4colborder style:

```
9084 \setglossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

9085 \renewenvironment{theglossary}%
9086 {\tablehead{\hline}\tabletail{\hline}%
9087 \begin{supertabular}%
9088 {lp{\glsdescwidth}lp{\glspagelistwidth}}%
9089 {\end{supertabular}}}%
9090 }

```

colheaderborder The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```
9091 \newglossarystyle{altsuper4colheaderborder}{%
```


Base it on the `glostylesuper4colheaderborder` style:

```
9092 \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:

```
9093 \renewenvironment{theglossary}%
9094   {\tablehead{\hline
9095     \bfseries\entryname &
9096     \bfseries\descriptionname &
9097     \bfseries\symbolname &
9098     \bfseries\pagelistname\tabularnewline\hline}%
9099   \tabletail{\hline}%
9100   \begin{supertabular}%
9101     {lllp{\glsdescwidth}llp{\glspagelistwidth}}}%
9102   {\end{supertabular}}%
9103 }
```

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.

```
9104 \ProvidesPackage{glossary-superragged}[2018/05/10 v4.38 (NLCT)]
```

Requires the package:

```
9105 \RequirePackage{array}
```

Requires the package:

```
9106 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
9107 \@ifundefined{glsdescwidth}{%
9108   \newlength\glsdescwidth
9109   \setlength{\glsdescwidth}{0.6\hsize}
9110 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
9111 \@ifundefined{glspagelistwidth}{%
9112   \newlength\glspagelistwidth
9113   \setlength{\glspagelistwidth}{0.1\hsize}
9114 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
9115 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9116 \renewenvironment{theglossary}%
9117   {\tablehead{}\tabletail{}}%
9118   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9119   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9120 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9121 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9122 \renewcommand{\glossentry}[2]{%
9123   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9124   \glossentrydesc{##1}\glspostdescription\space ##2%
9125   \tabularnewline
9126 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9127 \renewcommand{\subglossentry}[3]{%
9128   &
9129   \glssubentryitem{##2}%
9130   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9131   ##3%
9132   \tabularnewline
9133 }%
```

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>)

```
9134 \ifglsgroupskip
9135   \renewcommand*{\glsgroupskip}{}%
9136 \else
9137   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9138 \fi
9139 }
```

`\perraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
9140 \newglossarystyle{superraggedborder}{%
```

Base it on the `glostylesuperragged` style:

```
9141 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9142 \renewenvironment{theglossary}%
9143   {\tablehead{\hline}\tabletail{\hline}%
9144   \begin{supertabular}{l|l>{\raggedright}p{\glsgdescwidth}|}%
9145   {\end{supertabular}}%
9146 }
```

`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
9147 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
9148 \setglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
9149 \renewenvironment{theglossary}{%
```

```
9150 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
```

```
9151 \tabularnewline}%
```

```
9152 \tabletail{}}%
```

```
9153 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}%
```

```
9154 {\end{supertabular}}%
```

```
9155 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
9156 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
9157 \setglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
9158 \renewenvironment{theglossary}{%
```

```
9159 {\tablehead{\hline\bfseries \entryname &
```

```
9160 \bfseries \descriptionname\tabularnewline\hline}%
```

```
9161 \tabletail{\hline}
```

```
9162 \begin{supertabular}{ll|>{\raggedright}p{\glsgdescwidth}}}%
```

```
9163 {\end{supertabular}}%
```

```
9164 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
9165 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
9166 \renewenvironment{theglossary}{%
```

```
9167 {\tablehead{}\tabletail{}}%
```

```
9168 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
```

```
9169 >{\raggedright}p{\glspagelistwidth}}}%
```

```
9170 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9171 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9172 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9173 \renewcommand{\glossentry}[2]{%
```

```
9174 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9175 \glossentrydesc{##1} &
```

```

9176     ##2\tabularnewline
9177 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9178 \renewcommand{\subglossentry}[3]{%
9179     &
9180     \glssubentryitem{##2}%
9181     \glstarget{##2}{\strut}\glossentrydesc{##2} &
9182     ##3\tabularnewline
9183 }%

```

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>)

```

9184 \ifglsnogroupskip
9185 \renewcommand*{\glsgroupskip}{}%
9186 \else
9187 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9188 \fi
9189 }

```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```

9190 \newglossarystyle{superragged3colborder}{%

```

Base it on the glostypesuperragged3col style:

```

9191 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9192 \renewenvironment{theglossary}%
9193 {\tablehead{\hline}\tabletail{\hline}%
9194 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}||%
9195 >{\raggedright}p{\glspagerlistwidth}||}%
9196 {\end{supertabular}}%
9197 }

```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```

9198 \newglossarystyle{superragged3colheader}{%

```

Base it on the glostypesuperragged3col style:

```

9199 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9200 \renewenvironment{theglossary}%
9201 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9202 \bfseries\pagerlistname\tabularnewline}\tabletail{}%
9203 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9204 >{\raggedright}p{\glspagerlistwidth}}%
9205 {\end{supertabular}}%
9206 }

```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9207 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9208 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9209 \renewenvironment{theglossary}{%
9210   {\tablehead{\hline
9211     \bfseries\entryname&\bfseries\descriptionname&
9212     \bfseries\pagelistname\tabularnewline\hline}%
9213   \tabletail{\hline}%
9214   \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
9215     >{\raggedright}p{\glspagelistwidth}|}%
9216   {\end{supertabular}}}%
9217 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9218 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9219 \renewenvironment{theglossary}{%
9220   {\tablehead{}\tabletail{}%
9221   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9222     >{\raggedright}p{\glspagelistwidth}}}%
9223   {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
9224 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
9225 \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:

```
9226 \renewcommand{\glossentry}[2]{%
9227   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9228   \glossentrydesc{##1} &
9229   \glossentrysymbol{##1} & ##2\tabularnewline
9230 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9231 \renewcommand{\subglossentry}[3]{%
9232   &
9233   \glssubentryitem{##2}%
9234   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9235   \glossentrysymbol{##2} & ##3\tabularnewline
9236 }%
```

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>)

```
9237 \ifglsgroupskip
9238 \renewcommand*{\glsgroupskip}{}%
9239 \else
9240 \renewcommand*{\glsgroupskip}{& & & \tabularnewline}%
9241 \fi
9242 }
```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
9243 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
9244 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9245 \renewenvironment{theglossary}%
9246 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9247 \bfseries\symbolname &
9248 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9249 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9250 >{\raggedright}p{\glspagelistwidth}}}%
9251 {\end{supertabular}}}%
9252 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
9253 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9254 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9255 \renewenvironment{theglossary}%
9256 {\tablehead{\hline}\tabletail{\hline}%
9257 \begin{supertabular}%
9258 {l|>{\raggedright}p{\glsdescwidth}l|}%
9259 >{\raggedright}p{\glspagelistwidth}l|}%
9260 {\end{supertabular}}}%
9261 }
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
9262 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9263 \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:

```

9264 \renewenvironment{theglossary}%
9265   {\tablehead{\hline
9266     \bfseries\entryname &
9267     \bfseries\descriptionname &
9268     \bfseries\symbolname &
9269     \bfseries\pagelistname\tabularnewline\hline}%
9270   \tabletail{\hline}%
9271   \begin{supertabular}%
9272     {||>{\raggedright}p{\glsgdescwidth}||}%
9273     >{\raggedright}p{\glspagelistwidth}||}%
9274   {\end{supertabular}}%
9275 }

```

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.

```

9276 \ProvidesPackage{glossary-tree}[2018/05/10 v4.38 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9277 \providecommand{\indexspace}{%
9278   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9279 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glsglfont`.) This command was previously also used to format the group headings.

```

9280 \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.

```

9281 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9282 \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.

```

9283 \ifdef\@idxitem
9284 {\newcommand{\glstreeitem}{\@idxitem}}
9285 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```
9286 \ifdef\subitem
9287 {\let\glstreesubitem\subitem}
9288 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

`\glstreesubsubitem` Level 1 item used in index style.

```
9289 \ifdef\subsubitem
9290 {\let\glstreesubsubitem\subsubitem}
9291 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `alttree` style).

```
9292 \newcommand{\glstreepredesc}{\space}
```

`\glstreechildpredesc` Allow the user to adjust the space before the description for sub-entries (except for the `treenoname` and `alttree` style).

```
9293 \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.

```
9294 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9295 \renewenvironment{theglossary}%
9296 {\setlength{\parindent}{0pt}}%
9297 {\setlength{\parskip}{0pt plus 0.3pt}}%
9298 \let\item\glstreeitem
9299 \let\subitem\glstreesubitem
9300 \let\subsubitem\glstreesubsubitem
9301 }%
```

```
9302 {\par}}%
```

Do nothing at the start of the environment:

```
9303 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9304 \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.

```
9305 \renewcommand*{\glossentry}[2]{%
9306 \item\glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9307 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9308 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9309 }%
```


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.

```

9310 \renewcommand{\subglossentry}[3]{%
9311   \ifcase##1\relax
9312     % level 0
9313     \item
9314   \or
9315     % level 1
9316     \subitem
9317     \glssubentryitem{##2}%
9318   \else
9319     % all other levels
9320     \subsubitem
9321   \fi
9322   \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}%
9323   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9324   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9325 }%
```

Vertical gap between groups is the same as that used by indices:

```

9326 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

indexgroup The `indexgroup` style is like the `index` style but has headings.

```

9327 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```

9328 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9329 \renewcommand*{\glsgroupheading}[1]{%
9330   \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9331   \indexspace
9332 }%
9333 }
```

indexhypergroup The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

9334 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```

9335 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```

9336 \renewcommand*{\glossaryheader}{%
9337   \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.

```

9338 \renewcommand*{\glsgroupheading}[1]{%
9339   \item\glstreegroupheaderfmt

```

```

9340      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
9341      \indexspace}%
9342 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9343 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9344 \renewenvironment{theglossary}%
9345   {\setlength{\parindent}{0pt}%
9346    \setlength{\parskip}{0pt plus 0.3pt}}%
9347   {}%

```

Do nothing at the start of the theglossary environment:

```

9348 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9349 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

9350 \renewcommand{\glossentry}[2]{%
9351   \hangindent0pt\relax
9352   \parindent0pt\relax
9353   \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9354   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9355   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9356 }%

```

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.

```

9357 \renewcommand{\subglossentry}[3]{%
9358   \hangindent##1\glstreeindent\relax
9359   \parindent##1\glstreeindent\relax
9360   \ifnum##1=1\relax
9361     \glssubentryitem{##2}%
9362     \fi
9363     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9364     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9365     \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9366 }%

```

Vertical gap between groups is the same as that used by indices:

```

9367 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

treegroup Like the tree style but the glossary groups have headings.

```

9368 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

9369 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```
9370 \renewcommand{\glsgroupheading}[1]{\par
9371 \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par
9372 \indexspace}%
9373 }
```

treehypergroup The **treehypergroup** style is like the **treegroup** style, but has a set of links to the groups at the start of the glossary.

```
9374 \newglossarystyle{treehypergroup}{%
```

Base it on the **glostyletree** style:

```
9375 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the **theglossary** environment:

```
9376 \renewcommand*{\glossaryheader}{%
9377 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9378 \renewcommand*{\glsgroupheading}[1]{%
9379 \par\noindent
9380 \glstreegroupheaderfmt
9381 {\glsnahypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9382 \indexspace}%
9383 }
```

\glstreeindent Length governing left indent for each level of the tree style.

```
9384 \newlength\glstreeindent
9385 \setlength{\glstreeindent}{10pt}
```

treenoname The **treenoname** glossary style is like the **tree** style, but doesn't print the name or symbol for sub-levels.

```
9386 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9387 \renewenvironment{theglossary}%
9388 {\setlength{\parindent}{0pt}%
9389 \setlength{\parskip}{0pt plus 0.3pt}}%
9390 {}%
```

No header:

```
9391 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9392 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: the name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
9393 \renewcommand{\glossentry}[2]{%
9394 \hangindent0pt\relax
9395 \parindent0pt\relax
9396 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```

9397 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9398 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9399 }%

```

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.

```

9400 \renewcommand{\subglossentry}[3]{%
9401 \hangindent##1\glstreeindent\relax
9402 \parindent##1\glstreeindent\relax
9403 \ifnum##1=1\relax
9404 \glssubentryitem{##2}%
9405 \fi
9406 \glstarget{##2}{\strut}%
9407 \glossentrydesc{##2}\glspostdescription\space##3\par
9408 }%

```

Vertical gap between groups is the same as that used by indices:

```

9409 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9410 }

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```

9411 \newglossarystyle{treenonamegroup}{%
  Base it on the glostyletreenoname style:
9412 \setglossarystyle{treenoname}%
  Give each group a heading:
9413 \renewcommand{\glsgroupheading}[1]{\par
9414 \noindent\glstreegroupheaderfmt
9415 {\glsgrouptitle{##1}}\par\indexspace}%
9416 }

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9417 \newglossarystyle{treenonamehypergroup}{%
  Base it on the glostyletreenoname style:
9418 \setglossarystyle{treenoname}%

```

Put navigation links to the groups at the start of the `theglossary` environment:

```

9419 \renewcommand*{\glossaryheader}{%
9420 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9421 \renewcommand*{\glsgroupheading}[1]{%
9422 \par\noindent
9423 \glstreegroupheaderfmt
9424 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9425 \indexspace}%
9426 }

```

`esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```

9427 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9428   \dimen@=0pt\relax
9429   \gls@tmplen=0pt\relax
9430   \forallglossaries[#1]{\@gls@type}%
9431   {%
9432     \forallglsentries[\@gls@type]{\@glo@label}%
9433     {%
9434       \ifglsahasparent{\@glo@label}%
9435       }%
9436       {%
9437         \settowidth{\dimen@}%
9438           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
9439         \ifdim\dimen@>\gls@tmplen
9440           \gls@tmplen=\dimen@
9441           \letcs{\@glswidestname}{glo\glsdetoklabel{\@glo@label}@name}%
9442         \fi
9443       }%
9444     }%
9445   }%
9446 }
```

`\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.

```

9447 \newcommand*{\glssetwidest}[2][0]{%
9448   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9449     #2}%
9450 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```

9451 \newcommand*{\@glswidestname}{}
```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9452 \newcommand*{\glstreenamebox}[2]{%
9453   \makebox[#1][l]{#2}%
9454 }
```

`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`.

```

9455 \newglossarystyle{alttree}{%
```

Redefine theglossary environment.

```

9456 \renewenvironment{theglossary}%
9457   {\def\@gls@prevlevel{-1}%
9458    \mbox{}\par}%
9459   {\par}%

```

Set the header and group headers to nothing.

```

9460 \renewcommand*{\glossaryheader}{}%
9461 \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```
9462 \renewcommand{\glossentry}[2]{%
9463   \ifnum\@gls@prevlevel=0\relax
9464   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9465     \settowidth{\glstreeindent}{\glstreenamfmt{\@glswidestname\space}}%
9466   \fi
```

Set the hangindent and paragraph indent.

```
9467   \hangindent\glstreeindent
9468   \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9469   \makebox[0pt][r]{\glstreenambox{\glstreeindent}{%
9470     \glstryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9471   \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9472   \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9473   \def\@gls@prevlevel{0}%
9474 }%
```

Redefine the way sub-entries are displayed.

```
9475 \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.

```
9476   \ifnum##1=1\relax
9477     \glssubentryitem{##2}%
9478   \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9479   \ifnum\@gls@prevlevel=##1\relax
9480   \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9481     \@ifundefined{@glswidestname\romannumeral##1}{%
9482       \settowidth{\gls@tmplen}{\glstreenamfmt{\@glswidestname\space}}{%
9483       \settowidth{\gls@tmplen}{\glstreenamfmt{%
9484         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9485   \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9486      \setlength\glstreeindent\gls@tmplen
9487      \addtolength\glstreeindent\parindent
9488      \parindent\glstreeindent
9489      \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`.

```
9490      \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9491      \settowidth{\glstreeindent}{\glstreenamfmt{%
9492      \@glswidestname\space}}}{%
9493      \settowidth{\glstreeindent}{\glstreenamfmt{%
9494      \csname @glswidestname\romannumeral\@gls@prevlevel
9495      \endcsname\space}}}{%}
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9496      \addtolength\parindent{-\glstreeindent}%
9497      \setlength\glstreeindent\parindent
9498      \fi
9499      \fi
```

Set the hanging indentation.

```
9500      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9501      \makebox[0pt][r]{\glstreenambox{\gls@tmplen}{%
9502      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9503      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9504      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9505      \def\@gls@prevlevel{##1}%
9506      }%
```

Vertical gap between groups is the same as that used by indices:

```
9507      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9508 }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
9509 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
9510 \setglossarystyle{almtree}%
```

Give each group a heading.

```
9511 \renewcommand{\glsgroupheading}[1]{\par
9512 \def\@gls@prevlevel{-1}%
9513 \hangindent0pt\relax
```

```

9514     \parindent0pt\relax
9515     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
9516     \par\indexspace}%
9517 }

```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9518 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9519     \setglossarystyle{alttree}%
    Put the navigation links in the header
9520     \renewcommand*{\glossaryheader}{%
9521         \par
9522         \def\@gls@prevlevel{-1}%
9523         \hangindent0pt\relax
9524         \parindent0pt\relax
9525         \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9526     \renewcommand*{\glsgroupheading}[1]{%
9527         \par
9528         \def\@gls@prevlevel{-1}%
9529         \hangindent0pt\relax
9530         \parindent0pt\relax
9531         \glstreegroupheaderfmt
9532         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9533         \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.

```
9534 \NeedsTeXFormat{LaTeX2e}
9535 \ProvidesPackage{glossaries-compatible-207}[2018/05/10 v4.38 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9536 \ifglsxindy
9537   \renewcommand*\GlsAddXdyAttribute[1]{%
9538     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9539     \expandafter\toks@\expandafter{\@xdylocref}%
9540     \edef\@xdylocref{\the\toks@ ^^J}%
9541     (markup-locref
9542      :open \string"\string~n\string\setentrycounter
9543        {\noexpand\glscounter}%
9544        \expandafter\string\csize#1\endcsize
9545        \expandafter\@gobble\string\{\string" ^^J
9546        :close \string"\expandafter\@gobble\string\}\string" ^^J
9547        :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
9548 \fi
```

sAddXdyCounters

```
9549 \renewcommand*\GlsAddXdyCounters[1]{%
9550   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9551     in compatibility mode.}%
9552 }
```

Add predefined attributes

```
9553 \GlsAddXdyAttribute{glsnumberformat}
9554 \GlsAddXdyAttribute{textrm}
9555 \GlsAddXdyAttribute{textsf}
9556 \GlsAddXdyAttribute{texttt}
9557 \GlsAddXdyAttribute{textbf}
9558 \GlsAddXdyAttribute{textmd}
9559 \GlsAddXdyAttribute{textit}
9560 \GlsAddXdyAttribute{textup}
9561 \GlsAddXdyAttribute{textsl}
```

```

9562 \GlsAddXdyAttribute{textsc}
9563 \GlsAddXdyAttribute{emph}
9564 \GlsAddXdyAttribute{glshypernumber}
9565 \GlsAddXdyAttribute{hyperrm}
9566 \GlsAddXdyAttribute{hypersf}
9567 \GlsAddXdyAttribute{hypertt}
9568 \GlsAddXdyAttribute{hyperbf}
9569 \GlsAddXdyAttribute{hypermd}
9570 \GlsAddXdyAttribute{hyperit}
9571 \GlsAddXdyAttribute{hyperup}
9572 \GlsAddXdyAttribute{hypersl}
9573 \GlsAddXdyAttribute{hypersc}
9574 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9575 \ifglxindy
9576 \renewcommand*{\GlsAddXdyLocation}[2]{%
9577 \edef\@xdyuserlocationdefs{%
9578 \@xdyuserlocationdefs ^^J%
9579 (define-location-class \string"#1\string"^^J\space\space
9580 \space(#2))
9581 }%
9582 \edef\@xdyuserlocationnames{%
9583 \@xdyuserlocationnames^^J\space\space\space
9584 \string"#1\string"}%
9585 }
9586 \fi

```

\@do@wrglossary

```

9587 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9588 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9589 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9590 \def\@glo@range{}%
9591 \expandafter\if\@glo@prefix(\relax
9592 \def\@glo@range{:open-range}%
9593 \else
9594 \expandafter\if\@glo@prefix)\relax
9595 \def\@glo@range{:close-range}%
9596 \fi
9597 \fi

  Get the location and escape any special characters
9598 \protected@edef\@glslocref{\theglsentrycounter}%
9599 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9600 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9601 (indexentry :tkey (\csname glo@#1@index\endcsname)
9602   :locoref \string"\@glslocoref\string" %
9603   :attr \string"\@glo@suffix\string" \@glo@range
9604 )
9605 }%
9606 \else
    Convert the format information into the format required for makeindex
9607   \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat
    Write to the glossary file using makeindex syntax.
9608   \glossary[\csname glo@#1@type\endcsname]{%
9609     \string\glossaryentry{\csname glo@#1@index\endcsname
9610       \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9611 \fi
9612 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9613 \def\@set@glo@numformat#1#2#3{%
9614   \expandafter\@glo@check@mkidxrangechar#3\@nil
9615   \protected@edef#1{%
9616     \@glo@prefix setentrycounter[]{#2}%
9617     \expandafter\string\csname\@glo@suffix\endcsname
9618   }%
9619   \@gls@checkmkidxchars#1%
9620 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9621 \ifglxsindy
9622   \def\writeist{%
9623     \openout\glswrite=\istfilename
9624     \write\glswrite{;; xindy style file created by the glossaries
9625       package in compatible-2.07 mode}%
9626     \write\glswrite{;; for document '\jobname' on
9627       \the\year-\the\month-\the\day}%
9628     \write\glswrite{^^J; required styles^^J}
9629     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9630       \ifx\@xdystyle\@empty
9631       \else
9632         \protected@write\glswrite{{(require
9633           \string"\@xdystyle.xdy\string")}}%
9634       \fi
9635     }%
9636     \write\glswrite{^^J%
9637       ; list of allowed attributes (number formats)^^J}%
9638     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9639     \write\glswrite{^^J; user defined alphabets^^J}%
9640     \write\glswrite{\@xdyuseralphabets}%
9641     \write\glswrite{^^J; location class definitions^^J}%
9642     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9643     \string"roman-numbers-lowercase\string" :sep \string"}}%
9644 \@onelevel@sanitize\@gls@roman
9645 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9646     :sep \string"}}%
9647 \@onelevel@sanitize\@tmp
9648 \ifx\@tmp\@gls@roman
9649     \write\glswrite{(define-location-class
9650         \string"roman-page-numbers\string"^^J\space\space\space
9651         (\string"roman-numbers-lowercase\string")
9652         :min-range-length \@glsminrange)}}%
9653 \else
9654     \write\glswrite{(define-location-class
9655         \string"roman-page-numbers\string"^^J\space\space\space
9656         (:sep "\@gls@roman")
9657         :min-range-length \@glsminrange)}}%
9658 \fi
9659 \write\glswrite{(define-location-class
9660     \string"Roman-page-numbers\string"^^J\space\space\space
9661     (\string"roman-numbers-uppercase\string")
9662     :min-range-length \@glsminrange)}}%
9663 \write\glswrite{(define-location-class
9664     \string"arabic-page-numbers\string"^^J\space\space\space
9665     (\string"arabic-numbers\string")
9666     :min-range-length \@glsminrange)}}%
9667 \write\glswrite{(define-location-class
9668     \string"alpha-page-numbers\string"^^J\space\space\space
9669     (\string"alpha\string")
9670     :min-range-length \@glsminrange)}}%
9671 \write\glswrite{(define-location-class
9672     \string"Alpha-page-numbers\string"^^J\space\space\space
9673     (\string"ALPHA\string")
9674     :min-range-length \@glsminrange)}}%
9675 \write\glswrite{(define-location-class
9676     \string"Appendix-page-numbers\string"^^J\space\space\space
9677     (\string"ALPHA\string"
9678     :sep \string"\@glsAlphacompositor\string"
9679     \string"arabic-numbers\string")
9680     :min-range-length \@glsminrange)}}%
9681 \write\glswrite{(define-location-class
9682     \string"arabic-section-numbers\string"^^J\space\space\space
9683     (\string"arabic-numbers\string"
9684     :sep \string"\glscompositor\string"
9685     \string"arabic-numbers\string")
9686     :min-range-length \@glsminrange)}}%
9687 \write\glswrite{^^J; user defined location classes}%
9688 \write\glswrite{\@xdyuserlocationdefs}%
9689 \write\glswrite{^^J; define cross-reference class^^J}%
9690 \write\glswrite{(define-crossref-class \string"see\string"
9691     :unverified )}%

```

```

9692 \write\glswrite{(markup-crossref-list
9693 :class \string"see\string"^^J\space\space\space
9694 :open \string"\string\glseeformat\string"
9695 :close \string"{}\string")}%
9696 \write\glswrite{^^J; define the order of the location classes}%
9697 \write\glswrite{(define-location-class-order
9698 (\@xdylocationclassorder))}%
9699 \write\glswrite{^^J; define the glossary markup^^J}%
9700 \write\glswrite{(markup-index^^J\space\space\space
9701 :open \string"\string
9702 \glossarysection[\string\glossarytoctitle]{\string
9703 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9704 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9705 \space\space:close \string"\expandafter\@gobble
9706 \string%\string~n\string
9707 \end{theglossary}\string\glossarypostamble
9708 \string~n\string" ^^J\space\space\space
9709 :tree)}}%
9710 \write\glswrite{(markup-letter-group-list
9711 :sep \string"\string\glsgroupskip\string~n\string")}%
9712 \write\glswrite{(markup-indexentry
9713 :open \string"\string\relax \string\glresetentrylist
9714 \string~n\string")}%
9715 \write\glswrite{(markup-locclass-list :open
9716 \string"\glsoopenbrace\string\glossaryentrynumbers
9717 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9718 :sep \string", \string"
9719 :close \string"\glsclosebrace\glsclosebrace\string")}%
9720 \write\glswrite{(markup-locref-list
9721 :sep \string"\string\delimN\space\string")}%
9722 \write\glswrite{(markup-range
9723 :sep \string"\string\delimR\space\string")}%
9724 \@onelevel@sanitize\gls@suffixF
9725 \@onelevel@sanitize\gls@suffixFF
9726 \ifx\gls@suffixF\@empty
9727 \else
9728 \write\glswrite{(markup-range
9729 :close "\gls@suffixF" :length 1 :ignore-end)}%
9730 \fi
9731 \ifx\gls@suffixFF\@empty
9732 \else
9733 \write\glswrite{(markup-range
9734 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9735 \fi
9736 \write\glswrite{^^J; define format to use for locations^^J}%
9737 \write\glswrite{\@xdylocref}%
9738 \write\glswrite{^^J; define letter group list format^^J}%
9739 \write\glswrite{(markup-letter-group-list
9740 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9741 \write\glswrite{^^J; letter group headings^^J}%
9742 \write\glswrite{(markup-letter-group
9743   :open-head \string"\string\glsgroupheading
9744   \glsoopenbrace\string"^^J\space\space\space
9745   :close-head \string"\glsclosebrace\string")}%
9746 \write\glswrite{^^J; additional letter groups^^J}%
9747 \write\glswrite{\@xdylettergroups}%
9748 \write\glswrite{^^J; additional sort rules^^J}
9749 \write\glswrite{\@xdysortrules}%
9750 \noist}
9751 \else
9752 \edef\@gls@actualchar{\string?}
9753 \edef\@gls@encapchar{\string|}
9754 \edef\@gls@levelchar{\string!}
9755 \edef\@gls@quotechar{\string"}
9756 \def\writeist{\relax
9757   \openout\glswrite=\istfilename
9758   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9759     created by the glossaries package}
9760   \write\glswrite{\expandafter\@gobble\string\% for document
9761     '\jobname' on \the\year-\the\month-\the\day}
9762   \write\glswrite{actual '\@gls@actualchar'}
9763   \write\glswrite{encap '\@gls@encapchar'}
9764   \write\glswrite{level '\@gls@levelchar'}
9765   \write\glswrite{quote '\@gls@quotechar'}
9766   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9767   \write\glswrite{preamble \string"\string\glossarysection[\string
9768     \glossarytoctitle]{\string\glossarytitle}\string
9769     \glossarypreamble\string\n\string\begin{theglossary}\string
9770     \glossaryheader\string\n\string"}
9771   \write\glswrite{postamble \string"\string%\string\n\string
9772     \end{theglossary}\string\glossarypostamble\string\n
9773     \string"}
9774   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9775     \string"}
9776   \write\glswrite{item_0 \string"\string%\string\n\string"}
9777   \write\glswrite{item_1 \string"\string%\string\n\string"}
9778   \write\glswrite{item_2 \string"\string%\string\n\string"}
9779   \write\glswrite{item_01 \string"\string%\string\n\string"}
9780   \write\glswrite{item_x1
9781     \string"\string\relax \string\glsresetentrylist\string\n
9782     \string"}
9783   \write\glswrite{item_12 \string"\string%\string\n\string"}
9784   \write\glswrite{item_x2
9785     \string"\string\relax \string\glsresetentrylist\string\n
9786     \string"}
9787   \write\glswrite{delim_0 \string"\string\{\string
9788     \glossaryentrynumbers\string\{\string\relax \string"}
9789   \write\glswrite{delim_1 \string"\string\{\string

```

```

9790      \glossaryentrynumbers\string\{\string\relax \string}
9791      \write\glswrite{delim_2 \string"\string\{\string
9792      \glossaryentrynumbers\string\{\string\relax \string}
9793      \write\glswrite{delim_t \string"\string\}\string\}\string}
9794      \write\glswrite{delim_n \string"\string\delimN \string}
9795      \write\glswrite{delim_r \string"\string\delimR \string}
9796      \write\glswrite{headings_flag 1}
9797      \write\glswrite{heading_prefix
9798      \string"\string\glsgroupheading\string\{\string}
9799      \write\glswrite{heading_suffix
9800      \string"\string\}\string\relax
9801      \string\glsgroupresetentrylist \string}
9802      \write\glswrite{symhead_positive \string"glssymbols\string}
9803      \write\glswrite{numhead_positive \string"glslnumbers\string}
9804      \write\glswrite{page_compositor \string"glscpositor\string}
9805      \@gls@escbsdq\gls@suffixF
9806      \@gls@escbsdq\gls@suffixFF
9807      \ifx\gls@suffixF\@empty
9808      \else
9809      \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9810      \fi
9811      \ifx\gls@suffixFF\@empty
9812      \else
9813      \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9814      \fi
9815      \noist
9816    }
9817  \fi

```

\noist

```

9818 \renewcommand*{\noist}{\let\writeist\relax}

```

4.2 glossaries-compatible-307

```

9819 \NeedsTeXFormat{LaTeX2e}
9820 \ProvidesPackage{glossaries-compatible-307}[2018/05/10 v4.38 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9821 \newcommand{\compatglossarystyle}[2]{%
9822   \ifcsundef{@glscompstyle@#1}%
9823   {%
9824     \csdef{@glscompstyle@#1}{#2}%
9825   }%
9826   {%
9827     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9828   }%
9829 }

```

Backward compatible inline style.

```

9830 \compatglossarystyle{inline}{%
9831   \renewcommand{\glossaryentryfield}[5]{%
9832     \glsinlinedopostchild
9833     \gls@inlinesep
9834     \def\glo@desc{##3}%
9835     \def\@no@post@desc{\nopostdesc}%
9836     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
9837     \ifx\glo@desc\@no@post@desc
9838       \glsinlineemptydescformat{##4}{##5}%
9839     \else
9840       \ifstrepty{##3}%
9841         {\glsinlineemptydescformat{##4}{##5}}%
9842         {\glsinlinedescformat{##3}{##4}{##5}}%
9843     \fi
9844     \ifglshaschildren{##1}%
9845     {%
9846       \glsresetsubentrycounter
9847       \glsinlineparentchildseparator
9848       \def\gls@inlinesubsep{}%
9849       \def\gls@inlinepostchild{\glsinlinepostchild}%
9850     }%
9851   }%
9852   \def\gls@inlinesep{\glsinlineseparator}%
9853 }%
```

Sub-entries display description:

```

9854 \renewcommand{\glossarysubentryfield}[6]{%
9855   \gls@inlinesubsep%
9856   \glsinlinesubnameformat{##2}{##3}%
9857   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9858   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9859 }%
9860 }
```

Backward compatible list style.

```

9861 \compatglossarystyle{list}{%
9862   \renewcommand*{\glossaryentryfield}[5]{%
9863     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
9864       ##3\glspostdescription\space ##5}%
9865 }
```

Sub-entries continue on the same line:

```

9865 \renewcommand*{\glossarysubentryfield}[6]{%
9866   \glssubentryitem{##2}%
9867   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9868 }
```

Backward compatible listgroup style.

```

9869 \compatglossarystyle{listgroup}{%
9870   \csuse{@glscompstyle@list}%
9871 }%
```


Backward compatible listhypergroup style.

```
9872 \compatglossarystyle{listhypergroup}{%
9873   \csuse{@glscompstyle@list}%
9874 }%
```

Backward compatible altlist style.

```
9875 \compatglossarystyle{altlist}{%
9876   \renewcommand*{\glossaryentryfield}[5]{%
9877     \item[\glentryitem{##1}\glstarget{##1}{##2}]%
9878     \mbox{}\par\nobreak\@afterheading
9879     ##3\glspostdescription\space ##5}%
9880   \renewcommand{\glossarysubentryfield}[6]{%
9881     \par
9882     \glssubentryitem{##2}%
9883     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9884 }%
```

Backward compatible altlistgroup style.

```
9885 \compatglossarystyle{altlistgroup}{%
9886   \csuse{@glscompstyle@altlist}%
9887 }%
```

Backward compatible altlisthypergroup style.

```
9888 \compatglossarystyle{altlisthypergroup}{%
9889   \csuse{@glscompstyle@altlist}%
9890 }%
```

Backward compatible listdotted style.

```
9891 \compatglossarystyle{listdotted}{%
9892   \renewcommand*{\glossaryentryfield}[5]{%
9893     \item[\makebox[\glslistdottedwidth][l]{%
9894       \glentryitem{##1}\glstarget{##1}{##2}%
9895       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9896   \renewcommand*{\glossarysubentryfield}[6]{%
9897     \item[\makebox[\glslistdottedwidth][l]{%
9898       \glssubentryitem{##2}%
9899       \glstarget{##2}{##3}%
9900       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9901 }%
```

Backward compatible sublistdotted style.

```
9902 \compatglossarystyle{sublistdotted}{%
9903   \csuse{@glscompstyle@listdotted}%
9904   \renewcommand*{\glossaryentryfield}[5]{%
9905     \item[\glentryitem{##1}\glstarget{##1}{##2}]}%
9906 }%
```

Backward compatible long style.

```
9907 \compatglossarystyle{long}{%
9908   \renewcommand*{\glossaryentryfield}[5]{%
9909     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9910   \renewcommand*{\glossarysubentryfield}[6]{%
9911     \glssubentryitem{##2}\glstarget{##2}{##3} & ##4\glspostdescription\space ##5\\}%
9912 }%
```

```

9911      &
9912      \glssubentryitem{##2}%
9913      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9914 }%

```

Backward compatible longborder style.

```

9915 \compatglossarystyle{longborder}{%
9916   \csuse{@glscmpstyle@long}%
9917 }%

```

Backward compatible longheader style.

```

9918 \compatglossarystyle{longheader}{%
9919   \csuse{@glscmpstyle@long}%
9920 }%

```

Backward compatible longheaderborder style.

```

9921 \compatglossarystyle{longheaderborder}{%
9922   \csuse{@glscmpstyle@long}%
9923 }%

```

Backward compatible long3col style.

```

9924 \compatglossarystyle{long3col}{%
9925   \renewcommand*{\glossaryentryfield}[5]{%
9926     \glstarget{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
9927   \renewcommand*{\glossarysubentryfield}[6]{%
9928     &
9929     \glssubentryitem{##2}%
9930     \glstarget{##2}{\strut}##4 & ##6\\}%
9931 }%

```

Backward compatible long3colborder style.

```

9932 \compatglossarystyle{long3colborder}{%
9933   \csuse{@glscmpstyle@long3col}%
9934 }%

```

Backward compatible long3colheader style.

```

9935 \compatglossarystyle{long3colheader}{%
9936   \csuse{@glscmpstyle@long3col}%
9937 }%

```

Backward compatible long3colheaderborder style.

```

9938 \compatglossarystyle{long3colheaderborder}{%
9939   \csuse{@glscmpstyle@long3col}%
9940 }%

```

Backward compatible long4col style.

```

9941 \compatglossarystyle{long4col}{%
9942   \renewcommand*{\glossaryentryfield}[5]{%
9943     \glstarget{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9944   \renewcommand*{\glossarysubentryfield}[6]{%
9945     &
9946     \glssubentryitem{##2}%

```

9947 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
 9948 }%

Backward compatible long4colheader style.

9949 \compatglossarystyle{long4colheader}{%
 9950 \csuse{@glscompstyle@long4col}%
 9951 }%

Backward compatible long4colborder style.

9952 \compatglossarystyle{long4colborder}{%
 9953 \csuse{@glscompstyle@long4col}%
 9954 }%

Backward compatible long4colheaderborder style.

9955 \compatglossarystyle{long4colheaderborder}{%
 9956 \csuse{@glscompstyle@long4col}%
 9957 }%

Backward compatible altlong4col style.

9958 \compatglossarystyle{altlong4col}{%
 9959 \csuse{@glscompstyle@long4col}%
 9960 }%

Backward compatible altlong4colheader style.

9961 \compatglossarystyle{altlong4colheader}{%
 9962 \csuse{@glscompstyle@long4col}%
 9963 }%

Backward compatible altlong4colborder style.

9964 \compatglossarystyle{altlong4colborder}{%
 9965 \csuse{@glscompstyle@long4col}%
 9966 }%

Backward compatible altlong4colheaderborder style.

9967 \compatglossarystyle{altlong4colheaderborder}{%
 9968 \csuse{@glscompstyle@long4col}%
 9969 }%

Backward compatible long style.

9970 \compatglossarystyle{longragged}{%
 9971 \renewcommand*{\glossaryentryfield}[5]{%
 9972 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 9973 \tabularnewline}%
 9974 \renewcommand*{\glossarysubentryfield}[6]{%
 9975 &
 9976 \glssubentryitem{##2}%
 9977 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 9978 \tabularnewline}%
 9979 }%

Backward compatible longraggedborder style.

9980 \compatglossarystyle{longraggedborder}{%
 9981 \csuse{@glscompstyle@longragged}%
 9982 }%

Backward compatible longraggedheader style.

```
9983 \compatglossarystyle{longraggedheader}{%
9984 \csuse{@glscmpstyle@longragged}%
9985 }%
```

Backward compatible longraggedheaderborder style.

```
9986 \compatglossarystyle{longraggedheaderborder}{%
9987 \csuse{@glscmpstyle@longragged}%
9988 }%
```

Backward compatible longragged3col style.

```
9989 \compatglossarystyle{longragged3col}{%
9990 \renewcommand*{\glossaryentryfield}[5]{%
9991 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9992 \renewcommand*{\glossarysubentryfield}[6]{%
9993 &
9994 \glssubentryitem{##2}%
9995 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9996 }%
```

Backward compatible longragged3colborder style.

```
9997 \compatglossarystyle{longragged3colborder}{%
9998 \csuse{@glscmpstyle@longragged3col}%
9999 }%
```

Backward compatible longragged3colheader style.

```
10000 \compatglossarystyle{longragged3colheader}{%
10001 \csuse{@glscmpstyle@longragged3col}%
10002 }%
```

Backward compatible longragged3colheaderborder style.

```
10003 \compatglossarystyle{longragged3colheaderborder}{%
10004 \csuse{@glscmpstyle@longragged3col}%
10005 }%
```

Backward compatible altlongragged4col style.

```
10006 \compatglossarystyle{altlongragged4col}{%
10007 \renewcommand*{\glossaryentryfield}[5]{%
10008 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10009 \renewcommand*{\glossarysubentryfield}[6]{%
10010 &
10011 \glssubentryitem{##2}%
10012 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10013 }%
```

Backward compatible altlongragged4colheader style.

```
10014 \compatglossarystyle{altlongragged4colheader}{%
10015 \csuse{@glscmpstyle@altlong4col}%
10016 }%
```

Backward compatible altlongragged4colborder style.

```
10017 \compatglossarystyle{altlongragged4colborder}{%
```

```
10018 \csuse{@glscompstyle@altlong4col}%
10019 }%
```

Backward compatible altlongragged4colheaderborder style.

```
10020 \compatglossarystyle{altlongragged4colheaderborder}{%
10021 \csuse{@glscompstyle@altlong4col}%
10022 }%
```

Backward compatible index style.

```
10023 \compatglossarystyle{index}{%
10024 \renewcommand*{\glossaryentryfield}[5]{%
10025 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10026 \ifx\relax##4\relax
10027 \else
10028 \space{##4}%
10029 \fi
10030 \space ##3\glspostdescription \space ##5}%
10031 \renewcommand*{\glossarysubentryfield}[6]{%
10032 \ifcase##1\relax
10033 % level 0
10034 \item
10035 \or
10036 % level 1
10037 \subitem
10038 \glssubentryitem{##2}%
10039 \else
10040 % all other levels
10041 \subsubitem
10042 \fi
10043 \textbf{\glstarget{##2}{##3}}%
10044 \ifx\relax##5\relax
10045 \else
10046 \space{##5}%
10047 \fi
10048 \space##4\glspostdescription\space ##6}%
10049 }%
```

Backward compatible indexgroup style.

```
10050 \compatglossarystyle{indexgroup}{%
10051 \csuse{@glscompstyle@index}%
10052 }%
```

Backward compatible indexhypergroup style.

```
10053 \compatglossarystyle{indexhypergroup}{%
10054 \csuse{@glscompstyle@index}%
10055 }%
```

Backward compatible tree style.

```
10056 \compatglossarystyle{tree}{%
10057 \renewcommand{\glossaryentryfield}[5]{%
10058 \hangindent0pt\relax
```

```

10059 \parindent0pt\relax
10060 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10061 \ifx\relax##4\relax
10062 \else
10063 \space(##4)%
10064 \fi
10065 \space ##3\glspostdescription \space ##5\par}%
10066 \renewcommand{\glossarysubentryfield}[6]{%
10067 \hangindent##1\glstreeindent\relax
10068 \parindent##1\glstreeindent\relax
10069 \ifnum##1=1\relax
10070 \glssubentryitem{##2}%
10071 \fi
10072 \textbf{\glstarget{##2}{##3}}%
10073 \ifx\relax##5\relax
10074 \else
10075 \space(##5)%
10076 \fi
10077 \space##4\glspostdescription\space ##6\par}%
10078 }%

```

Backward compatible treegroup style.

```

10079 \compatglossarystyle{treegroup}{%
10080 \csuse{@glscmpstyle@tree}%
10081 }%

```

Backward compatible treehypergroup style.

```

10082 \compatglossarystyle{treehypergroup}{%
10083 \csuse{@glscmpstyle@tree}%
10084 }%

```

Backward compatible treenoname style.

```

10085 \compatglossarystyle{treenoname}{%
10086 \renewcommand{\glossaryentryfield}[5]{%
10087 \hangindent0pt\relax
10088 \parindent0pt\relax
10089 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10090 \ifx\relax##4\relax
10091 \else
10092 \space(##4)%
10093 \fi
10094 \space ##3\glspostdescription \space ##5\par}%
10095 \renewcommand{\glossarysubentryfield}[6]{%
10096 \hangindent##1\glstreeindent\relax
10097 \parindent##1\glstreeindent\relax
10098 \ifnum##1=1\relax
10099 \glssubentryitem{##2}%
10100 \fi
10101 \glstarget{##2}{\strut}%
10102 ##4\glspostdescription\space ##6\par}%
10103 }%

```

Backward compatible treenonamegroup style.

```
10104 \compatglossarystyle{treenonamegroup}{%
10105   \csuse{@glscompstyle@treenoname}%
10106 }%
```

Backward compatible treenonamehypergroup style.

```
10107 \compatglossarystyle{treenonamehypergroup}{%
10108   \csuse{@glscompstyle@treenoname}%
10109 }%
```

Backward compatible alttree style.

```
10110 \compatglossarystyle{alttree}{%
10111   \renewcommand{\glossaryentryfield}[5]{%
10112     \ifnum\@gls@prevlevel=0\relax
10113     \else
10114       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10115       \hangindent\glstreeindent
10116       \parindent\glstreeindent
10117     \fi
10118     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10119       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10120     \ifx\relax##4\relax
10121     \else
10122       (##4)\space
10123     \fi
10124     ##3\glspostdescription \space ##5\par
10125     \def\@gls@prevlevel{0}%
10126   }%
10127   \renewcommand{\glossarysubentryfield}[6]{%
10128     \ifnum##1=1\relax
10129     \glssubentryitem{##2}%
10130     \fi
10131     \ifnum\@gls@prevlevel=##1\relax
10132     \else
10133       \@ifundefined{@glswidestname\romannumeral##1}{%
10134         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10135         \settowidth{\gls@tmplen}{\textbf{%
10136           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10137         \ifnum\@gls@prevlevel<##1\relax
10138           \setlength\glstreeindent\gls@tmplen
10139           \addtolength\glstreeindent\parindent
10140           \parindent\glstreeindent
10141         \else
10142           \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10143             \settowidth{\glstreeindent}{\textbf{%
10144               \@glswidestname\space}}{%
10145             \settowidth{\glstreeindent}{\textbf{%
10146               \csname @glswidestname\romannumeral\@gls@prevlevel
10147               \endcsname\space}}}%
10148             \addtolength\parindent{-\glstreeindent}%

```

```

10149         \setlength\glstreeindent\parindent
10150     \fi
10151 \fi
10152 \hangindent\glstreeindent
10153 \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10154     \textbf{\glstarget{##2}{##3}}}%
10155 \ifx##5\relax\relax
10156 \else
10157     (##5)\space
10158 \fi
10159 ##4\glspostdescription\space ##6\par
10160 \def\@gls@prevlevel{##1}%
10161 }%
10162 }%

```

Backward compatible alttreegroup style.

```

10163 \compatglossarystyle{alttreegroup}{%
10164 \csuse{@glscompstyle@almtree}%
10165 }%

```

Backward compatible almtreehypergroup style.

```

10166 \compatglossarystyle{almtreehypergroup}{%
10167 \csuse{@glscompstyle@almtree}%
10168 }%

```

Backward compatible mcolindex style.

```

10169 \compatglossarystyle{mcolindex}{%
10170 \csuse{@glscompstyle@index}%
10171 }%

```

Backward compatible mcolindexgroup style.

```

10172 \compatglossarystyle{mcolindexgroup}{%
10173 \csuse{@glscompstyle@index}%
10174 }%

```

Backward compatible mcolindexhypergroup style.

```

10175 \compatglossarystyle{mcolindexhypergroup}{%
10176 \csuse{@glscompstyle@index}%
10177 }%

```

Backward compatible mcoltree style.

```

10178 \compatglossarystyle{mcoltree}{%
10179 \csuse{@glscompstyle@tree}%
10180 }%

```

Backward compatible mcoltreegroup style.

```

10181 \compatglossarystyle{mcolindextreegroup}{%
10182 \csuse{@glscompstyle@tree}%
10183 }%

```

Backward compatible mcoltreehypergroup style.

```

10184 \compatglossarystyle{mcolindextreehypergroup}{%

```



```

10185 \csuse{@glscompstyle@tree}%
10186 }%

    Backward compatible mcoltreenoname style.
10187 \compatglossarystyle{mcoltreenoname}{%
10188 \csuse{@glscompstyle@tree}%
10189 }%

    Backward compatible mcoltreenonamegroup style.
10190 \compatglossarystyle{mcoltreenonamegroup}{%
10191 \csuse{@glscompstyle@tree}%
10192 }%

    Backward compatible mcoltreenonamehypergroup style.
10193 \compatglossarystyle{mcoltreenonamehypergroup}{%
10194 \csuse{@glscompstyle@tree}%
10195 }%

    Backward compatible mcolalmtree style.
10196 \compatglossarystyle{mcolalmtree}{%
10197 \csuse{@glscompstyle@almtree}%
10198 }%

    Backward compatible mcolalmtreegroup style.
10199 \compatglossarystyle{mcolalmtreegroup}{%
10200 \csuse{@glscompstyle@almtree}%
10201 }%

    Backward compatible mcolalmtreehypergroup style.
10202 \compatglossarystyle{mcolalmtreehypergroup}{%
10203 \csuse{@glscompstyle@almtree}%
10204 }%

    Backward compatible superragged style.
10205 \compatglossarystyle{superragged}{%
10206 \renewcommand*{\glossaryentryfield}[5]{%
10207 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10208 \tabularnewline}%
10209 \renewcommand*{\glossarysubentryfield}[6]{%
10210 &
10211 \glssubentryitem{##2}%
10212 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10213 \tabularnewline}%
10214 }%

    Backward compatible superraggedborder style.
10215 \compatglossarystyle{superraggedborder}{%
10216 \csuse{@glscompstyle@superragged}%
10217 }%

    Backward compatible superraggedheader style.
10218 \compatglossarystyle{superraggedheader}{%
10219 \csuse{@glscompstyle@superragged}%
10220 }%

```

Backward compatible superraggedheaderborder style.

```
10221 \compatglossarystyle{superraggedheaderborder}{%
10222   \csuse{@glscompstyle@superragged}%
10223 }%
```

Backward compatible superragged3col style.

```
10224 \compatglossarystyle{superragged3col}{%
10225   \renewcommand*{\glossaryentryfield}[5]{%
10226     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10227   \renewcommand*{\glossarysubentryfield}[6]{%
10228     &
10229     \glssubentryitem{##2}%
10230     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10231 }%
```

Backward compatible superragged3colborder style.

```
10232 \compatglossarystyle{superragged3colborder}{%
10233   \csuse{@glscompstyle@superragged3col}%
10234 }%
```

Backward compatible superragged3colheader style.

```
10235 \compatglossarystyle{superragged3colheader}{%
10236   \csuse{@glscompstyle@superragged3col}%
10237 }%
```

Backward compatible superragged3colheaderborder style.

```
10238 \compatglossarystyle{superragged3colheaderborder}{%
10239   \csuse{@glscompstyle@superragged3col}%
10240 }%
```

Backward compatible altsuperragged4col style.

```
10241 \compatglossarystyle{altsuperragged4col}{%
10242   \renewcommand*{\glossaryentryfield}[5]{%
10243     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10244   \renewcommand*{\glossarysubentryfield}[6]{%
10245     &
10246     \glssubentryitem{##2}%
10247     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10248 }%
```

Backward compatible altsuperragged4colheader style.

```
10249 \compatglossarystyle{altsuperragged4colheader}{%
10250   \csuse{@glscompstyle@altsuperragged4col}%
10251 }%
```

Backward compatible altsuperragged4colborder style.

```
10252 \compatglossarystyle{altsuperragged4colborder}{%
10253   \csuse{@glscompstyle@altsuperragged4col}%
10254 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10255 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10256 \csuse{@glscompstyle@altsuperragged4col}%
10257 }%
```

Backward compatible super style.

```
10258 \compatglossarystyle{super}{%
10259   \renewcommand*{\glossaryentryfield}[5]{%
10260     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10261   \renewcommand*{\glossarysubentryfield}[6]{%
10262     &
10263     \glssubentryitem{##2}%
10264     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10265 }%
```

Backward compatible superborder style.

```
10266 \compatglossarystyle{superborder}{%
10267   \csuse{@glscompstyle@super}%
10268 }%
```

Backward compatible superheader style.

```
10269 \compatglossarystyle{superheader}{%
10270   \csuse{@glscompstyle@super}%
10271 }%
```

Backward compatible superheaderborder style.

```
10272 \compatglossarystyle{superheaderborder}{%
10273   \csuse{@glscompstyle@super}%
10274 }%
```

Backward compatible super3col style.

```
10275 \compatglossarystyle{super3col}{%
10276   \renewcommand*{\glossaryentryfield}[5]{%
10277     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10278   \renewcommand*{\glossarysubentryfield}[6]{%
10279     &
10280     \glssubentryitem{##2}%
10281     \glstarget{##2}{\strut}##4 & ##6\\}%
10282 }%
```

Backward compatible super3colborder style.

```
10283 \compatglossarystyle{super3colborder}{%
10284   \csuse{@glscompstyle@super3col}%
10285 }%
```

Backward compatible super3colheader style.

```
10286 \compatglossarystyle{super3colheader}{%
10287   \csuse{@glscompstyle@super3col}%
10288 }%
```

Backward compatible super3colheaderborder style.

```
10289 \compatglossarystyle{super3colheaderborder}{%
10290   \csuse{@glscompstyle@super3col}%
10291 }%
```

Backward compatible super4col style.

```
10292 \compatglossarystyle{super4col}{%
10293   \renewcommand*{\glossaryentryfield}[5]{%
10294     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10295   \renewcommand*{\glossarysubentryfield}[6]{%
10296     &
10297     \glssubentryitem{##2}%
10298     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10299 }%
```

Backward compatible super4colheader style.

```
10300 \compatglossarystyle{super4colheader}{%
10301   \csuse{@glscompstyle@super4col}%
10302 }%
```

Backward compatible super4colborder style.

```
10303 \compatglossarystyle{super4colborder}{%
10304   \csuse{@glscompstyle@super4col}%
10305 }%
```

Backward compatible super4colheaderborder style.

```
10306 \compatglossarystyle{super4colheaderborder}{%
10307   \csuse{@glscompstyle@super4col}%
10308 }%
```

Backward compatible altsuper4col style.

```
10309 \compatglossarystyle{altsuper4col}{%
10310   \csuse{@glscompstyle@super4col}%
10311 }%
```

Backward compatible altsuper4colheader style.

```
10312 \compatglossarystyle{altsuper4colheader}{%
10313   \csuse{@glscompstyle@super4col}%
10314 }%
```

Backward compatible altsuper4colborder style.

```
10315 \compatglossarystyle{altsuper4colborder}{%
10316   \csuse{@glscompstyle@super4col}%
10317 }%
```

Backward compatible altsuper4colheaderborder style.

```
10318 \compatglossarystyle{altsuper4colheaderborder}{%
10319   \csuse{@glscompstyle@super4col}%
10320 }%
```

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.

```
10321 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10322 \ProvidesPackage{glossaries-accsupp}[2018/05/10 v4.38 (NLCT)]
```

```
10323 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10324 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10325 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10326 \@ifpackageloaded{glossaries-extra}
```

```
10327 {%
```

If the accsupp option was used, \@glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10328 \ifx\@glsxtr@doaccsupp\empty
```

```
10329 \GlossariesWarning{The 'glossaries-accsupp'
```

```
10330 package has been loaded\MessageBreak
```

```
10331 after the 'glossaries-extra' package. This\MessageBreak
```

```
10332 can cause a failure to integrate both packages. \MessageBreak
```

```
10333 Either use the 'accsupp' option when you load\MessageBreak
```

```
10334 'glossaries-extra' or load 'glossaries-accsupp'\MessageBreak
```

```
10335 before loading 'glossaries-extra'}%
```

```
10336 \fi
```

```
10337 }
```

```
10338 {}
```

tibleglossentry Override style compatibility macros:

```
10339 \def\compatibleglossentry#1#2{%
```

```
10340 \toks0{#2}%
```

```
10341 \protected@edef\do@glossentry{%
```

```
10342 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10343 {\noexpand\glsnamefont
```

```
10344 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```

10345     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10346     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10347     {\the\toks@}%
10348   }%
10349   \@do@glossentry
10350 }

```

lesubglossentry

```

10351 \def\compatiblesubglossentry#1#2#3{%
10352   \toks@{#3}%
10353   \protected@edef\@do@subglossentry{%
10354     \noexpand\accsuppglossarysubentryfield{\number#1}%
10355     {#2}%
10356     {\noexpand\glsnamefont
10357      {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}%
10358     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10359     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10360     {\the\toks@}%
10361   }%
10362   \@do@subglossentry
10363 }

```

Required packages:

```

10364 \RequirePackage{glossaries}
10365 \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:

```

10366 \define@key{glossentry}{access}{%
10367   \def\@glo@access{#1}%
10368 }

```

textaccess The replacement text corresponding to the text key:

```

10369 \define@key{glossentry}{textaccess}{%
10370   \def\@glo@textaccess{#1}%
10371 }

```

firstaccess The replacement text corresponding to the first key:

```

10372 \define@key{glossentry}{firstaccess}{%
10373   \def\@glo@firstaccess{#1}%
10374 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10375 \define@key{glossentry}{pluralaccess}{%
10376   \def\@glo@pluralaccess{#1}%
10377 }
```

rstpluralaccess The replacement text corresponding to the firstplural key:

```

10378 \define@key{glossentry}{firstpluralaccess}{%
10379   \def\@glo@firstpluralaccess{#1}%
10380 }
```

symbolaccess The replacement text corresponding to the symbol key:

```

10381 \define@key{glossentry}{symbolaccess}{%
10382   \def\@glo@symbolaccess{#1}%
10383 }
```

bolpluralaccess The replacement text corresponding to the symbolplural key:

```

10384 \define@key{glossentry}{symbolpluralaccess}{%
10385   \def\@glo@symbolpluralaccess{#1}%
10386 }
```

scriptionaccess The replacement text corresponding to the description key:

```

10387 \define@key{glossentry}{descriptionaccess}{%
10388   \def\@glo@descaccess{#1}%
10389 }
```

ionpluralaccess The replacement text corresponding to the descriptionplural key:

```

10390 \define@key{glossentry}{descriptionpluralaccess}{%
10391   \def\@glo@descpluralaccess{#1}%
10392 }
```

shortaccess The replacement text corresponding to the short key:

```

10393 \define@key{glossentry}{shortaccess}{%
10394   \def\@glo@shortaccess{#1}%
10395 }
```

ortpluralaccess The replacement text corresponding to the shortplural key:

```

10396 \define@key{glossentry}{shortpluralaccess}{%
10397   \def\@glo@shortpluralaccess{#1}%
10398 }
```

longaccess The replacement text corresponding to the long key:

```

10399 \define@key{glossentry}{longaccess}{%
10400   \def\@glo@longaccess{#1}%
10401 }
```

ongpluralaccess The replacement text corresponding to the longplural key:

```

10402 \define@key{glossentry}{longpluralaccess}{%
10403   \def\@glo@longpluralaccess{#1}%
10404 }
```

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:

```
10405 \appto\@gls@keymap{,%
10406   {access}{access},%
10407   {textaccess}{textaccess},%
10408   {firstaccess}{firstaccess},%
10409   {pluralaccess}{pluralaccess},%
10410   {firstpluralaccess}{firstpluralaccess},%
10411   {symbolaccess}{symbolaccess},%
10412   {symbolpluralaccess}{symbolpluralaccess},%
10413   {descaccess}{descaccess},%
10414   {descpluralaccess}{descpluralaccess},%
10415   {shortaccess}{shortaccess},%
10416   {shortpluralaccess}{shortpluralaccess},%
10417   {longaccess}{longaccess},%
10418   {longpluralaccess}{longpluralaccess}}%
10419 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
10420 \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):

```
10421 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10422 \renewcommand*{\@newglossaryentryprehook}{%
10423   \@gls@oldnewglossaryentryprehook
10424   \def\@glo@access{\@glo@symbol}}%
```

Initialise the other keys:

```
10425 \def\@glo@textaccess{\@glo@access}%
10426 \def\@glo@firstaccess{\@glo@access}%
10427 \def\@glo@pluralaccess{\@glo@textaccess}%
10428 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10429 \def\@glo@symbolaccess{\relax}%
10430 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10431 \def\@glo@descaccess{\relax}%
10432 \def\@glo@descpluralaccess{\@glo@descaccess}%
10433 \def\@glo@shortaccess{\relax}%
10434 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10435 \def\@glo@longaccess{\relax}%
10436 \def\@glo@longpluralaccess{\@glo@longaccess}%
10437 }
```

Add to the end hook:

```
10438 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10439 \renewcommand*{\@newglossaryentryposthook}{%
10440   \@gls@oldnewglossaryentryposthook}
```


Store the access information:

```
10441 \expandafter
10442 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10443 \@glo@access}%
10444 \expandafter
10445 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10446 \@glo@textaccess}%
10447 \expandafter
10448 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10449 \@glo@firstaccess}%
10450 \expandafter
10451 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10452 \@glo@pluralaccess}%
10453 \expandafter
10454 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10455 \@glo@firstpluralaccess}%
10456 \expandafter
10457 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10458 \@glo@symbolaccess}%
10459 \expandafter
10460 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10461 \@glo@symbolpluralaccess}%
10462 \expandafter
10463 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10464 \@glo@descaccess}%
10465 \expandafter
10466 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10467 \@glo@descpluralaccess}%
10468 \expandafter
10469 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10470 \@glo@shortaccess}%
10471 \expandafter
10472 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10473 \@glo@shortpluralaccess}%
10474 \expandafter
10475 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10476 \@glo@longaccess}%
10477 \expandafter
10478 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10479 \@glo@longpluralaccess}%
10480 }
```

5.2 Accessing Replacement Text

`\glentryaccess` Get the value of the access key for the entry with the given label:

```
10481 \newcommand*{\glentryaccess}[1]{%
10482 \@gls@entry@field{#1}{access}%
10483 }
```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
10484 \newcommand*{\glentrytextaccess}[1]{%
10485   \@gls@entry@field{#1}{textaccess}%
10486 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10487 \newcommand*{\glentryfirstaccess}[1]{%
10488   \@gls@entry@field{#1}{firstaccess}%
10489 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10490 \newcommand*{\glentrypluralaccess}[1]{%
10491   \@gls@entry@field{#1}{pluralaccess}%
10492 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10493 \newcommand*{\glentryfirstpluralaccess}[1]{%
10494   \csname glo@#1@firstpluralaccess\endcsname
10495 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10496 \newcommand*{\glentrysymbolaccess}[1]{%
10497   \@gls@entry@field{#1}{symbolaccess}%
10498 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10499 \newcommand*{\glentrysymbolpluralaccess}[1]{%
10500   \@gls@entry@field{#1}{symbolpluralaccess}%
10501 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10502 \newcommand*{\glentrydescaccess}[1]{%
10503   \@gls@entry@field{#1}{descaccess}%
10504 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10505 \newcommand*{\glentrydescpluralaccess}[1]{%
10506   \@gls@entry@field{#1}{descaccess}%
10507 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10508 \newcommand*{\glentryshortaccess}[1]{%
10509   \@gls@entry@field{#1}{shortaccess}%
10510 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10511 \newcommand*{\glentryshortpluralaccess}[1]{%
10512   \@gls@entry@field{#1}{shortpluralaccess}%
10513 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10514 \newcommand*{\glsentrylongaccess}[1]{%
10515   \@gls@entry@field{#1}{longaccess}%
10516 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10517 \newcommand*{\glsentrylongpluralaccess}[1]{%
10518   \@gls@entry@field{#1}{longpluralaccess}%
10519 }
```

`\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.)

```
10520 \newcommand*{\glsaccsupp}[2]{%
10521   \BeginAccSupp{ActualText={#1}}#2\EndAccSupp{}%
10522 }
```

`\xglsaccsupp` Fully expands replacement text before calling `\glsaccsupp`

```
10523 \newcommand*{\xglsaccsupp}[2]{%
10524   \protected@edef\@gls@replacementtext{#1}%
10525   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10526 }
```

`@access@display`

```
10527 \newcommand*{\@gls@access@display}[2]{%
10528   \protected@edef\@glo@access{#2}%
10529   \ifx\@glo@access\@gls@noaccess
10530     #1%
10531   \else
10532     \xglsaccsupp{\@glo@access}{#1}%
10533   \fi
10534 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10535 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10536   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10537 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10538 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10539   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10540 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10541 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
10542   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10543 }
```

staccessdisplay As above but for the firstaccess replacement text.

```
10544 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
10545   \@gls@access@display{#1}{\glsentryfirstaccess{#2}}%
10546 }
```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```
10547 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
10548   \@gls@access@display{#1}{\glsentryfirstpluralaccess{#2}}%
10549 }
```

olaccessdisplay As above but for the symbolaccess replacement text.

```
10550 \DeclareRobustCommand*\glsymbolaccessdisplay}[2]{%
10551   \@gls@access@display{#1}{\glsentrysymbolaccess{#2}}%
10552 }
```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```
10553 \DeclareRobustCommand*\glsymbolpluralaccessdisplay}[2]{%
10554   \@gls@access@display{#1}{\glsentrysymbolpluralaccess{#2}}%
10555 }
```

onaccessdisplay As above but for the descriptionaccess replacement text.

```
10556 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10557   \@gls@access@display{#1}{\glsentrydescaccess{#2}}%
10558 }
```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```
10559 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10560   \@gls@access@display{#1}{\glsentrydescpluralaccess{#2}}%
10561 }
```

rtaccessdisplay As above but for the shortaccess replacement text.

```
10562 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10563   \@gls@access@display{#1}{\glsentryshortaccess{#2}}%
10564 }
```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```
10565 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10566   \@gls@access@display{#1}{\glsentryshortpluralaccess{#2}}%
10567 }
```

ngaccessdisplay As above but for the longaccess replacement text.

```
10568 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10569   \@gls@access@display{#1}{\glsentrylongaccess{#2}}%
10570 }
```

alaccessdisplay As above but for the longpluralaccess replacement text.

```
10571 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10572   \@gls@access@display{#1}{\glsentrylongpluralaccess{#2}}%
10573 }
```

lsaccessdisplay Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```
10574 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10575   \ifundefined{gls#1accessdisplay}%
10576   {%
10577     \PackageError{glossaries-accsupp}{No accessibility support
10578       for key ‘#1’}{%
10579     }%
10580   {%
10581     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10582   }%
10583 }
```

efault@entryfmt Redefine the default entry format to use accessibility information

```
10584 \renewcommand*\@@gls@default@entryfmt}[2]{%
10585   \ifdefempty\glscustomtext
10586   {%
10587     \glsifplural
10588     {%
```

Plural form

```
10589     \glscapscase
10590     {%
```

Don't adjust case

```
10591     \ifglsused\glslabel
10592     {%
```

Subsequent use

```
10593     #2{\glspluralaccessdisplay
10594         {\glsentryplural{\glslabel}}{\glslabel}}%
10595     {\glsdescriptionpluralaccessdisplay
10596         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10597     {\glsymbolpluralaccessdisplay
10598         {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10599     {\glsinsert}%
10600   }%
10601   {%
```

First use

```
10602     #1{\glsfirstpluralaccessdisplay
10603         {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10604     {\glsdescriptionpluralaccessdisplay
10605         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10606     {\glsymbolpluralaccessdisplay
```

```

10607          {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10608      {\glsinsert}%
10609  }%
10610  }%
10611  {%

```

Make first letter upper case

```

10612      \ifglsused\glslabel
10613      {%

```

Subsequent use.

```

10614      #2{\glspluralaccessdisplay
10615          {\Glsentryplural{\glslabel}}{\glslabel}}%
10616          {\glsdescriptionpluralaccessdisplay
10617              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10618          {\glssymbolpluralaccessdisplay
10619              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10620          {\glsinsert}%
10621      }%
10622      {%

```

First use

```

10623      #1{\glsfirstpluralaccessdisplay
10624          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10625          {\glsdescriptionpluralaccessdisplay
10626              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10627          {\glssymbolpluralaccessdisplay
10628              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10629          {\glsinsert}%
10630      }%
10631      }%
10632      {%

```

Make all upper case

```

10633      \ifglsused\glslabel
10634      {%

```

Subsequent use

```

10635      \MakeUppercase{%
10636          #2{\glspluralaccessdisplay
10637              {\glsentryplural{\glslabel}}{\glslabel}}%
10638              {\glsdescriptionpluralaccessdisplay
10639                  {\glsentrydescplural{\glslabel}}{\glslabel}}%
10640                  {\glssymbolpluralaccessdisplay
10641                      {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10642                      {\glsinsert}}%
10643      }%
10644      {%

```

First use

```

10645      \MakeUppercase{%
10646          #1{\glsfirstpluralaccessdisplay

```

```

10647         {\glentryfirstplural{\glslabel}}{\glslabel}}%
10648     {\glsdescriptionpluralaccessdisplay
10649         {\glentrydescplural{\glslabel}}{\glslabel}}%
10650     {\glssymbolpluralaccessdisplay
10651         {\glentrysymbolplural{\glslabel}}{\glslabel}}%
10652     {\glsinsert}}%
10653 }%
10654 }%
10655 }%
10656 {%

```

Singular form

```

10657     \glscapscase
10658     {%

```

Don't adjust case

```

10659     \ifglsused\glslabel
10660     {%

```

Subsequent use

```

10661     #2{\glstextaccessdisplay
10662         {\glentrytext{\glslabel}}{\glslabel}}%
10663     {\glsdescriptionaccessdisplay
10664         {\glentrydesc{\glslabel}}{\glslabel}}%
10665     {\glssymbolaccessdisplay
10666         {\glentrysymbol{\glslabel}}{\glslabel}}%
10667     {\glsinsert}}%
10668 }%
10669 {%

```

First use

```

10670     #1{\glsfirstaccessdisplay
10671         {\glentryfirst{\glslabel}}{\glslabel}}%
10672     {\glsdescriptionaccessdisplay
10673         {\glentrydesc{\glslabel}}{\glslabel}}%
10674     {\glssymbolaccessdisplay
10675         {\glentrysymbol{\glslabel}}{\glslabel}}%
10676     {\glsinsert}}%
10677 }%
10678 }%
10679 {%

```

Make first letter upper case

```

10680     \ifglsused\glslabel
10681     {%

```

Subsequent use

```

10682     #2{\glstextaccessdisplay
10683         {\glentrytext{\glslabel}}{\glslabel}}%
10684     {\glsdescriptionaccessdisplay
10685         {\glentrydesc{\glslabel}}{\glslabel}}%
10686     {\glssymbolaccessdisplay

```

```

10687         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10688     {\glsinsert}%
10689 }%
10690 {%

```

First use

```

10691     #1{\glsfirstaccessdisplay
10692         {\glsentryfirst{\glslabel}}{\glslabel}}%
10693     {\glsdescriptionaccessdisplay
10694         {\glsentrydesc{\glslabel}}{\glslabel}}%
10695     {\glsymbolaccessdisplay
10696         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10697     {\glsinsert}%
10698 }%
10699 }%
10700 {%

```

Make all upper case

```

10701     \ifglsused\glslabel
10702     {%

```

Subsequent use

```

10703     \MakeUppercase{%
10704     #2{\glsfirstaccessdisplay
10705         {\glsentrytext{\glslabel}}{\glslabel}}%
10706     {\glsdescriptionaccessdisplay
10707         {\glsentrydesc{\glslabel}}{\glslabel}}%
10708     {\glsymbolaccessdisplay
10709         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10710     {\glsinsert}}%
10711 }%
10712 {%

```

First use

```

10713     \MakeUppercase{%
10714     #1{\glsfirstaccessdisplay
10715         {\glsentryfirst{\glslabel}}{\glslabel}}%
10716     {\glsdescriptionaccessdisplay
10717         {\glsentrydesc{\glslabel}}{\glslabel}}%
10718     {\glsymbolaccessdisplay
10719         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10720     {\glsinsert}}%
10721 }%
10722 }%
10723 }%
10724 }%
10725 {%

```

Custom text provided in \glsdisp

```

10726     \ifglsused{\glslabel}%
10727     {%

```


Subsequent use

```

10728      #2{\glscustomtext}%
10729      {\glsdescriptionaccessdisplay
10730       {\glentrydesc{\glslabel}}{\glslabel}}%
10731      {\glssymbolaccessdisplay
10732       {\glentrysymbol{\glslabel}}{\glslabel}}%
10733      {\glsinsert}%
10734    }%
10735    {%

```

First use

```

10736      #1{\glscustomtext}%
10737      {\glsdescriptionaccessdisplay
10738       {\glentrydesc{\glslabel}}{\glslabel}}%
10739      {\glssymbolaccessdisplay
10740       {\glentrysymbol{\glslabel}}{\glslabel}}%
10741      {\glsinsert}%
10742    }%
10743  }%
10744 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10745 \renewcommand*{\glsgenentryfmt}{%
10746   \ifdefempty\glscustomtext
10747   {%
10748     \glsifplural
10749     {%

```

Plural form

```

10750     \glscapscase
10751     {%

```

Don't adjust case

```

10752     \ifglused\glslabel
10753     {%

```

Subsequent use

```

10754       \glspluralaccessdisplay
10755       {\glentryplural{\glslabel}}{\glslabel}%
10756       \glsinsert
10757     }%
10758     {%

```

First use

```

10759       \glsfirstpluralaccessdisplay
10760       {\glentryfirstplural{\glslabel}}{\glslabel}%
10761       \glsinsert
10762     }%
10763   }%
10764   {%

```

Make first letter upper case

10765 \ifglused\glslabel
10766 {%

Subsequent use.

10767 \glspluralaccessdisplay
10768 {\Glentryplural{\glslabel}}{\glslabel}%
10769 \glsinsert
10770 }%
10771 {%

First use

10772 \glsfirstpluralaccessdisplay
10773 {\Glentryfirstplural{\glslabel}}{\glslabel}%
10774 \glsinsert
10775 }%
10776 }%
10777 {%

Make all upper case

10778 \ifglused\glslabel
10779 {%

Subsequent use

10780 \glspluralaccessdisplay
10781 {\mfirstucMakeUppercase{\glentryplural{\glslabel}}}%
10782 {\glslabel}%
10783 \mfirstucMakeUppercase{\glsinsert}%
10784 }%
10785 {%

First use

10786 \glsfirstpluralacessdisplay
10787 {\mfirstucMakeUppercase{\glentryfirstplural{\glslabel}}}%
10788 {\glslabel}%
10789 \mfirstucMakeUppercase{\glsinsert}%
10790 }%
10791 }%
10792 }%
10793 {%

Singular form

10794 \glscapscase
10795 {%

Don't adjust case

10796 \ifglused\glslabel
10797 {%

Subsequent use

10798 \glstextaccessdisplay{\glentrytext{\glslabel}}{\glslabel}%
10799 \glsinsert

```

10800      }%
10801      {%

```

First use

```

10802      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10803      \glsinsert
10804      }%
10805      }%
10806      {%

```

Make first letter upper case

```

10807      \ifglsused\glslabel
10808      {%

```

Subsequent use

```

10809      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10810      \glsinsert
10811      }%
10812      {%

```

First use

```

10813      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10814      \glsinsert
10815      }%
10816      }%
10817      {%

```

Make all upper case

```

10818      \ifglsused\glslabel
10819      {%

```

Subsequent use

```

10820      \glstextaccessdisplay
10821      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10822      \mfirstucMakeUppercase{\glsinsert}%
10823      }%
10824      {%

```

First use

```

10825      \glsfirstaccessdisplay
10826      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10827      \mfirstucMakeUppercase{\glsinsert}%
10828      }%
10829      }%
10830      }%
10831      }%
10832      {%

```

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.

```

10833      \glscustomtext\glsinsert
10834      }%
10835      }

```

`\glsgenacfmt` Redefine to include accessibility information.

```
10836 \renewcommand*{\glsgenacfmt}{%
10837   \ifdefempty\glscustomtext
10838     {%
10839       \ifglused\glslabel
10840       {%
```

Subsequent use:

```
10841     \glsifplural
10842     {%
```

Subsequent plural form:

```
10843     \glscapscase
10844     {%
```

Subsequent plural form, don't adjust case:

```
10845     \acronymfont
10846     {\glsshortpluralaccessdisplay
10847       {\glentryshortpl{\glslabel}}{\glslabel}}%
10848     \glsinsert
10849     }%
10850     {%
```

Subsequent plural form, make first letter upper case:

```
10851     \acronymfont
10852     {\glsshortpluralaccessdisplay
10853       {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10854     \glsinsert
10855     }%
10856     {%
```

Subsequent plural form, all caps:

```
10857     \mfirstucMakeUppercase
10858     {\acronymfont
10859     {\glsshortpluralaccessdisplay
10860       {\glentryshortpl{\glslabel}}{\glslabel}}%
10861     \glsinsert}%
10862     }%
10863     }%
10864     {%
```

Subsequent singular form

```
10865     \glscapscase
10866     {%
```

Subsequent singular form, don't adjust case:

```
10867     \acronymfont
10868     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10869     \glsinsert
10870     }%
10871     {%
```

Subsequent singular form, make first letter upper case:

```
10872      \acronymfont
10873      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10874      \glsinsert
10875      }%
10876      {%
```

Subsequent singular form, all caps:

```
10877      \mfirstucMakeUppercase
10878      {\acronymfont{%
10879      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10880      \glsinsert}%
10881      }%
10882      }%
10883      }%
10884      {%
```

First use:

```
10885      \glsifplural
10886      {%
```

First use plural form:

```
10887      \glscapscase
10888      {%
```

First use plural form, don't adjust case:

```
10889      \genplacrfullformat{\glslabel}{\glsinsert}%
10890      }%
10891      {%
```

First use plural form, make first letter upper case:

```
10892      \Genplacrfullformat{\glslabel}{\glsinsert}%
10893      }%
10894      {%
```

First use plural form, all caps:

```
10895      \mfirstucMakeUppercase
10896      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10897      }%
10898      }%
10899      {%
```

First use singular form

```
10900      \glscapscase
10901      {%
```

First use singular form, don't adjust case:

```
10902      \genacrfullformat{\glslabel}{\glsinsert}%
10903      }%
10904      {%
```

First use singular form, make first letter upper case:

```
10905      \Genacrfullformat{\glslabel}{\glsinsert}%
10906      }%
10907      {%
```

First use singular form, all caps:

```
10908      \mfirstucMakeUppercase
10909      {\genacrfullformat{\glslabel}{\glsinsert}}%
10910      }%
10911      }%
10912      }%
10913      }%
10914      {%
```

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.

```
10915      \glscustomtext
10916      }%
10917 }
```

enacrfullformat Redefine to include accessibility information.

```
10918 \renewcommand*{\genacrfullformat}[2]{%
10919   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10920   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
10921 }
```

enacrfullformat Redefine to include accessibility information.

```
10922 \renewcommand*{\Genacrfullformat}[2]{%
10923   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10924   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
10925 }
```

placrfullformat Redefine to include accessibility information.

```
10926 \renewcommand*{\genplacrfullformat}[2]{%
10927   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10928   (\glsshortpluralaccessdisplay
10929     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10930 }
```

placrfullformat Redefine to include accessibility information.

```
10931 \renewcommand*{\Genplacrfullformat}[2]{%
10932   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10933   (\glsshortpluralaccessdisplay
10934     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10935 }
```

\@acrshort

```
10936 \def\@acrshort#1#2[#3]{%
10937   \glsdoifexists{#2}%
```

```

10938 {%
10939   \let\do@gls@link@checkfirsthyper\relax

10940   \let\glsifplural\@secondoftwo
10941   \let\glscapscase\@firstofthree
10942   \let\glsinsert\@empty
10943   \def\glscustomtext{%
10944     \acronymfont{\glsshortaccessdisplay{\glentryshort{#2}}{#2}}#3%
10945   }%

   Call \@gls@link
10946   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10947 }%

10948 \glspostlinkhook
10949 }

```

\@Acrshort

```

10950 \def\@Acrshort#1#2[#3]{%
10951   \glsdoifexists{#2}%
10952   {%
10953     \let\do@gls@link@checkfirsthyper\relax

10954     \let\glsifplural\@secondoftwo
10955     \let\glscapscase\@secondofthree
10956     \let\glsinsert\@empty
10957     \def\glscustomtext{%
10958       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10959     }%

     Call \@gls@link
10960     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10961   }%

10962   \glspostlinkhook
10963 }

```

\@ACRshort

```

10964 \def\@ACRshort#1#2[#3]{%
10965   \glsdoifexists{#2}%
10966   {%
10967     \let\do@gls@link@checkfirsthyper\relax

10968     \let\glsifplural\@secondoftwo
10969     \let\glscapscase\@thirdofthree
10970     \let\glsinsert\@empty
10971     \def\glscustomtext{%
10972       \acronymfont{\glsshortaccessdisplay
10973         {\MakeUppercase{\glentryshort{#2}}}{#2}}#3%
10974     }%

```

```

Call \@gls@link
10975   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10976   }%

10977   \glspostlinkhook
10978 }

```

\@acrlong

```

10979 \def\@acrlong#1#2[#3]{%
10980   \glsdoifexists{#2}%
10981   {%
10982     \let\do@gls@link@checkfirsthyper\relax

10983     \let\glsifplural\@secondoftwo
10984     \let\glscapscase\@firstofthree
10985     \let\glsinsert\@empty
10986     \def\glscustomtext{%
10987       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10988     }%

```

```

Call \@gls@link
10989   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10990   }%

10991   \glspostlinkhook
10992 }

```

\@Acrlong

```

10993 \def\@Acrlong#1#2[#3]{%
10994   \glsdoifexists{#2}%
10995   {%
10996     \let\do@gls@link@checkfirsthyper\relax

10997     \let\glsifplural\@secondoftwo
10998     \let\glscapscase\@firstofthree
10999     \let\glsinsert\@empty
11000     \def\glscustomtext{%
11001       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
11002     }%

```

```

Call \@gls@link
11003   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11004   }%

11005   \glspostlinkhook
11006 }

```

\@ACRlong

```

11007 \def\@ACRlong#1#2[#3]{%
11008   \glsdoifexists{#2}%
11009   {%
11010     \let\do@gls@link@checkfirsthyper\relax

```



```

11011 \let\glsifplural\@secondoftwo
11012 \let\glsifscapscase\@firstofthree
11013 \let\glsinsert\@empty
11014 \def\glscustomtext{%
11015     \acronymfont{\glslongaccessdisplay{%
11016         \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
11017     }%

    Call \@gls@link
11018     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
11019 }%

11020 \glspostlinkhook
11021 }

```

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.

```

11022 \renewcommand*{\glossentryname}[1]{%
11023     \glsdoifexists{#1}%
11024     {%
11025         \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
11026     }%
11027 }

11028 \renewcommand*{\glossentryname}[1]{%
11029     \glsdoifexists{#1}%
11030     {%
11031         \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
11032     }%
11033 }

11034 \renewcommand*{\glossentrydesc}[1]{%
11035     \glsdoifexists{#1}%
11036     {%
11037         \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
11038     }%
11039 }

11040 \renewcommand*{\Glossentrydesc}[1]{%
11041     \glsdoifexists{#1}%
11042     {%
11043         \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
11044     }%
11045 }

```

```

11046 \renewcommand*{\glossentrysymbol}[1]{%
11047   \glsdoifexists{#1}%
11048   {%
11049     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
11050   }%
11051 }

11052 \renewcommand*{\Glossentrysymbol}[1]{%
11053   \glsdoifexists{#1}%
11054   {%
11055     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
11056   }%
11057 }

```

ssaryentryfield

```

11058 \newcommand*{\accsuppglossaryentryfield}[5]{%
11059   \glossaryentryfield{#1}%
11060   {\glsnameaccessdisplay{#2}{#1}}%
11061   {\glsdescriptionaccessdisplay{#3}{#1}}%
11062   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
11063 }

```

rysubentryfield

```

11064 \newcommand*{\accsuppglossarysubentryfield}[6]{%
11065   \glossarysubentryfield{#1}{#2}%
11066   {\glsnameaccessdisplay{#3}{#2}}%
11067   {\glsdescriptionaccessdisplay{#4}{#2}}%
11068   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
11069 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

11070 \renewacronymstyle{long-short}%
11071 {%

```

Check for long form in case this is a mixed glossary.

```

11072   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11073 }%
11074 {%
11075   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11076   \renewcommand*{\genacrfullformat}[2]{%
11077     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
11078     (\glsshortaccessdisplay
11079       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11080   }%
11081   \renewcommand*{\Genacrfullformat}[2]{%

```

```

11082 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11083 (\glsshortaccessdisplay
11084   {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
11085 }%
11086 \renewcommand*{\genplacrfullformat}[2]{%
11087   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11088   (\glsshortpluralaccessdisplay
11089     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11090 }%
11091 \renewcommand*{\Genplacrfullformat}[2]{%
11092   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11093   (\glsshortpluralaccessdisplay
11094     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11095 }%
11096 \renewcommand*{\acronymentry}[1]{%
11097   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
11098 \renewcommand*{\acronymsort}[2]{##1}%
11099 \renewcommand*{\acronymfont}[1]{##1}%
11100 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11101 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11102 }

```

short-long (*short*) (*long*) acronym style.

```

11103 \renewacronymstyle{short-long}%
11104 {%

```

Check for long form in case this is a mixed glossary.

```

11105 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11106 }%
11107 {%
11108 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11109 \renewcommand*{\genacrfullformat}[2]{%
11110   \glsshortaccessdisplay
11111     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
11112     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11113 }%
11114 \renewcommand*{\Genacrfullformat}[2]{%
11115   \glsshortaccessdisplay
11116     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
11117     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11118 }%
11119 \renewcommand*{\genplacrfullformat}[2]{%
11120   \glsshortpluralaccessdisplay
11121     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
11122     (\glslongpluralaccessdisplay
11123       {\glsentrylongpl{##1}}{##1}})%
11124 }%
11125 \renewcommand*{\Genplacrfullformat}[2]{%
11126   \glsshortpluralaccessdisplay
11127     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

11128 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11129 }%
11130 \renewcommand*{\acronymentry}[1]{%
11131   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11132 \renewcommand*{\acronymsort}[2]{##1}%
11133 \renewcommand*{\acronymfont}[1]{##1}%
11134 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11135 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11136 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

11137 \renewacronymstyle{long-short-desc}%
11138 {%
11139   \GlsUseAcrEntryDispStyle{long-short}%
11140 }%
11141 {%
11142   \GlsUseAcrStyleDefs{long-short}%
11143   \renewcommand*{\GenericAcronymFields}{}%
11144   \renewcommand*{\acronymsort}[2]{##2}%
11145   \renewcommand*{\acronymentry}[1]{%
11146     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11147     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11148 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11149 \renewacronymstyle{long-sc-short-desc}%
11150 {%
11151   \GlsUseAcrEntryDispStyle{long-sc-short}%
11152 }%
11153 {%
11154   \GlsUseAcrStyleDefs{long-sc-short}%
11155   \renewcommand*{\GenericAcronymFields}{}%
11156   \renewcommand*{\acronymsort}[2]{##2}%
11157   \renewcommand*{\acronymentry}[1]{%
11158     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11159     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11160 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11161 \renewacronymstyle{long-sm-short-desc}%
11162 {%
11163   \GlsUseAcrEntryDispStyle{long-sm-short}%
11164 }%
11165 {%
11166   \GlsUseAcrStyleDefs{long-sm-short}%
11167   \renewcommand*{\GenericAcronymFields}{}%

```

```

11168 \renewcommand*{\acronymsort}[2]{##2}%
11169 \renewcommand*{\acronymentry}[1]{%
11170   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11171   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11172 }

```

short-long-desc *(short)* (*{<long>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11173 \renewacronymstyle{short-long-desc}%
11174 {%
11175   \GlsUseAcrEntryDispStyle{short-long}%
11176 }%
11177 {%
11178   \GlsUseAcrStyleDefs{short-long}%
11179   \renewcommand*{\GenericAcronymFields}{}%
11180   \renewcommand*{\acronymsort}[2]{##2}%
11181   \renewcommand*{\acronymentry}[1]{%
11182     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11183     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11184 }

```

short-long-desc *(long)* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11185 \renewacronymstyle{sc-short-long-desc}%
11186 {%
11187   \GlsUseAcrEntryDispStyle{sc-short-long}%
11188 }%
11189 {%
11190   \GlsUseAcrStyleDefs{sc-short-long}%
11191   \renewcommand*{\GenericAcronymFields}{}%
11192   \renewcommand*{\acronymsort}[2]{##2}%
11193   \renewcommand*{\acronymentry}[1]{%
11194     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11195     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11196 }

```

short-long-desc *(long)* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11197 \renewacronymstyle{sm-short-long-desc}%
11198 {%
11199   \GlsUseAcrEntryDispStyle{sm-short-long}%
11200 }%
11201 {%
11202   \GlsUseAcrStyleDefs{sm-short-long}%
11203   \renewcommand*{\GenericAcronymFields}{}%
11204   \renewcommand*{\acronymsort}[2]{##2}%
11205   \renewcommand*{\acronymentry}[1]{%
11206     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11207     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

11208 }

dua *<long>* only acronym style.

11209 \renewacronymstyle{dua}%

11210 {%

Check for long form in case this is a mixed glossary.

11211 \ifdefempty\glscustomtext

11212 {%

11213 \ifglshaslong{\glslabel}%

11214 {%

11215 \glssifplural

11216 {%

Plural form:

11217 \glscapscase

11218 {%

Plural form, don't adjust case:

11219 \glslongpluralaccessdisplay{\glssentrylongpl{\glslabel}}{\glslabel}%

11220 \glssinsert

11221 }%

11222 {%

Plural form, make first letter upper case:

11223 \glslongpluralaccessdisplay{\Glssentrylongpl{\glslabel}}{\glslabel}%

11224 \glssinsert

11225 }%

11226 {%

Plural form, all caps:

11227 \glslongpluralaccessdisplay

11228 {\mfirstucMakeUppercase{\glssentrylongpl{\glslabel}}}{\glslabel}%

11229 \mfirstucMakeUppercase{\glssinsert}%

11230 }%

11231 }%

11232 {%

Singular form

11233 \glscapscase

11234 {%

Singular form, don't adjust case:

11235 \glslongaccessdisplay{\glssentrylong{\glslabel}}{\glslabel}\glssinsert

11236 }%

11237 {%

Subsequent singular form, make first letter upper case:

11238 \glslongaccessdisplay{\Glssentrylong{\glslabel}}{\glslabel}\glssinsert

11239 }%

11240 {%

Subsequent singular form, all caps:

```

11241      \glslongaccessdisplay
11242      {\mfirstucMakeUppercase
11243       {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11244      \mfirstucMakeUppercase{\glsinsert}%
11245      }%
11246      }%
11247      }%
11248      {%

```

Not an acronym:

```

11249      \glsgenentryfmt
11250      }%
11251      }%
11252      {\glscustomtext\glsinsert}%
11253      }%
11254      {%
11255      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11256      \renewcommand*{\acrfullfmt}[3]{%
11257        \glslink[##1]{##2}{%
11258          \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11259          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11260      \renewcommand*{\Acrfullfmt}[3]{%
11261        \glslink[##1]{##2}{%
11262          \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11263          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11264      \renewcommand*{\ACRfullfmt}[3]{%
11265        \glslink[##1]{##2}{%
11266          \glslongaccessdisplay
11267          {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11268          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11269      \renewcommand*{\acrfullplfmt}[3]{%
11270        \glslink[##1]{##2}{%
11271          \glslongpluralaccessdisplay
11272          {\glsentrylongpl{##2}}{##2}##3\space
11273          (\glsshortpluralaccessdisplay
11274          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11275      \renewcommand*{\ACRfullplfmt}[3]{%
11276        \glslink[##1]{##2}{%
11277          \glslongpluralaccessdisplay
11278          {\Glsentrylongpl{##2}}{##2}##3\space
11279          (\glsshortpluralaccessdisplay
11280          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11281      \renewcommand*{\ACRfullplplfmt}[3]{%
11282        \glslink[##1]{##2}{%
11283          \glslongpluralaccessdisplay
11284          {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11285          (\glsshortpluralaccessdisplay
11286          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11287      \renewcommand*{\glsentryfull}[1]{%

```

```

11288 \glslongaccessdisplay{\glsentrylong{##1}}\space
11289 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11290 }%
11291 \renewcommand*{\Glsentryfull}[1]{%
11292 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11293 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11294 }%
11295 \renewcommand*{\glsentryfullpl}[1]{%
11296 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11297 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11298 }%
11299 \renewcommand*{\Glsentryfullpl}[1]{%
11300 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11301 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11302 }%
11303 \renewcommand*{\acronymentry}[1]{%
11304 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11305 \renewcommand*{\acronymsort}[2]{##1}%
11306 \renewcommand*{\acronymfont}[1]{##1}%
11307 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11308 }

```

dua-desc <long> only acronym style with user-supplied description.

```

11309 \renewacronymstyle{dua-desc}%
11310 {%
11311 \GlsUseAcrEntryDispStyle{dua}%
11312 }%
11313 {%
11314 \GlsUseAcrStyleDefs{dua}%
11315 \renewcommand*{\GenericAcronymFields}{}%
11316 \renewcommand*{\acronymentry}[1]{%
11317 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11318 \renewcommand*{\acronymsort}[2]{##2}%
11319 }%

```

footnote <short>\footnote{<long>} acronym style.

```

11320 \renewacronymstyle{footnote}%
11321 {%
11322 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11323 }%
11324 {%
11325 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11326 \glshyperfirstfalse
11327 \renewcommand*{\genacrfullformat}[2]{%
11328 \glsshortaccessdisplay
11329 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```



```

11330 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11331 }%
11332 \renewcommand*{\Genacrfullformat}[2]{%
11333 \glsshortaccessdisplay
11334   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11335 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11336 }%
11337 \renewcommand*{\genplacrfullformat}[2]{%
11338 \glsshortpluralaccessdisplay
11339   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
11340 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11341 }%
11342 \renewcommand*{\Genplacrfullformat}[2]{%
11343 \glsshortpluralaccessdisplay
11344   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11345 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11346 }%
11347 \renewcommand*{\acronymentry}[1]{%
11348 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11349 \renewcommand*{\acronymsort}[2]{##1}%
11350 \renewcommand*{\acronymfont}[1]{##1}%
11351 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11352 \renewcommand*{\acrfullfmt}[3]{%
11353 \glslink{##1}{##2}{%
11354 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11355 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11356 \renewcommand*{\Acrfullfmt}[3]{%
11357 \glslink{##1}{##2}{%
11358 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11359 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11360 \renewcommand*{\ACRfullfmt}[3]{%
11361 \glslink{##1}{##2}{%
11362 \glsshortaccessdisplay
11363   {\mfirstucMakeUppercase
11364   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11365   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11366 \renewcommand*{\acrfullplfmt}[3]{%
11367 \glslink{##1}{##2}{%
11368 \glsshortpluralaccessdisplay
11369   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11370   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11371 \renewcommand*{\Acrfullplfmt}[3]{%
11372 \glslink{##1}{##2}{%
11373 \glsshortpluralaccessdisplay
11374   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11375   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11376 \renewcommand*{\ACRfullplfmt}[3]{%
11377 \glslink{##1}{##2}{%

```

```

11378 \glsshortpluralaccessdisplay
11379 {\mfirstucMakeUppercase
11380 {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
11381 (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

11382 \renewcommand*{\glentryfull}[1]{%
11383 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
11384 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11385 \renewcommand*{\Glsentryfull}[1]{%
11386 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
11387 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11388 \renewcommand*{\glentryfullpl}[1]{%
11389 \glsshortpluralaccessdisplay
11390 {\acronymfont{\glentryshortpl{##1}}{##1}\space
11391 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11392 \renewcommand*{\Glsentryfullpl}[1]{%
11393 \glsshortpluralaccessdisplay
11394 {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
11395 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11396 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11397 \renewacronymstyle{footnote-sc}%
11398 {%
11399 \GlsUseAcrEntryDispStyle{footnote}%
11400 }%
11401 {%
11402 \GlsUseAcrStyleDefs{footnote}%
11403 \renewcommand{\acronymentry}[1]{%
11404 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11405 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11406 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11407 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11408 \renewacronymstyle{footnote-sm}%
11409 {%
11410 \GlsUseAcrEntryDispStyle{footnote}%
11411 }%
11412 {%
11413 \GlsUseAcrStyleDefs{footnote}%
11414 \renewcommand{\acronymentry}[1]{%
11415 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11416 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11417 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11418 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11419 \renewacronymstyle{footnote-desc}%
11420 {%
11421   \GlsUseAcrEntryDisplayStyle{footnote}%
11422 }%
11423 {%
11424   \GlsUseAcrStyleDefs{footnote}%
11425   \renewcommand*{\GenericAcronymFields}{}%
11426   \renewcommand*{\acronymsort}[2]{##2}%
11427   \renewcommand*{\acronymentry}[1]{%
11428     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11429     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11430 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11431 \renewacronymstyle{footnote-sc-desc}%
11432 {%
11433   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11434 }%
11435 {%
11436   \GlsUseAcrStyleDefs{footnote-sc}%
11437   \renewcommand*{\GenericAcronymFields}{}%
11438   \renewcommand*{\acronymsort}[2]{##2}%
11439   \renewcommand*{\acronymentry}[1]{%
11440     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11441     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11442 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11443 \renewacronymstyle{footnote-sm-desc}%
11444 {%
11445   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11446 }%
11447 {%
11448   \GlsUseAcrStyleDefs{footnote-sm}%
11449   \renewcommand*{\GenericAcronymFields}{}%
11450   \renewcommand*{\acronymsort}[2]{##2}%
11451   \renewcommand*{\acronymentry}[1]{%
11452     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11453     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11454 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11455 \renewcommand*{\newacronymhook}{%
11456   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11457     \the\glskeylisttok}%
11458   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11459 }

1tNewAcronymDef Modify default style to use access text:

```
11460 \renewcommand*{\DefaultNewAcronymDef}{%
11461   \edef\@do@newglossaryentry{%
11462     \noexpand\newglossaryentry{\the\glslabeltok}%
11463     {%
11464       type=\acronymtype,%
11465       name={\the\glsshorttok},%
11466       description={\the\glslongtok},%
11467       descriptionaccess=\relax,%
11468       text={\the\glsshorttok},%
11469       access={\noexpand\@glo@textaccess},%
11470       sort={\the\glsshorttok},%
11471       short={\the\glsshorttok},%
11472       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11473       shortaccess={\the\glslongtok},%
11474       long={\the\glslongtok},%
11475       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11476       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11477       first={\noexpand\glslongaccessdisplay
11478         {\the\glslongtok}{\the\glslabeltok}\space
11479         (\noexpand\glsshortaccessdisplay
11480           {\the\glsshorttok}{\the\glslabeltok})},%
11481       plural={\the\glsshorttok\acrpluralsuffix},%
11482       firstplural={\noexpand\glslongpluralaccessdisplay
11483         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11484         (\noexpand\glsshortpluralaccessdisplay
11485           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11486       firstaccess=\relax,%
11487       firstpluralaccess=\relax,%
11488       textaccess={\noexpand\@glo@shortaccess},%
11489       \the\glskeylisttok
11490     }%
11491   }%
11492   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11493   \let\@org@gls@assign@plural\gls@assign@plural
11494   \let\@org@gls@assign@descplural\gls@assign@descplural
11495   \def\gls@assign@firstpl##1##2{%
11496     \@@gls@expand@field{##1}{firstpl}{##2}%
11497   }%
11498   \def\gls@assign@plural##1##2{%
11499     \@@gls@expand@field{##1}{plural}{##2}%
11500   }%
11501   \def\gls@assign@descplural##1##2{%
11502     \@@gls@expand@field{##1}{descplural}{##2}%
11503   }%
11504   \@do@newglossaryentry
11505   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11506 \let\gls@assign@plural\@org@gls@assign@plural
11507 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11508 }

```

teNewAcronymDef

```

11509 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11510 \edef\@do@newglossaryentry{%
11511 \noexpand\newglossaryentry{\the\glslabeltok}%
11512 {%
11513 type=\acronymtype,%
11514 name={\noexpand\acronymfont{\the\glsshorttok}},%
11515 sort={\the\glsshorttok},%
11516 text={\the\glsshorttok},%
11517 short={\the\glsshorttok},%
11518 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11519 shortaccess={\the\glslongtok},%
11520 long={\the\glslongtok},%
11521 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11522 access={\noexpand\@glo@textaccess},%
11523 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11524 symbol={\the\glslongtok},%
11525 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11526 firstpluralaccess=\relax,
11527 textaccess={\noexpand\@glo@shortaccess},%
11528 \the\glskeylisttok
11529 }%
11530 }%
11531 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11532 \let\@org@gls@assign@plural\gls@assign@plural
11533 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11534 \def\gls@assign@firstpl##1##2{%
11535 \@@gls@expand@field{##1}{firstpl}{##2}%
11536 }%
11537 \def\gls@assign@plural##1##2{%
11538 \@@gls@expand@field{##1}{plural}{##2}%
11539 }%
11540 \def\gls@assign@symbolplural##1##2{%
11541 \@@gls@expand@field{##1}{symbolplural}{##2}%
11542 }%
11543 \do@newglossaryentry
11544 \let\gls@assign@plural\@org@gls@assign@plural
11545 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11546 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11547 }

```

onNewAcronymDef

```

11548 \renewcommand*{\DescriptionNewAcronymDef}{%
11549 \edef\@do@newglossaryentry{%
11550 \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11551 {%
11552     type=\acronymtype,%
11553     name={\noexpand
11554         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11555     access={\noexpand\@glo@textaccess},%
11556     sort={\the\glsshorttok},%
11557     short={\the\glsshorttok},%
11558     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11559     shortaccess={\the\glslongtok},%
11560     long={\the\glslongtok},%
11561     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11562     first={\the\glslongtok},%
11563     firstaccess=\relax,
11564     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11565     text={\the\glsshorttok},%
11566     textaccess={\the\glslongtok},%
11567     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11568     symbol={\noexpand\@glo@text},%
11569     symbolaccess={\noexpand\@glo@textaccess},%
11570     symbolplural={\noexpand\@glo@plural},%
11571     firstpluralaccess=\relax,
11572     textaccess={\noexpand\@glo@shortaccess},%
11573     \the\glskeylisttok}%
11574 }%
11575 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11576 \let\@org@gls@assign@plural\gls@assign@plural
11577 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11578 \def\gls@assign@firstpl##1##2{%
11579     \@@gls@expand@field{##1}{firstpl}{##2}%
11580 }%
11581 \def\gls@assign@plural##1##2{%
11582     \@@gls@expand@field{##1}{plural}{##2}%
11583 }%
11584 \def\gls@assign@symbolplural##1##2{%
11585     \@@gls@expand@field{##1}{symbolplural}{##2}%
11586 }%
11587 \do@newglossaryentry
11588 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11589 \let\gls@assign@plural\@org@gls@assign@plural
11590 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11591 }

```

teNewAcronymDef

```

11592 \renewcommand*{\FootnoteNewAcronymDef}{%
11593     \edef\@do@newglossaryentry{%
11594         \noexpand\newglossaryentry{\the\glslabeltok}%
11595         {%
11596             type=\acronymtype,%
11597             name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11598     sort={\the\glsshorttok},%
11599     text={\the\glsshorttok},%
11600     textaccess={\the\glslongtok},%
11601     access={\noexpand\@glo@textaccess},%
11602     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11603     short={\the\glsshorttok},%
11604     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11605     long={\the\glslongtok},%
11606     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11607     description={\the\glslongtok},%
11608     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11609     \the\glskeylisttok
11610   }%
11611 }%
11612 \let\@org@gls@assign@plural\gls@assign@plural
11613 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11614 \let\@org@gls@assign@descplural\gls@assign@descplural
11615 \def\gls@assign@firstpl##1##2{%
11616   \@@gls@expand@field{##1}{firstpl}{##2}%
11617 }%
11618 \def\gls@assign@plural##1##2{%
11619   \@@gls@expand@field{##1}{plural}{##2}%
11620 }%
11621 \def\gls@assign@descplural##1##2{%
11622   \@@gls@expand@field{##1}{descplural}{##2}%
11623 }%
11624 \do@newglossaryentry
11625 \let\gls@assign@plural\@org@gls@assign@plural
11626 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11627 \let\gls@assign@descplural\@org@gls@assign@descplural
11628 }

```

11NewAcronymDef

```

11629 \renewcommand*{\SmallNewAcronymDef}{%
11630   \edef\@do@newglossaryentry{%
11631     \noexpand\newglossaryentry{\the\glslabeltok}%
11632     {%
11633       type=\acronymtype,%
11634       name={\noexpand\acronymfont{\the\glsshorttok}},%
11635       access={\noexpand\@glo@symbolaccess},%
11636       sort={\the\glsshorttok},%
11637       short={\the\glsshorttok},%
11638       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11639       shortaccess={\the\glslongtok},%
11640       long={\the\glslongtok},%
11641       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11642       text={\noexpand\@glo@short},%
11643       textaccess={\noexpand\@glo@shortaccess},%
11644       plural={\noexpand\@glo@shortpl},%

```

```

11645     first={\the\glslongtok},%
11646     firstaccess=\relax,
11647     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11648     description={\noexpand\@glo@first},%
11649     descriptionplural={\noexpand\@glo@firstplural},%
11650     symbol={\the\glsshorttok},%
11651     symbolaccess={\the\glslongtok},%
11652     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11653     \the\glskeylisttok
11654   }%
11655 }%
11656 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11657 \let\@org@gls@assign@plural\gls@assign@plural
11658 \let\@org@gls@assign@descplural\gls@assign@descplural
11659 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11660 \def\gls@assign@firstpl##1##2{%
11661   \@@gls@expand@field{##1}{firstpl}{##2}%
11662 }%
11663 \def\gls@assign@plural##1##2{%
11664   \@@gls@expand@field{##1}{plural}{##2}%
11665 }%
11666 \def\gls@assign@descplural##1##2{%
11667   \@@gls@expand@field{##1}{descplural}{##2}%
11668 }%
11669 \def\gls@assign@symbolplural##1##2{%
11670   \@@gls@expand@field{##1}{symbolplural}{##2}%
11671 }%
11672 \do@newglossaryentry
11673 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11674 \let\gls@assign@plural\@org@gls@assign@plural
11675 \let\gls@assign@descplural\@org@gls@assign@descplural
11676 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11677 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
11678 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
11679 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
11680 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
11681 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```


5.5 Debugging Commands

owglongnameaccess

```
11682 \newcommand*{\showglongnameaccess}[1]{%
11683   \expandafter\show\csname glo@\glsdetoklabel{#1}@access\endcsname
11684 }
```

owglotextaccess

```
11685 \newcommand*{\showglotextaccess}[1]{%
11686   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11687 }
```

glopluralaccess

```
11688 \newcommand*{\showglopluralaccess}[1]{%
11689   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11690 }
```

wglofirstaccess

```
11691 \newcommand*{\showwglofirstaccess}[1]{%
11692   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11693 }
```

rstpluralaccess

```
11694 \newcommand*{\showglofirstpluralaccess}[1]{%
11695   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11696 }
```

glosymbolaccess

```
11697 \newcommand*{\showglosymbolaccess}[1]{%
11698   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11699 }
```

bolpluralaccess

```
11700 \newcommand*{\showglosymbolpluralaccess}[1]{%
11701   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11702 }
```

owglodescaccess

```
11703 \newcommand*{\showglodescaccess}[1]{%
11704   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11705 }
```

escpluralaccess

```
11706 \newcommand*{\showglodescpluralaccess}[1]{%
11707   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11708 }
```

wgloshortaccess

```
11709 \newcommand*{\showgloshortaccess}[1]{%  
11710   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11711 }
```

ortpluralaccess

```
11712 \newcommand*{\showgloshortpluralaccess}[1]{%  
11713   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11714 }
```

owglolongaccess

```
11715 \newcommand*{\showglolongaccess}[1]{%  
11716   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11717 }
```

ongpluralaccess

```
11718 \newcommand*{\showglolongpluralaccess}[1]{%  
11719   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11720 }
```

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.

```
11721 \NeedsTeXFormat{LaTeX2e}
11722 \ProvidesPackage{glossaries-babel}[2018/05/10 v4.38 (NLCT)]
```

Load tracklang to obtain language settings.

```
11723 \RequirePackage{tracklang}
11724 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11725 \AnyTrackedLanguages
11726 {%
11727   \ForEachTrackedDialect{\this@dialect}{%
11728     \IfTrackedLanguageFileExists{\this@dialect}%
11729       {glossaries-}% prefix
11730       {.ldf}%
11731       {%
11732         \RequireGlossariesLang{\CurrentTrackedTag}%
11733       }%
11734       {%
11735         \PackageWarningNoLine{glossaries}%
11736           {No language module detected for ‘\this@dialect’.\MessageBreak
11737             Language modules need to be installed separately.\MessageBreak
11738             Please check on CTAN for a bundle called\MessageBreak
11739             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11740       }%
11741     }%
11742   }%
11743 }
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11744 \NeedsTeXFormat{LaTeX2e}
11745 \ProvidesPackage{glossaries-polyglossia}[2018/05/10 v4.38 (NLCT)]
```

Load tracklang to obtain language settings.

```
11746 \RequirePackage{tracklang}
11747 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11748 \AnyTrackedLanguages
```

```

11749 {%
11750     \ForEachTrackedDialect{\this@dialect}{%
11751         \IfTrackedLanguageFileExists{\this@dialect}%
11752         {glossaries-}% prefix
11753         {.ldf}%
11754         {%
11755             \RequireGlossariesLang{\CurrentTrackedTag}%
11756         }%
11757         {%
11758             \PackageWarningNoLine{glossaries}%
11759             {No language module detected for ‘\this@dialect’.\MessageBreak
11760             Language modules need to be installed separately.\MessageBreak
11761             Please check on CTAN for a bundle called\MessageBreak
11762             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11763         }%
11764     }%
11765 }%
11766 {}%

```

Glossary

`makeindex` An indexing application. [9](#), [13](#), [29](#), [30](#), [179](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [9](#), [13](#), [29](#), [30](#), [179](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 7
General: Added range facility in format key 114	1.12 (2008-03-08)
\writeist: Added spaces after \delimN and \delimR in ist file 161	\@GLSpl: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 128
1.04 (2007-08-03)	\@Glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 127
General: Added \glstextformat 98	\@glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 126
1.05 (2007-08-10)	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 122
\glossarysection: added \@mkboth to \glossarysection 41	descriptionplural: new 64
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 82	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 82
1.07 (2007-09-13)	descriptionplural support added 82
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 112	symbolplural support added 82
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 158	\Glsentrydescplural: New 151
1.08 (2007-10-13)	\glentrydescplural: New 151
General: Added babel support 35	\Glsentrysymbolplural: New 152
listgroup: changed listgroup style to use \glsgetgrouptitle 274	\glentrysymbolplural: New 152
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 275	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 240
1.1 (2008-02-22)	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 246
\@glossarysection: numbered sections and auto label added 42	symbolplural: new 65
\@gls@tmpb: changed \toksdef to \newtoks 116	
\@gls@toc: numberline added 44	
\@p@glossarysection: numbered sections and auto label added 42	
General: amsgen now loaded (\new@ifnextchar needed) 4	
translate: translate option added 26	
\setglossarysection: new 42	
numberedsection: numberedsection package option added 8	

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	129–137
\ACRfullpl: new	221
\Acrfullpl: new	221
\acrfullpl: new	220
\acrpluralsuffix: New	218
\gls@defglossaryentry: Changed default first value	82
Changed default firstplural value	82
Removed restriction on only using \newglossaryentry in the preamble	87
\newacronym: Removed restriction on only using \newacronym in the preamble	218
1.14 (2008-06-17)	
\@gls@hypergroup: new	269
General: added nonumberlist key to \printglossary	205
added numberedsection key to \printglossary	204
\firstacronymfont: new	222
\glsautoprefix: new	7
\glsnavhyperlink: changed \edef to \protected@edef	268
\glsnavhypertarget: added write to aux file	268
\glsnavigation: changed to only use labels for groups that are present ..	270
1.15 (2008-08-15)	
\@gls@link: added \glslabel	112
\gls@defglossaryentry: check for \@glo@first in description	86
check for \@glo@text in symbol	86
\gls@hypergroup: new	269
\glsnavhypertarget: added check if rerun required	268
\glssettoctitle: new	34
\printglossary: changed the way the TOC title is set	190
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	125
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	128
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	125
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	127
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	124
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	129
\@glspl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	126
\@glstarget: raised the hypertarget so the target text doesn't scroll off the top of the page	122
\gls@defglossaryentry: Changed def to let	82
1.17 (2008-12-26)	
\@do@esc@wrglossary: new	183
\@do@seeglossary: new	187
\@glo@storeentry: new	88
\@gls@glossary: changed definition to use \index instead of \@index	180
\@glsdefaultplural: new	69
\@glsdefaultsort: new	69
\@glshypernumber: new	215
\@glsnoname: new	68
\@glsnonextpages: new	205
General: added xindy support	29
parent: new	66
see: new	66
\gls@defglossaryentry: added nonumberlist key	82
added parent key	82
added see key	82
Stored main part of entry format when entry is defined	87
\gls@suffixF: new	39
\gls@suffixFF: new	39
\gls@wrglossary: modified to allow for xindy support	180

\glshyperlink: new	157	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	240
\glshypernumber: modified to allow material to be attached to location	215	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	246
\glshnavhyperlink: replaced \hyperlink to \@glslink	268	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	249
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	268	2.01 (2009 May 30) \@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	113
\glsssee: new	188	\forall glossaries: replaced \ifthenelse with \ifx	53
\glssseeformat: new	188	\forall glossentries: replaced \ifthenelse with \ifx	53
\glssSetSuffixF: new	39	\glssdefmain: new	16
\glssSetSuffixFF: new	39	\glssdescwidth: changed \linewidth to \hsize	276, 298
\ifglssxindy: new	29	\glsslistdottedwidth: changed \linewidth to \hsize	276
\istfilename: added xindy support ...	38	\glsspagelistwidth: changed \linewidth to \hsize	276, 298
\newglossarystyle: made \newglossarystyle long	214	nomain: added nomain package option	16
\nopostdesc: new	37	\writeist: removed item_02 - no such makeindex key	165
nonumberlist: new	66	2.02 (2007-07-13) \@printglossary: suppressed warning globally rather than locally	192
\printglossary: added check to determine if \printglossary is already defined	190	2.02 (2009-07-13) \glossarysection: changed \@mkboth to \glossarymark	41
added print language to aux file	190	\glssglossarymark: New	41
order: order package option added	28	2.03 (2009-09-23) \@GLS@: Added check for hyperfirst	125
\writeist: added xindy support	161	\@GLSpl: Added check for hyperfirst	128
1.18 (2009-01-14) \@glsl@loadlist: new	10	\@GLS@: Added check for hyperfirst	125
\@glsl@loadlong: new	9	\@GLspl@: Added check for hyperfirst	127
\@glsl@loadsuper: new	10	\@glsl@: Added check for hyperfirst	124
\@glsl@loadtree: new	10	\@glsl@@link: new	111
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort	82	\@glsl@link: added \leavevmode	112
moved sort sanitization to \newglossaryentry	86	Moved entry existence check to avoid duplicate code	112
\glstarget: new	208	\@glsl@disp: Added check for hyperfirst	129
\oldacronym: new	217	\@glspl@: Added check for hyperfirst	126
nolist: new	10	\glssglossarymark: Added check to see if it's already defined	41
nolong: new	9	hyperfirst: new	27
sort: moved sanitization to \newglossaryentry	64		
nostyles: new	10		
nosuper: new	10		
notree: new	10		
1.19 (2009-03-02) \glsclearpage: new	43		
\glssdisp: new	128		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	244		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	125
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms	128
\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms	125
\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms	127
\@glossaryentryfield: new	88
\@glossarysubentryfield: new	88
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	124
\@glsacronymlists: new	17
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms	129
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	126
\@newglossaryentryposthook: new ..	87
\@newglossaryentryprehook: new ...	87
acronymlists: new	19
\DeclareAcronymList: new	18
\DefineAcronymSynonyms: new	235
\gls@defglossaryentry: added user1-6 keys	82
\glsadd: fixed bug that ignored counter	158
\Glsentryuseri: new	153
\glsentryuseri: new	153
\Glsentryuserii: new	154
\glsentryuserii: new	154
\Glsentryuseriii: new	154
\glsentryuseriii: new	154
\Glsentryuseriv: new	154
\glsentryuseriv: new	154
\Glsentryuserv: new	154
\glsentryuserv: new	154
\Glsentryuservi: new	155
\glsentryuservi: new	155
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	61
\SetAcronymLists: new	19
\SetDefaultAcronymDisplayStyle: new	237
\SetDefaultAcronymStyle: new	238
\SetDescriptionAcronymDisplayStyle: new	242
\SetDescriptionDUAAcronymDisplayStyle: new	241
\SetDescriptionFootnoteAcronymDisplayStyle: new	238
\SetDUADisplayStyle: new	250
\SetFootnoteAcronymDisplayStyle: new	245
\SetSmallAcronymDisplayStyle: new	247
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco	128
Removed spurious brace. Patch provided by Sergiu Dotenco	129
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	166
2.06 (2010-06-14)	
\altnewglossary: new	62
\CustomAcronymFields: new	252
\CustomNewAcronymDef: new	252
\SetCustomDisplayStyle: new	252
\SetCustomStyle: new	253
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	158
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites	170
\gls@wrglossary: modified to take into account savewrites	180
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument	114
3.0 (2011-04-02)	
\@do@esc@wrglossary: added check for hyper location prefix	185
modified to use new format	183
\@@glossarysec: replaced \@ifundefined with \ifcsundef ...	7
\@do@seeglossary: Sanitize and escape cross-referencing information	187
\@gls@counterwithin: new	12

\@gls@ifinlist: new	44	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter	158
\@gls@saveentrycounter	113	\GlsAddXdyCounters: new	45
added \@gls@setsort	113	\glseentrycounterlabel: new	207
\@gls@saveentrycounter: new	113	\glseentryitem: new	207
\@gls@setupsort@def: new	14	\glseentrylong: new	155
\@gls@setupsort@standard: new	13	\glseentrylong: new	155
\@gls@setupsort@use: new	14	\glseentrylongpl: new	155
\@gls@xdy@locationlist: new	47	\glseentrylongpl: new	155
\@glslink: replaced \@ifundefined		\glseentryshort: new	155
with \ifcsundef	122	\glseentryshort: new	155
\@glsnextpages: new	206	\glseentryshortpl: new	155
\@print@glossary: replaced		\glseentryshortpl: new	155
\@ifundefined with \ifcsundef ..	193	\glsgetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	212
\currentglossary	192	\gls glossarymark: replaced	
added \glsnextpages	192	\@ifundefined with \ifcsundef ..	41
make toctitle default to title	191	\glshyperlink: changed default from	
\@xdyattributelist: new	44	\glseentryname to \glseentrytext	157
General: added prefix to hyperlink	216	\glshypernumber: replaced	
etoolbox now loaded	4	\@ifundefined with \ifcsundef ..	215
replaced \@ifundefined with		\glsnumberformat: replaced	
\ifcsundef	33, 36, 109, 204	\@ifundefined with \ifcsundef ..	39
\acrfootnote: new	238	\glsrefentry: new	207
\ACRfull: added starred version	220	\glsresetsubentrycounter: new ...	206
\Acrfull: added starred version	220	\glsseeitem: hyperlink uses	
\acrfull: added starred version	219	\glsseeitemformat instead of	
\ACRfullpl: added starred version ...	221	\glseentryname	189
\Acrfullpl: added starred version ...	221	\glsseeitemformat: new	189
\acrfullpl: added starred version ...	220	\glssortnumberfmt: new	13
\acrlinkfootnote: new	238	\glsstepentry: new	206
\acrlinkfootnote: new	238	\glsstepsubentry: new	207
savewrites: new	30	\glssubentrycounterlabel: new ...	207
see: added \@glo@seeautonumberlist	66	\glssubentryitem: new	208
seeautonumberlist: new	9	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef	208
\@ifundefined with \ifcsundef ..	41	short: new	68
\glossarystyle: replaced		shortplural: new	68
\@ifundefined with \ifcsundef ..	213	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	54
\@ifundefined with \ifcsundef ..	29	\ifglseentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	54
\@gls@defsort	86	\istfile: deprecated	178
added short and long keys	83	glossaryentry: new	11
replaced \@ifundefined with		glossarysubentry: new	12
\ifcsundef	83	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	43	\newrobustcmd	71

\newglossarystyle: replaced	
\@ifundefined with \ifcsundef .	214
\ns@newglossary: added	
\@gls@defsortcount	62
replaced \@ifundefined with	
\ifcsundef	61
entrycounter: new	11
\oldacronym: replaced \@ifundefined	
with \ifcsundef	217
compatible-2.07: compatible-2.07	
option added	30
long: new	68
longplural: new	68
nonumberlist: now boolean	66
sort: new	12
counter: replaced \@ifundefined with	
\ifcsundef	65
counterwithin: new	11
\printglossary: replaced	
\@ifundefined with \ifcsundef .	190
\SetDescriptionFootnoteAcronymDisplayStyle	
expanded options link options	238
\setentrycounter: added optional	
argument	213
\showacronymlists: new	258
\showglocounter: new	255
\showgloDESC: new	256
\showgloDESCplural: new	257
\showglofirst: new	255
\showglofirstpl: new	255
\showgloflag: new	258
\showgloindex: new	258
\showglolevel: new	254
\showglongname: new	256
\showgloparent: new	254
\showgloplural: new	254
\showglosort: new	257
\showglossaries: new	258
\showglossarycounter: new	259
\showglossaryentries: new	259
\showglossaryin: new	259
\showglossaryout: new	259
\showglossarytitle: new	259
\showglosymbol: new	257
\showglosymbolplural: new	257
\showgloTEXT: new	254
\showgloTYPE: new	255
\showglouserI: new	255
\showglouserII: new	255
\showglouserIII: new	256
\showglouserIV: new	256
\showglouserV: new	256
\showglouserVI: new	256
subentrycounter: new	12
\writeist: added xindy-only macro	
definitions to glossary open tag	163
modified to support new format	161
3.01 (2011-04-12)	
\@glswritefiles: added check for	
empty glossaries	179
General: made robust	125
\ACRfull: made robust	220
\Acrfull: made robust	219
\acrfull: made robust	219
\acrfullformat: removed	
\acronymfont as it should already be	
set in the second argument.	219
\ACRfullpl: made robust	221
\Acrfullpl: made robust	221
\acrfullpl: made robust	220
\ACRlong: made robust	146
\Acrlong: made robust	146
\acrlong: made robust	145
\ACRlongpl: made robust	148
\Acrlongpl: made robust	148
\acrlongpl: made robust	147
\ACRshort: made robust	143
\Acrshort: made robust	142
\acrshort: made robust	141
\ACRshortpl: made robust	144
\Acrshortpl: made robust	144
\acrshortpl: made robust	143
\Gls: made robust	124
\glsadd: made robust	158
\glsaddall: made robust	158
\GLSdesc: made robust	134
\Glsdesc: made robust	134
\glsdesc: made robust	133
\GLSdescplural: made robust	135
\Glsdescplural: made robust	134
\glsdescplural: made robust	134
\glsfirst: made robust	130
\GLSfirstplural: made robust	132
\Glsfirstplural: made robust	132
\glsfirstplural: made robust	132
\glslink: made robust	111
\GLSname: made robust	133
\Glsname: made robust	133

\glsname: made robust	133	\@printglossary: add a way to fetch	
\GLSpl: made robust	127	current entry label	192
\Glspl: made robust	127	savenumberlist: new	9
\glspl: made robust	126	ucmark: new	11
\GLSplural: made robust	131	\gls@defglossaryentry: added	
\GLSsymbol: made robust	136	numberlist element	86
\Glsymbol: made robust	135	\gls@save@numberlist: new	189
\glssymbol: made robust	135	\gls@wrglossary: added check for	
\GLSsymbolplural: made robust	136	glossary file defined	181
\Glsymbolplural: made robust	136	\glsdisplaynumberlist: new	156
\glssymbolplural: made robust	136	\glsentrycounter: set default value ..	113
\Glstext: made robust	130	\Glsentryfull: fixed bug (replaced	
\glstext: made robust	129	\glsentryshortpl with	
\GLSuseri: made robust	137	\glsentryshort)	156
\Glsuseri: made robust	137	\glsentryfullpl: fixed bug (replaced	
\glsuseri: made robust	137	\glsentryshort with	
\GLSuserii: made robust	138	\glsentryshortpl)	156
\Glsuserii: made robust	138	\glsentrynumberlist: new	156
\glsuserii: made robust	137	\glsmoveentry: new	88
\GLSuseriii: made robust	139	\glsresetsubentrycounter: new ...	206
\Glsuseriii: made robust	139	\ifglshaschildren: new	56
\glsuseriii: made robust	138	\ifglshasparent: new	56
\GLSuseriv: made robust	140	\makeglossaries: added list parser ..	173
\Glsuseriv: made robust	139	indexonlyfirst: new	27
\glsuseriv: made robust	139	\renewglossarystyle: new	214
\GLSuserv: made robust	140	\showglossaryentries: fixed misspelt	
\Glsuserv: made robust	140	command	259
\glsuserv: made robust	140	\SmallNewAcronymDef: fixed broken	
\GLSuservi: made robust	141	short and long plural	248
\Glsuservi: made robust	141	3.03 (2012/09/21)	
\glsuservi: made robust	141	\@gls@sanitizesort: new	22
3.02 (2012-05-19)		\@gls@setupsort@standard: used	
\glsnumlistlastsep: new	157	\@gls@sanitizesort	13
\glsnumlistsep: new	157	\@printglossary: allow title to override	
3.02 (2012-05-21)		default toctitle	191
\@do@wrglossary: changed		General: allow title to set toctitle	203
\@glslocref to		\glsinlinedescformat: new	272
\theglsentrycounter	186	\glsinlineemptydescformat: new ..	272
\@do@wrglossary: changed		\glsinlinenameformat: new	272
\@do@wr@glossary to test for		\glsinlinepostchild: new	272
indexonlyfirst option; put old		\glsinlinesubdescformat: new	272
\@do@wr@glossary code into		\glsinlinesubnameformat: new	272
\@do@wrglossary	181	\glspostinline: replaced “.” with	
\@gls@missingnumberlist: new	69	\glspostdescription	272
\@glswritefiles: added check for		list: added check for glsnogroupskip ..	274
existence of token in case		altlongragged4col: added check for	
\makeglossaries has been		glsnogroupskip	291
omitted	179	altsuperragged4col: added check for	
		glsnogroupskip	310

alttree: added check for glsnogroupskip	319	\gls@disablepagerefexpansion: new	181
index: added check for glsnogroupskip	313	\gls@numberpage: new	182
nogroupskip: new	11	\gls@protected@pagefmts: new	181
long: added check for glsnogroupskip .	277	\gls@romanpage: new	182
long3col: added check for glsnogroupskip	279	\glsdefmain: added check for doc package	16
long4col: added check for glsnogroupskip	280	\glsorg@endtheglossary: new	5
longragged: added check for glsnogroupskip	288	\glsorg@theglossary: new	5
longragged3col: added check for glsnogroupskip	289	\PrintChanges: new	5
nopostdot: new	11	3.05 (2013-04-21)	
tree: added check for glsnogroupskip .	314	\@do@esc@wrglossary: add Roman case. Fixed bugs in the else statements	185
treenoname: added check for glsnogroupskip	316	\@gls@link: added check for “nohypertypes”	112
super: added check for glsnogroupskip	299	mcocalttree: replaced ‘2’ with \glsmcols	297
super3col: added check for glsnogroupskip	301	mcocalindex: replaced ‘2’ with \glsmcols	293
super4col: added check for glsnogroupskip	303	mcocalindexspannav: replaced ‘2’ with \glsmcols	294
superragged: added check for glsnogroupskip	306	mcocaltree: replaced ‘2’ with \glsmcols	294
superragged3col: added check for glsnogroupskip	308	mcocaltreenoname: replaced ‘2’ with \glsmcols	296
3.04 (2012-11-11)		mcocaltreesspannav: replaced ‘2’ with \glsmcols	295
altlist: replaced \newline with paragraph break	274	\gls@protected@pagefmts: added Roman to list	181
3.04 (2012-11-18)		\gls@Romanpage: new	182
\@do@wrglossary: changed \theglsentrycounter back to \@glslocref	186	\glsgetgrouplabel: fixed bug (typo in \equal)	212
\@do@esc@wrglossary: modified to compensate for possible incorrect page number	185	\nopostdesc: made robust	37
\@gls@escbsdq: unsanitize \gls@numberpage, \gls@alphpage, \gls@Alphpage and \gls@romanpage	115	3.05 (2013/04/21)	
\@print@glossary: Moved aux write to end of document to prevent unwanted whatsit occurring here. . .	193	\@gls@nohyperlist: new	19
General: Added check for doc package . . .	4	\GlsDeclareNoHyperList: new	19
added datatool-base as a required package	4	nohypertypes: new	19
added local key	109	3.06 (2013/06/17)	
\gls@Alphpage: new	182	\@xdy@main@language: Changed back to using \language name	29
\gls@alphpage: new	182	\findrootlanguage: Obsoleted	51
		3.07 (2013-07-05)	
		\@gls@link: fixed bug that failed to find entry in list	112
		\glossarypreamble: modified to work with \setglossarypreamble	40
		\gls@docclearpage: added check for openright	43
		\glspostdescription: Added spacefactor code	10

\GlsSetXdyCodePage: Added check for fontspec	52	\glseelist: made robust	188
\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	242	\ifglshasdesc: new	57
\setglossarypreamble: new	40	\ifglshassymbol: new	57
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	291
list: updated list style to use \glossentry and \subglossentry	273	alttree: updated to use \glossentry and \subglossentry	318
listdotted: updated listdotted style to use \glossentry and \subglossentry	275	index: added paragraph break at end of environment	312
altlist: updated altlist style to use \glossentry and \subglossentry	274	updated to use \glossentry and \subglossentry	312
inline: updated inline style to use \glossentry and \subglossentry	271	long: updated to use \glossentry and \subglossentry	277
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	288
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	89	longragged3col: updated to use \glossentry and \subglossentry	289
updated for \glossentry	89	tree: updated to use \glossentry and \subglossentry	314
\@glossaryentryfield: switched to \glossentry	88	\setglossarystyle: new	213
\@glossarysubentryfield: switched to \subglossentry	88	\setglossentrycompatibility: new	210
General: added nogroupskip key to \printglossary	204	superragged: updated to use \glossentry and \subglossentry	306
removed definition of \@glossaryentryfield	361	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield	361	\@glsc@assign@symbolplural@field: new	22
\compatibleglossentry: new	208	\@glsc@default@value: new	65
\compatiblesubglossentry: new ...	210	\Glsentrydesc: made robust	151
\glossaryentryfield: deprecated ...	210	\Glsentrydescplural: made robust ..	151
\Glossentrydesc: new	209	\Glsentryfirst: made robust	152
\glossentrydesc: new	209	\Glsentryfirstplural: made robust .	153
\Glossentryname: new	209	\Glsentryfull: made robust	156
\glossentryname: new	209	\Glsentryfullpl: made robust	156
\Glossentrysymbol: new	210	\Glsentrylong: made robust	155
\glossentrysymbol: new	209	\Glsentrylongpl: made robust	155
\glsc@assign@desc@field: new	21	\Glsentryname: made robust	150
\glsc@assign@descplural@field: new	21	\Glsentryplural: made robust	152
\glsc@assign@field: new	71	\Glsentryshort: made robust	155
\glsc@ifnotmeasuring: new	90	\Glsentryshortpl: made robust	155
\glsc@addallunused: new	159	\Glsentrysymbol: made robust	152
\glsexpandfields: new	71	\Glsentrysymbolplural: made robust	152
\glscnoexpandfields: new	71	\Glsentrytext: made robust	151
\glscsee: made robust	188	\Glsentryuseri: made robust	153
\glscseeformat: made robust	188	\Glsentryuserii: made robust	154
\glscseeeitem: made robust	189	\Glsentryuseriii: made robust	154
		\Glsentryuseriv: made robust	154
		\Glsentryuserv: made robust	154
		\Glsentryuservi: made robust	155

<code>\glstextup: new</code>	218	<code>\@Gls@: add \glsifplural,</code>	
<code>\ifglshassymbol: changed test to check</code>		<code>\glscapscase, \glscustomtext and</code>	
<code>for \@gls@default@symbol</code>	57	<code>\glsinsert</code>	125
3.10a (2013-09-28)		change to using <code>\glentryfmt</code> style	
<code>\gls@assign@type@field: new</code>	21	commands	125
3.10a (2013-10-13)		removed <code>\makefirstuc</code> (now dealt	
<code>\@gls@keymap: new</code>	73	with in <code>\glentryfmt</code>)	125
<code>\@gls@provide@newglossary: new</code> ...	60	<code>\@Glspl@: add \glsifplural,</code>	
<code>\@gls@writedef: new</code>	73	<code>\glscapscase, \glscustomtext and</code>	
<code>\@glsdefaultplural: Obsolete</code>	69	<code>\glsinsert</code>	127
<code>\@glsnodesc: new</code>	69	change to using <code>\glentryfmt</code> style	
<code>\@print@glossary: Added</code>		commands	127
<code>providecommand</code> code to aux file ..	193	removed <code>\makefirstuc</code> (now dealt	
<code>\gls@defglossaryentry: Changed to</code>		with in <code>\glentryfmt</code>)	127
<code>using \@gls@default@value</code>	82	<code>\@acrlong: added \glslabel,</code>	
<code>new</code>	81	<code>\glsifplural, \glscapscase,</code>	
<code>\glswritedefhook: new</code>	81	<code>\glsinsert and \glscustomtext</code>	360
<code>\makeglossaries: Added</code>		<code>\@acrshort: added \glslabel,</code>	
<code>providecommand</code> code to aux file ..	172	<code>\glsifplural, \glscapscase,</code>	
<code>\new@glossaryentry: new</code>	72	<code>\glsinsert and \glscustomtext</code>	359
<code>\ns@newglossary: added</code>		<code>\@gls@: add \glslabel, \glsifplural,</code>	
<code>\@gls@provide@newglossary</code>	61	<code>\glscapscase, \glscustomtext and</code>	
3.11a (2013-10-15)		<code>\glsinsert</code>	124
<code>\@ACRlong: added \glslabel,</code>		change to using <code>\glentryfmt</code> style	
<code>\glsifplural, \glscapscase,</code>		commands	124
<code>\glsinsert and \glscustomtext</code>	361	<code>\@gls@noexpand@fields: Fixed bug</code>	
<code>\@ACRshort: added \glslabel,</code>		expand replaced with <code>noexpand</code>	70
<code>\glsifplural, \glscapscase,</code>		<code>\@glsdisp: add \glslabel,</code>	
<code>\glsinsert and \glscustomtext</code>	359	<code>\glsifplural, \glscapscase,</code>	
<code>\@Acrlong: added \glslabel,</code>		<code>\glscustomtext and \glsinsert</code>	128
<code>\glsifplural, \glscapscase,</code>		change to using <code>\glentryfmt</code> style	
<code>\glsinsert and \glscustomtext</code>	360	commands	129
<code>\@Acrshort: added \glslabel,</code>		<code>\@glspl@: add \glslabel,</code>	
<code>\glsifplural, \glscapscase,</code>		<code>\glsifplural, \glscapscase,</code>	
<code>\glsinsert and \glscustomtext</code>	359	<code>\glscustomtext and \glsinsert</code>	126
<code>\@GLS@: add \glslabel, \glsifplural,</code>		change to using <code>\glentryfmt</code> style	
<code>\glscapscase, \glscustomtext and</code>		commands	126
<code>\glsinsert</code>	125	General: added <code>\glslabel,</code>	
change to using <code>\glentryfmt</code> style		<code>\glsifplural, \glscapscase,</code>	
commands	125	<code>\glsinsert and</code>	
removed <code>\MakeUppercase</code> (now		<code>\glscustomtext</code>	142–148
moved to <code>\glentryfmt</code>)	125	changed to just use	
<code>\@GLSpl: add \glslabel,</code>		<code>\Glsentrydescplural</code>	135
<code>\glsifplural, \glscapscase,</code>		changed to just use	
<code>\glscustomtext and \glsinsert</code>	128	<code>\glentrydescplural</code>	134, 135
change to using <code>\glentryfmt</code> style		changed to just use <code>\Glsentrydesc</code> .	134
commands	128	changed to just use <code>\glentrydesc</code> .	134
removed <code>\MakeUppercase</code> as now		changed to just use	
dealt with in <code>\glentryfmt</code>	128	<code>\Glsentryfirstplural</code>	132

changed to just use		
\glentryfirstplural	132	
changed to just use \Glentryfirst	131	
changed to just use		
\glentryfirst	130, 131	
changed to just use \Glentryname	133	
changed to just use \glentryname	133	
changed to just use \Glentryplural	131	
changed to just use		
\glentryplural	131, 132	
changed to just use		
\Glentrysymbolplural	136	
changed to just use		
\glentrysymbolplural	136, 137	
changed to just use \Glentrysymbol	135	
changed to just use		
\glentrysymbol	135, 136	
Changed to just use \Glentrytext	130	
changed to just use \glentrytext	129	
changed to just use		
\Glentryuseriii	139	
changed to just use		
\glentryuseriii	138, 139	
changed to just use \Glentryuserii	138	
changed to just use \glentryuserii	138	
changed to just use \Glentryuseriv	139	
changed to just use		
\glentryuseriv	139, 140	
changed to just use \Glentryuseri	137	
changed to just use \glentryuseri	137	
changed to just use \Glentryuservi	141	
changed to just use \glentryuservi	141	
changed to just use \Glentryuserv	140	
changed to just use		
\glentryuserv	140, 141	
Now requires textcase	4	
acronymlists: replaced		
\@addtoacronymlists with		
\DeclareAcronymList	19	
\defgldisplay: obsoleted	108	
\defgldisplayfirst: obsoleted	108	
\defglentryfmt: new	60	
\forglentries: replaced \ifx with		
\ifdefempty	53	
\gls@assign@desc: new	81	
\gls@defglossaryentry: Fixed default		
counter if none supplied	85	
\gls@doentryfmt: new	60	
\gldisplay: obsoleted	107	
\gldisplayfirst: obsoleted	107	
\glsgenentryfmt: new	102	
\glsgetgrouptitle: Added check in		
case non-Latin alphabet in use	212	
\glsglossarymark: replaced		
\MakeUppercase with		
\mfirstucMakeUppercase	41	
\glsnavigation: switched to using		
\@gls@getgrouptitle	270	
\ifglshasdesc: replaced \ifdefempty		
with \ifcseempty	57	
\ifglshaslong: new	57	
\ifglshasshort: new	57	
\ifglshassymbol: replaced		
\ifdefempty with \ifcseempty	57	
\ifglused: replaced \ifthenelse with		
\ifbool	54	
\longnewglossaryentry: new	81	
\ns@newglossary: replaced		
\gldisplay and		
\gldisplayfirst with		
\glentryfmt	61	
compatible-3.07: cnew	30	
\SetCustomDisplayStyle: updated to		
use \defglentryfmt	252	
\SetDefaultAcronymDisplayStyle:		
changed to use \defglentryfmt	237	
\SetDescriptionAcronymDisplayStyle:		
updated to use \defglentryfmt	242	
\SetDescriptionDUAAcronymDisplayStyle:		
updated to use \defglentryfmt	241	
\SetDescriptionFootnoteAcronymDisplayStyle:		
updated to use \defglentryfmt	238	
\SetDUADisplayStyle: updated to use		
\defglentryfmt	250	
\SetFootnoteAcronymDisplayStyle:		
updated to use \defglentryfmt	245	
\SetSmallAcronymDisplayStyle:		
updated to use \defglentryfmt	247	
\setupglossaries: new	32	
\showglolong: new	257	
\showgloshort: new	257	
numbers: new	31	
symbols: new	31	
3.12a (2013-10-16)		
\gls@defglossaryentry: added		
\glslabel	82	
\glsaddkey: new	75	

3.13a (2013-11-05)		long: switched to \tabularnewline ..	277
\@gls@assign@symbol@field: changed		long3col: switched to	
to use \glsetnoexpandfield	22	\tabularnewline	278
\@gls@assign@symbolplural@field:		long3colheader: switched to	
changed to use		\tabularnewline	279
\glsetnoexpandfield	22	long3colheaderborder: switched to	
\@gls@link: removed \relax	113	\tabularnewline	279
\@gls@notranslatorhook: new	25	long4col: switched to	
\@gls@setupsort@standard: moved		\tabularnewline	280
\@gls@santizesort to		long4colheader: switched to	
\glsprestandardsort	13	\tabularnewline	280
ucmark: added check for memoir	11	longheader: switched to	
see: added \gls@checkseeallowed ...	66	\tabularnewline	278
\glossarysection: changed		longheaderborder: switched to	
\glossarymark to		\tabularnewline	278
\glsglossarymark	41	\SetFootnoteAcronymDisplayStyle:	
\glossarystyle: fixed bug caused by		fixed missing argument bug	245
using \ifdef instead of \ifcsdef .	214	super: switched to \tabularnewline .	299
\gls@assign@desc@field: changed to		super3col: switched to	
use \glsetnoexpandfield	21	\tabularnewline	301
\gls@assign@descplural@field:		super3colheader: switched to	
changed to use		\tabularnewline	301
\glsetnoexpandfield	21	super4col: switched to	
\gls@assign@name@field: changed to		\tabularnewline	302
use \glsetnoexpandfield	21	super4colheader: switched to	
\gls@assign@type@field: changed to		\tabularnewline	303
use \glsetexpandfield	21	super4colheaderborder: switched to	
\gls@checkseeallowed: new	66	\tabularnewline	303
\glsaddallunused: set default to		superheader: switched to	
\@glo@types	159	\tabularnewline	300
\Glsentryfull: changed to use		superheaderborder: switched to	
\acrfullformat	156	\tabularnewline	300
\Glsentryfull: changed to use			
\acrfullformat	156	3.14a (2013-11-12)	
\Glsentryfullpl: changed to use		\@glswritefiles: renamed	
\acrfullformat	156	\glswritefiles to	
\Glsentryfullpl: changed to use		\@glswritefiles and used	
\acrfullformat	156	“savewrites” option to set	
\glsglossarymark: renamed		\glswritefiles	178
\glossarymark to		General: new	261
\glsglossarymark to avoid conflict		acronyms: new	17
with memoir	41	\gls@defglossaryentry: added check	
\glsprestandardsort: new	13	for existence of default glossary	83
\glsetexpandfield: new	21	set the default for firstplural to be the	
\glsetnoexpandfield: new	21	value of plural	85
altsuper4colheader: switched to		xindygloss: new	29
\tabularnewline	304	\longprovideglossaryentry: new ...	81
altsuper4colheaderborder: switched		compatible-2.07: added check for 2.07	
to \tabularnewline	305	before setting 3.07 compatibility	30
		notranslate: new	26

\provideglossaryentry:new	72	index:new	31
4.0 (2013-11-14)		\newacronymstyle:new	224
\gls@defglossaryentry: added check		long-sc-short:new	227
for first key	85	long-sc-short-desc:new	228
super: fixed typo in \subglossentry		long-short:new	225
(\glossentrydesc)	299	long-short-desc:new	228
4.01 (2013-11-16)		long-sm-short:new	227
General: fixed non-value options so that		long-sm-short-desc:new	229
they can be passed to document class .	8	long-sp-short-desc:new	228
\CustomAcronymFields: inserted		footnote:new	232
missing comma	252	footnote-desc:new	234
4.02 (2013-12-05)		footnote-sc:new	233
\@acrfull: now using \acrfullfmt ..	219	footnote-sc-desc:new	234
\@gls@indexdef:new	31	footnote-sm:new	234
\@gls@numbersdef:new	31	footnote-sm-desc:new	235
\@gls@symbolsdef:new	31	\setacronymstyle:new	224
General: Removed \acronymfont .	145–149	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new	220	Moved check for empty custom text to	
\Acrfullfmt:new	220	prevent unwanted parenthetical	
\acrfullfmt:new	219	material	242
\ACRfullplfmt:new	221	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new	221	Moved check for empty custom text to	
\acrfullplfmt:new	221	prevent unwanted parenthetical	
\acronymentry:new	223	material	238
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here	25	Moved check for empty custom text to	
sc-short-long:new	227	prevent unwanted parenthetical	
sc-short-long-desc:new	229	material	245
\Genacrfullformat:new	107	\SetGenericNewAcronym:new	222
\genacrfullformat:new	107	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new	223	Moved check for empty custom text to	
\Genplacrfullformat:new	107	prevent unwanted parenthetical	
\genplacrfullformat:new	107	material	247
\Glsentryfull: bug fix: added missing		dua:new	230
\acronymfont	156	dua-desc:new	232
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont	156	option	8
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont	156	\makeglossaries: made preamble only	174
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont	156	General: changed default to \@empty	
\glsgenacfmt:new	105	instead of \relax	30
\GlsUseAcrEntryDisplayStyle:new ...	225	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new	225	\@do@esc@wrglossary: added	
short-long:new	226	\glsdetoklabel	186
short-long-desc:new	229	\@do@noesc@wrglossary: added	
xindynoglsnumbers:new	29	\glsdetoklabel	183
sm-short-long:new	228	\@ACRlong: removed \glslabel	
sm-short-long-desc:new	230	(defined in \@gls@link)	361

\@ACRshort: removed \glslabel (defined in \@gls@link)	359	\Genplacrfullformat: redefined to use accessibility information	358
\@Acrlong: removed \glslabel (defined in \@gls@link)	360	\genplacrfullformat: redefined to use accessibility information	358
\@Acrshort: removed \glslabel (defined in \@gls@link)	359	\glossentryname: added \glsdetoklabel	209
\@GLS@: removed \glslabel (defined in \@gls@link)	125	\gls@defglossaryentry: added \glsdetoklabel	81
\@GLSpl: removed \glslabel (defined in \@gls@link)	128	replaced #1 with \@gls@label	83
\@Gls@: removed \glslabel (defined in \@gls@link)	125	replaced \ifthenelse with \ifdefequal	84
\@Gls@entry@field: new	149	\glsadd: added \glsdetoklabel	158
\@Glspl@: removed \glslabel (defined in \@gls@link)	127	\glsaddkey: switched to using \@gls@field@link	76
\@acrlong: removed \glslabel (defined in \@gls@link)	360	\glsdetoklabel: new	54
\@acrshort: removed \glslabel (defined in \@gls@link)	359	\glsdisplaynumberlist: added \glsdetoklabel	157
\@gls@: removed \glslabel (defined in \@gls@link)	124	\glsdoifexistsorwarn: new	55
\@gls@access@display: new	347	\glsentryaccess: switched to using \@gls@entry@field	345
\@gls@entry@field: new	149	\glsentrydescaccess: switched to using \@gls@entry@field	346
\@gls@fetchfield: new	74	\glsentrydescpluralaccess: switched to using \@gls@entry@field	346
\@gls@field@link: new	129	\glsentryfirstaccess: switched to using \@gls@entry@field	346
\@gls@link: added \glsdetoklabel .	112	\glsentryfirstplural: added \glsdetoklabel	153
moved \@gls@link@opts and \@gls@link@label to \@gls@link	112	\glsentrylongaccess: switched to using \@gls@entry@field	347
\@gls@writedef: added \glsdetoklabel	73	\glsentrylongpluralaccess: switched to using \@gls@entry@field	347
\@glsdisp: removed \glslabel (defined in \@gls@link)	128	\glsentrypluralaccess: switched to using \@gls@entry@field	346
\@Glspl@: removed \glslabel (defined in \@gls@link)	126	\glsentryshortaccess: switched to using \@gls@entry@field	346
\@printglossary: added \glsdetoklabel	192	\glsentryshortpluralaccess: switched to using \@gls@entry@field	346
General: removed \glslabel (defined in \@gls@link)	142	\glsentrysymbolaccess: switched to using \@gls@entry@field	346
sc-short-long-desc: redefined to use accessibility information	365	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field	346
\compatibleglossentry: added \glsdetoklabel	341	\glsentrytextaccess: switched to using \@gls@entry@field	346
\compatiblesubglossentry: added \glsdetoklabel	342	\glsngenacfmt: redefined to use accessibility information	356
\Genacrfullformat: redefined to use accessibility information	358		
\genacrfullformat: redefined to use accessibility information	358		

\glsgenentryfmt: redefined to use accessibility information	353	sm-short-long-desc: redefined to use accessibility information	365
\glshyperlink: added \glsdetoklabel	157	long-sc-short-desc: redefined to use accessibility information	364
\glslocalreset: added \glsdetoklabel	90	long-short: redefined to use accessibility information	362
\glslocalunset: added \glsdetoklabel	91	long-short-desc: redefined to use accessibility information	364
\glsmoveentry: added \glsdetoklabel	88	long-sm-short-desc: redefined to use accessibility information	364
replaced \ifthenelse with \ifdefequal	88	footnote: redefined to use accessibility information	368
\glsrefentry: added \glsdetoklabel	207	footnote-desc: redefined to use accessibility information	370
\glsreset: added \glsdetoklabel ...	90	footnote-sc: redefined to use accessibility information	370
\glsseelist: added \expandafter commands	189	footnote-sc-desc: redefined to use accessibility information	371
\glsstepentry: added \glsdetoklabel	206	footnote-sm: redefined to use accessibility information	370
\glsstepsubentry: added \glsdetoklabel	207	footnote-sm-desc: redefined to use accessibility information	371
\glsunset: added \glsdetoklabel ...	91	\renewacronymstyle: new	224
short-long: commented spurious EOL	227	\showglocounter: added \glsdetoklabel	255
redefined to use accessibility information	363	\showglodesc: added \glsdetoklabel	256
short-long-desc: redefined to use accessibility information	365	\showglodescaccess: added \glsdetoklabel	377
\ifglshdescsuppressed: added \glsdetoklabel	57	\showglodescplural: added \glsdetoklabel	257
fixed typo	57	\showglodescpluralaccess: added \glsdetoklabel	377
\ifglshentryexists: added \glsdetoklabel	54	\showglofirst: added \glsdetoklabel	255
\ifglshaschildren: added \glsdetoklabel	56	\showglofirstaccess: added \glsdetoklabel	377
\ifglshasdesc: added \glsdetoklabel	57	\showglofirstpl: added \glsdetoklabel	255
\ifglshasfield: new	58	\showglofirstpluralaccess: added \glsdetoklabel	377
\ifglshaslong: added \glsdetoklabel	57	\showgloflag: added \glsdetoklabel	258
\ifglshasparent: added \glsdetoklabel	56	\showgloindex: added \glsdetoklabel	258
\ifglshasshort: added \glsdetoklabel	57	\showglolevel: added \glsdetoklabel	254
\ifglshassymbol: added \glsdetoklabel	57	\showglolong: added \glsdetoklabel	257
replaced \ifcempty with \ifdefempty and replaced \ifx with \ifdefequal	57	\showglolongaccess: added \glsdetoklabel	378
\ifglshused: added \glsdetoklabel ..	54		

\showglolongpluralaccess: added		redefined to use accessibility	
\glsdetoklabel	378	information	368
\showglongname: added \glsdetoklabel	256	4.04 (2014-03-04)	
\showglongnameaccess: added		\@gls@getcounterprefix: added	
\glsdetoklabel	377	warning if no prefix can be formed .	187
\showgloparent: added		4.04 (2014-03-06)	
\glsdetoklabel	254	\@gls@noidx@nosanitizesort: new .	22
\showgloplural: added		\@gls@noidx@sanitizesort: new ...	22
\glsdetoklabel	254	\@gls@nosanitizesort: new	22
\showglopluralaccess: added		\@gls@sanitizesort: new	22
\glsdetoklabel	377	\@glo@addchildren: new	194
\showgloshort: added		\@glo@do@sortentries: new	195
\glsdetoklabel	257	\@glo@grabfirst: new	200
\showgloshortaccess: added		\@glo@sortedinsert: new	196
\glsdetoklabel	378	\@glo@sortentries: new	194
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new	196
\glsdetoklabel	378	\@glo@sorthandler@letter: new ...	196
\showglosort: added \glsdetoklabel	257	\@glo@sorthandler@nocase: new ...	197
\showglosymbol: added		\@glo@sorthandler@word: new	196
\glsdetoklabel	257	\@glo@sortmacro@case: new	198
\showglosymbolaccess: added		\@glo@sortmacro@def: new	198
\glsdetoklabel	377	\@glo@sortmacro@def@do: new	199
\showglosymbolplural: added		\@glo@sortmacro@letter: new	197
\glsdetoklabel	257	\@glo@sortmacro@nocase: new	198
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ...	197
\glsdetoklabel	377	\@glo@sortmacro@use: new	199
\showglotext: added \glsdetoklabel	254	\@glo@sortmacro@word: new	197
\showglotextaccess: added		\@gls@noidx@do: new	201
\glsdetoklabel	377	\@gls@noidx@getgrouptitle: new ..	212
\showglotype: added \glsdetoklabel	255	\@gls@noref@warn: new	178
\showglouseri: added		\@gls@reference: new	203
\glsdetoklabel	255	\@gls@warnonglossdefined: new	20
\showglouserii: added		\@gls@warnontheGLOSSdefined: new .	20
\glsdetoklabel	255	\@no@makeglossaries: new	178
\showglouseriii: added		\@print@glossary: new	192
\glsdetoklabel	256	\@print@noidx@glossary: new	199
\showglouseriv: added		\@printgloss@setsort: new	190
\glsdetoklabel	256	\@printglossary: new	191
\showglouserv: added		General: added sort key to printgloss	
\glsdetoklabel	256	group	205
\showglouservi: added		\compatibleglossentry: changed	
\glsdetoklabel	256	\newcommand to \def as is may or	
dua: fixed bug in \acrfullfmt	231	may not be defined	341
fixed bug in \Acrfullplfmt	231	\compatiblesubglossentry: changed	
fixed bug in \acrfullplfmt	231	\newcommand to \def as is may or	
redefined to use accessibility		may not be defined	342
information	366	\defglsdisplayfirst: fixed unwanted	
dua-desc: commented spurious EOL ..	232	space	108
		\glo@grabfirst: new	200

\gls@defglossaryentry: replaced \ifx with \ifdefvoid	87	4.08 (2014-07-30)	
\glsnoidxdisplayloc: new	203	\@ACRlong: added	
\glsnoidxdisplaylocclishandler: new	202	\do@gls@link@checkfirsthyper	360
\glsnoidxloclist: new	202	\@ACRshort: added	
\glsnoidxlocclishandler: new	202	\do@gls@link@checkfirsthyper	359
\glsnoidxstripaccents: new	23	\@Acrlong: added	
alttree: moved hangindent and parindent assignments outside level test	318	\do@gls@link@checkfirsthyper	359
\makeglossaries: Moved definition of \glswrite to \makeglossaries ..	172	\@GLS@: moved \glsifhyper	125
\makenoidxglossaries: new	174	moved check for first use to \@gls@link	125
\printglossary: changed to use new \@printglossary	190	\@GLSpl: moved \glsifhyper	128
\printnoidxglossaries: new	190	moved check for first use to \@gls@link	128
\printnoidxglossary: new	190	\@Gls@: moved \glsifhyper	125
\showgloclolist: new	258	moved check for first use to \@gls@link	125
\warn@noprintglossary: Activate warning in \makeglossaries	190	\@Glspl@: moved \glsifhyper	127
\writeist: checked for definition of \glswrite	161, 165	moved check for first use to \@gls@link	127
4.06 (2014-03-12)		\@acrlong: added	
\@GLS@: added \glsifhyper	125	\do@gls@link@checkfirsthyper	360
\@GLSpl: added \glsifhyper	128	\@acrshort: added	
\@Gls@: added \glsifhyper	125	\do@gls@link@checkfirsthyper	358
\@Glspl@: added \glsifhyper	127	\@closegls: new	171
\@gls@: added \glsifhyper	124	\@gls@: moved \glsifhyper	124
\@gls@numbersdef: added hook to set toc title	31	moved check for first use to \@gls@link	124
\@gls@symbolsdef: added hook to set toc title	31	\@gls@automake: new	171
\@glsdisp: added \glsifhyper	129	\@gls@doautomake: new	30
\@glspl@: added \glsifhyper	126	\@gls@field@link: added assignment of \do@gls@link@checkfirsthyper	129
General: added \glsifhyper	142–149	\@gls@forbidtexext: new	60
acronym: added hook to set toc title	17	\@gls@hyp@opt: new	110
acronyms: added hook to set toc title ...	17	\@gls@link: removed redundancy	112
\glsdefmain: added hook to set toc title	16	renamed \gls@type to \glstype ...	112
4.07 (2014-04-04)		\@gls@link@checkfirsthyper: new .	111
\@glossarysection: added optional argument when using unstarred version	42	\@glsdisp: moved \glsifhyper	129
\@gls@noidx@do: added \global in case it's used in a tabular-like style	201	moved check for first use to \@gls@link	129
\Acrfullplfmt: fixed no case change bug	221	\@glspl@: moved \glsifhyper	126
\glsletentryfield: new	149	moved check for first use to \@gls@link	126
		\@ignored@glossaries: new	63
		General: added entrycounter option to printgloss family	205

added nopostdot option to printgloss family	204	removed \@sglstext	129
added subentrycounter option to printgloss family	205	removed \@sGLSuseriii	139
explicitly initialise hyper key	109	removed \@sGlsuseriii	139
moved \glsifhyper	142–149	removed \@sglsuseriii	138
removed \@sACRlongpl	148	removed \@sGLSuserii	138
removed \@sAcrlongpl	148	removed \@sGlsuserii	138
removed \@sacrlongpl	147	removed \@sGLSuseriv	140
removed \@sACRlong	146	removed \@sGlsuseriv	139
removed \@sAcrlong	146	removed \@sglsuseriv	139
removed \@sacrlong	145	removed \@sGLSuseri	137
removed \@sACRshortpl	144	removed \@sGlsuseri	137
removed \@sAcrshortpl	144	removed \@sglsuseri	137
removed \@sacrshortpl	143	removed \@sGLSuservi	141
removed \@sACRshort	143	removed \@sGlsuservi	141
removed \@sAcrshort	142	removed \@sglsuservi	141
removed \@sacrshort	142	removed \@sGLSuser	140
removed \@sgls@link	111	removed \@sGlsuser	140
removed \@sGLSdescplural	135	removed \@sglsuser	140
removed \@sGlsdescplural	135	removed \@sGLS	125
removed \@sglsdescplural	134	removed \@sGls	124
removed \@sGLSdesc	134	removed \@sgls	123
removed \@sGlsdesc	134	removed \@thirdofthree (defined in kernel)	123
removed \@sglsdesc	133	removed sPGLS	266
removed \@sglsdisp	128	removed sPgls	264
removed \@sGLSfirstplural	132	removed spgls	263
removed \@sGlsfirstplural	132	removed sPGLSpl	266
removed \@sglsfirstplural	132	removed sPglspl	265
removed \@sGLSfirst	131	removed spglspl	264
removed \@sGlsfirst	130	\ACRfull: removed \s@ACRfull	220
removed \@sglsfirst	130	switched to using \@gls@hyp@opt ..	220
removed \@sGLSname	133	\Acrfull: removed \s@Acrfull	220
removed \@sGlsname	133	switched to using \@gls@hyp@opt ..	219
removed \@sglsname	133	\acrfull: removed \@sacrfull	219
removed \@sGLSplural	132	switched to using \@gls@hyp@opt ..	219
removed \@sGlsplural	131	\ACRfullpl: removed \s@ACRfullpl ..	221
removed \@sglsplural	131	switched to using \@gls@hyp@opt ..	221
removed \@sGLSpl	127	\Acrfullpl: removed \s@Acrfullpl ..	221
removed \@sGlspl	127	switched to using \@gls@hyp@opt ..	221
removed \@sglspl	126	\acrfullpl: removed \s@acrfullpl ..	220
removed \@sGLSsymbolplural	136	switched to using \@gls@hyp@opt ..	220
removed \@sGlsymbolplural	136	\ACRlong: switched to using \@gls@hyp@opt	146
removed \@sglsymbolplural	136	\Acrlong: switched to using \@gls@hyp@opt	146
removed \@sGLSsymbol	135	\acrlong: switched to using \@gls@hyp@opt	145
removed \@sGlsymbol	135		
removed \@sglsymbol	135		
removed \@sGLStext	130		
removed \@sGlstext	130		

\ACRlongpl: switched to using \@gls@hyp@opt	148	\glsenablehyper: added \KV@glslink@hypertrue to definition	123
\Acrlongpl: switched to using \@gls@hyp@opt	148	\GLSfirst: switched to using \@gls@hyp@opt	131
\acrlongpl: switched to using \@gls@hyp@opt	147	\Glsfirst: switched to using \@gls@hyp@opt	130
\ACRshort: switched to using \@gls@hyp@opt	143	\glsfirst: switched to using \@gls@hyp@opt	130
\Acrshort: switched to using \@gls@hyp@opt	142	\GLSfirstplural: switched to using \@gls@hyp@opt	132
\acrshort: switched to using \@gls@hyp@opt	141	\Glsfirstplural: switched to using \@gls@hyp@opt	132
\ACRshorttpl: switched to using \@gls@hyp@opt	144	\glsfirstplural: switched to using \@gls@hyp@opt	132
\Acrshorttpl: switched to using \@gls@hyp@opt	144	\glsifhyper: deprecated	110
\acrshorttpl: switched to using \@gls@hyp@opt	143	\glslink: switched to using \@gls@hyp@opt	111
\forallacronyms: new	53	\glslinkcheckfirsthyperhook: new	112
\GLS: switched to using \@gls@hyp@opt	125	\glslinkvar: new	110
\Gls: switched to using \@gls@hyp@opt	124	\GLSname: switched to using \@gls@hyp@opt	133
\gls: switched to using \@gls@hyp@opt	123	\Glsname: switched to using \@gls@hyp@opt	133
\gls@defglossaryentry: added check for ignored glossary	83	\glsname: switched to using \@gls@hyp@opt	133
\gls@istfilebase: new	38	\GLSpl: switched to using \@gls@hyp@opt	127
\glsaddkey: removed \@sGLS@user@<key>	77	\Glspl: switched to using \@gls@hyp@opt	127
removed \@sGls@user@<key>	77	\glspl: switched to using \@gls@hyp@opt	126
removed \@sgls@user@<key>	76	\GLSplural: switched to using \@gls@hyp@opt	131
switched to using \@gls@hyp@opt	76, 77	\Glsplural: switched to using \@gls@hyp@opt	131
\GLSdesc: switched to using \@gls@hyp@opt	134	\glsplural: switched to using \@gls@hyp@opt	131
\Glsdesc: switched to using \@gls@hyp@opt	134	\glsspace: new	219
\glsdesc: switched to using \@gls@hyp@opt	133	\GLSsymbol: switched to using \@gls@hyp@opt	136
\GLSdescplural: switched to using \@gls@hyp@opt	135	\Glsymbol: switched to using \@gls@hyp@opt	135
\Glsdescplural: switched to using \@gls@hyp@opt	134	\glsymbol: switched to using \@gls@hyp@opt	135
\glsdescplural: switched to using \@gls@hyp@opt	134	\GLSsymbolplural: switched to using \@gls@hyp@opt	136
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	123	\Glsymbolplural: switched to using \@gls@hyp@opt	136
\glsdisp: switched to using \@gls@hyp@opt	128		
\glsdohyperlink: new	122		
\glsdohypertarget: new	122		

\glssymbolplural: switched to using \@gls@hyp@opt	136	\newglossary: added starred version ..	61
\GLStext: switched to using \@gls@hyp@opt	130	\newignoredglossary: new	63
\Glstext: switched to using \@gls@hyp@opt	130	\ns@newglossary: added \@glotype@<name>@log	61
\glstext: switched to using \@gls@hyp@opt	129	new	61
\glstreenamefmt: new	311	\p@gls@hyp@opt: new	110
\GLSuseri: switched to using \@gls@hyp@opt	137	\PGLS: changed to use \@gls@hyp@opt	266
\Glsuseri: switched to using \@gls@hyp@opt	137	\Pgls: changed to use \@gls@hyp@opt	264
\glsuseri: switched to using \@gls@hyp@opt	137	\pgls: changed to use \@gls@hyp@opt	263
\GLSuserii: switched to using \@gls@hyp@opt	138	\PGLSpl: changed to use \@gls@hyp@opt	266
\Glsuserii: switched to using \@gls@hyp@opt	138	\Pglspl: changed to use \@gls@hyp@opt	265
\glsuserii: switched to using \@gls@hyp@opt	137	\pglspl: changed to use \@gls@hyp@opt	264
\GLSuseriii: switched to using \@gls@hyp@opt	139	\s@gls@hyp@opt: new	110
\Glsuseriii: switched to using \@gls@hyp@opt	139	\s@newglossary: new	61
\glsuseriii: switched to using \@gls@hyp@opt	138	automake: new	30
\GLSuseriv: switched to using \@gls@hyp@opt	140	4.09 (2014-08-12)	
\Glsuseriv: switched to using \@gls@hyp@opt	139	\glsaddkey: fixed bug in user commands	76
\glsuseriv: switched to using \@gls@hyp@opt	139	4.10 (2014-08-27)	
\GLSuseriv: switched to using \@gls@hyp@opt	140	\@Gls@acentryname: new	150
\Glsuseriv: switched to using \@gls@hyp@opt	140	\@Gls@entryname: new	150
\GLSuserv: switched to using \@gls@hyp@opt	141	\@gls@glossary: Renamed \@glossary to \@gls@glossary	180
\Glsuserv: switched to using \@gls@hyp@opt	141	\glspercentchar: new	160
\glsuserv: switched to using \@gls@hyp@opt	141	\glstildechar: new	160
\GLSuservi: switched to using \@gls@hyp@opt	141	alttree: moved space after symbol	318, 319
\Glsuservi: switched to using \@gls@hyp@opt	141	4.11 (2014-09-01)	
\glsuservi: switched to using \@gls@hyp@opt	141	\@do@esc@wrglossary: added hook ..	185
\ifignoredglossary: new	63	sanitize: none option	25
altlongragged4col: fixed bug that displayed description instead of symbol	291	\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	180
		\glsaddprotectedpagefmt: new	182
		\glsbackslash: new	160
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute	47
		\@gls@adjustmode: new	158
		\@gls@notranslatorhook: removed ...	25
		\@gls@toc: added \protect to \numberline	44
		\@gls@usetranslator: new	25
		\glsacrpluralsuffix: new	35
		\glsadd: added check for vertical mode	158
		\glsaddallunused: replaced @gobble with glsignore	159
		\glsifusedtranslatordict: new	26
		\glsignore: new	159

\glsupacrpluralsuffix:new	35	4.16 (2015-06-18)	
\ProvidesGlossariesLang:new	35	\glsaddstoragekey:new	74
\RequireGlossariesLang:new	35	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	361
\indexspace:new	273, 293, 311	\@ACRshort: added \glspostlinkhook	360
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	360
\@glslocalreset:new	92	\@Acrshort: added \glspostlinkhook	359
\@glslocalunset:new	91	\@GLS@: added \glspostlinkhook ...	126
\@glsreset:new	92	\@GLSpl: added \glspostlinkhook ..	128
\@glsunset:new	91	\@Gls@: added \glspostlinkhook ...	125
\@newglossaryentry@defcounters:		\@Glspl@: added \glspostlinkhook ..	127
new	93	\@acrlong: added \glspostlinkhook	360
\cGls:new	96	\@acrshort: added \glspostlinkhook	359
\cGls@:new	96	\@gls@: added \glspostlinkhook ...	124
\cGlspl@:new	97	\@gls@@link: added	
\cgls:new	96	\glspostlinkhook	111
\cgls@:new	96	\@gls@field@link: added	
\cglspl@:new	97	\glspostlinkhook	129
\cglspl@:new	97	\@gls@link: moved definition of	
\gls@entry@count:new	96	\glsifhyperon outside of this	
\gls@increment@currcount:new ...	95	macro	113
\gls@local@increment@currcount:		\@glsdisp: added \glspostlinkhook	129
new	95	\@glspl@: added \glspostlinkhook ..	126
\gls@write@entrycounts:new	95	General: added \glspostlinkhook	142–149
\glslocalreset:new	92	\glsacspace:new	226
\glslocalunset:new	91	\glsadd: changed \@do@wrglossary to	
\glsreset:new	92	\@do@wrglossary	158
\glsunset:new	91	\glsfielddef:new	79
\@newglossaryentry@defcounters:		\glsfieldedef:new	78
new	87	\glsfieldfetch:new	79
\cGls:new	96	\glsfieldgdef:new	78
\cgls:new	96	\glsfieldxdef:new	78
\cGlsformat:new	97	\glsifhyperon: moved definition of	
\cglsformat:new	96	\glsifhyperon	112
\cGlspl:new	97	\glslinkpostsetkeys:new	112
\cglspl:new	97	\glspostlinkhook:new	111
\cGlsplformat:new	98	\glswriteentry:new	181
\cglsplformat:new	97	\ifglsfieldcseq:new	80
\gls@defdocnewglossaryentry:new ..	71	\ifglsfielddefeq:new	80
\glsenableentrycount:new	93	\ifglsfielddeq:new	79
\glslocalreset: switched to		long-sp-short:new	225
\@glslocalreset	90	\showglofield:new	258
\glslocalunset: switched to		4.18 (2015-09-09)	
\@glslocalunset	91	General: split mfirstuc into separate	
\glsreset: switched to \@glsreset ...	90	bundle	4
\glsunset: switched to \@glsunset ...	91	4.19 (2015-10-31)	
4.15 (2015-03-16)		\glstreenamebox:new	317
General: bug fix replaced \@glo@type		4.19 (2015-11-22)	
with \glstype	148	\@gls@link@nocheckfirsthyper:new	129

\@gls@preglossaryhook: new	191	\glsfindwidesttoplevelname: new ..	317
\@printglossary: added		\glslistgroupheaderfmt: new	273
\@gls@preglossaryhook	192	\glslistnavigationitem: new	273
\do@glsglisablehyperinlist: new ..	112	\glstreegroupheaderfmt: new	311
\doifglossarynoexistsordo: new ...	56	\glstreenavigationfmt: new	311
\gls@gobbleopt: new	60	\ifglswrallowprimitivemods: new ..	183
\glsglsoifexistsordo: new	55	list: fixed missing space before	
4.20 (2015-11-30)		description	273
\@gls@link: added		long: fixed typo in \glossentrydesc ..	277
\@gls@setdefault@glslink@opts ..	112	super4col: fixed bug in \glossentry ..	302
added \glsglsonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed	113	\glsglcurrentfieldvalue: new	59
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new	112	\glsglcurrentfieldvalue	58, 59
\glsgl@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	66	nogroupskip changed	291
\glsglsonohyperlink: new	122	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	310
\@printglossary: warn if no style has		long: check for nogroupskip changed ..	277
been set	191	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	283
assignment	142–148	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed	279
already set	214	long3col-booktabs: check for	
\glsglLTpenaltycheck: new	286	nogroupskip changed	284
\glsglpatchLToutput: new	286	long4col: check for nogroupskip	
\glsglspenaltygroupskip: new	286	changed	280
altlong4col-booktabs: new	284	long4col-booktabs: check for	
altlongragged4col-booktabs: new ..	285	nogroupskip changed	284
long-booktabs: new	283	longragged: check for nogroupskip	
long3col-booktabs: new	283	changed	288
long4col-booktabs: new	284	longragged3col: check for nogroupskip	
longragged-booktabs: new	285	changed	289
longragged3col-booktabs: new	285	super: check for nogroupskip changed ..	299
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set	213	changed	301
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@esc@wrglossary: added check		changed	303
for \@arabic	185	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	306
modifications and added arabic case	185	superragged3col: check for	
mcolalttreespannav: new	298	nogroupskip changed	308
mcolindexspannav: new	294	4.24 (2016-05-27)	
mcoltreenonamespannav: new	296	\@gls@extramakeindexopts: new ...	170
mcoltreespannav: new	295	\@gls@glossary: added check for debug	
\glsgl@arabicpage: new	182	mode	180
\glsgl@protected@pagefmts: added		\@gls@see@noindex: new	6
arabic to list	181	debug: new	5
\glsglentrytitlecase: new	153	seenoinindex: new	6

\glsnomakeindexwarning: new	44	added \TH, \dh and \DH	23
\GlsSetQuote: new	167	4.31 (2017-08-10)	
\GlsSetWriteIstHook: new	167	nolist: added check for “list” style	10
4.25 (2016-06-09)		4.31 (2017-09-10)	
\@gls@enablesavenonumberlist: new	67	style: changed \renewcommand to \def	8
\@gls@initnonumberlist: new	67	4.32 (2017-08-24)	
\@gls@savenonumberlist: new	67	\@glsnavhypertarget: new	268
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	293	\gls@the and \gls@number	185
mcolindexspannav: replaced \@idxitem		renamed from	
with \glstreeitem	294	\@do@esc@wrglossary	183
\glstreechildpredesc: new	312	\@do@noesc@wrglossary: new	183
\glstreeitem: new	311	\@do@wrglossary: changed to check	
\glstreepredesc: new	312	for esclocations	183
\glstreesubitem: new	312	\@gls@missinglang@warn: new	20
\glstreesubsubitem: new	312	\GlsSetXdyFirstLetterAfterDigits:	
4.28 (2017-01-07)		added starred version	160
\glspatchtabularx: new	90	\GlsSetXdyNumberGroupOrder: new	160
4.29 (2017-01-19)		esclocations: new	9
\@gls@noidx@do: current letter group		4.34 (2017-11-03)	
assignment made global	202	mcolalttreespannav: removed spurious	
\@print@noidx@glossary: moved		space	298
definition of		\glsshowtarget: modified to check for	
\@gls@currentlettergroup outside		math mode and inner	6
of theglossary environment	199	4.35 (2017-11-14)	
General: added check for		\glsadd: added \@gls@setsort (in case	
\@glsxtr@doaccsupp	341	of sort=use)	158
\glsnavhyperlinkname: new	268	4.36 (2018-03-07)	
4.30 (2017-06-11)		\@gls@glossary: removed \index	180
\@glo@autosee: new	87	4.37 (2018-04-07)	
\@glo@autoseehook: new	87	\gls@begindocdefs: new	72
\@glo@check@sortallowed: new	13	4.38 (2018-05-10)	
\@gls@noidx@do: letter group		\@gls@define@glossaryentrycounter:	
assignment made global	201	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	49	new	12
\@xdylocationclassorder: bug fix:		prepended \currentglossary. to	
changed \edef to \def	50	\theHglossarysubentry and	
\glosortentrieswarning: new	20	removed spurious eol space	12
\gls@set@xr@key: new	66	\glsaccsupp: added braces around	
\gls@xr@key: new	66	actual text argument	347
\GlsAddXdyLocation: bug fix: changed		\glsentrycounterlabel: bug fix: move	
#1 to #2	50	conditional inside command	207
\glsnoidxstripaccents: added \a	23	\GlsEntryCounterLabelPrefix: new	204

\glsentryitem: bug fix: move conditional inside command	207	\glsstepsubentry: bug fix: move conditional inside command	207
\glsrefentry: bug fix: move conditional inside command	207	\glssubentrycounterlabel: bug fix: move conditional inside command .	207
\glsresetsubentrycounter: bug fix: move conditional inside command .	206	\glssubentryitem: bug fix: move conditional inside command	208
\glsstepentry: bug fix: move conditional inside command	206	\showglonameaccess: bug fix: corrected field (was showing text access field)	377

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	
\!	118, 119
\"	23, 116–119, 121
\#	163, 164
\%	160, 166, 325, 326
\&	35, 157
\'	23
\.	10, 23
\=	23
\?	116–118, 168
\@	72
\@@delimN	216
\@@do@@wrglossary	175, 183, 186
\@@do@esc@wrglossary	183
\@@do@noesc@wrglossary	183
\@@do@wrglossary	158, 181
\@@glo@assign@sortkey	176
\@@glo@list	53
\@@glo@sort	23
\@@glo@type	190
\@@glossarysec	7, 42, 43
\@@glossaryseclabel	8, 42, 43, 204
\@@glossarysecstar	8, 42, 43, 204
\@@gls@checkactual	120
\@@gls@checkbar	119
\@@gls@checkescactual	118
\@@gls@checkescbar	118
\@@gls@checkesclevel	119
\@@gls@checkescquote	117, 169, 170
\@@gls@checklevel	120
\@@gls@checkquote	116, 117, 167, 168
\@@gls@default@entryfmt	99, 108
\@@gls@expand@field	21, 70, 71, 75, 76, 237, 239–241, 244, 246, 249–251, 372–376
\@@gls@extramakeindexopts	167, 172, 173
\@@gls@fixbraces	188
\@@gls@noexpand@field	21, 69, 70
\@@gls@noidx@no@sanitizesort	22, 23
\@@gls@noidx@nosanitizesort	177
\@@gls@nosanitizesort	22, 177
\@@gls@sanitizesort	22, 177
\@@gls@xdycheckbackslash	121, 122
\@@gls@xdycheckquote	121
\@@gls@localreset	92, 94
\@@gls@localunset	91, 94
\@@glsreset	92, 94
\@@glsunset	91, 94
\@@newglossaryentry@defcounters	93
\@@this@glo@	54
\@ACRfull	220
\@ACRfullpl	221
\@ACRlong	146, 220
\@ACRlongpl	148, 221
\@ACRshort	143, 220
\@ACRshortpl	144, 145, 221
\@Acrfull	220
\@Acrfullpl	221
\@Acrlong	146, 220
\@Acrlongpl	148, 221
\@Acrshort	142
\@Acrshortpl	144
\@Alph	182, 185
\@GLS	125
\@GLS@	125, 266
\@GLSdesc	134
\@GLSdesc@	134
\@GLSdescplural	135
\@GLSdescplural@	135
\@GLSfirst	131
\@GLSfirst@	131
\@GLSfirstplural	132
\@GLSfirstplural@	132
\@GLSname	133
\@GLSname@	133

\@GLSpl	127	\@Glsuseri	137
\@GLSpl@	127, 128, 267	\@Glsuseri@	137
\@GLSplural	131, 132	\@Glsuserii	138
\@GLSplural@	132	\@Glsuserii@	138
\@GLSsymbol	136	\@Glsuseriii	139
\@GLSsymbol@	136	\@Glsuseriii@	139
\@GLSsymbolplural	136	\@Glsuseriv	139
\@GLSsymbolplural@	136, 137	\@Glsuseriv@	139
\@GLStext	130	\@Glsuserv	140
\@GLStext@	130	\@Glsuserv@	140
\@GLSuseri	137	\@Glsuservi	141
\@GLSuseri@	137	\@Glsuservi@	141
\@GLSuserii	138	\@Mi	286
\@GLSuserii@	138	\@PGLS	266
\@GLSuseriii	139	\@PGLS@	266
\@GLSuseriii@	139	\@PGLSpl	266
\@GLSuseriv	140	\@PGLSpl@	266
\@GLSuseriv@	140	\@Pgls	264
\@GLSuserv	140	\@Pgls@	264
\@GLSuserv@	140, 141	\@Pglspl	265
\@GLSuservi	141	\@Pglspl@	265
\@GLSuservi@	141	\@Roman	182, 185
\@Gls	124	\@acrfull	219
\@Gls@	94, 96, 124, 265	\@acrfullpl	220
\@Gls@acrentryname	222	\@acrlong	145, 219
\@Gls@entry@field	76, 150–155	\@acrlongpl	147, 221
\@Gls@entryname	150, 222	\@acrshort	142, 219, 220
\@GlsSetXdyFirstLetterAfterDigits	160	\@acrshortpl	143, 221
\@GlsSetXdyNumberGroupOrder	160, 161	\@addtoacronymlists	17, 18
\@Glsdesc	134	\@after	18
\@Glsdesc@	134	\@afterheading	274, 329
\@Glsdescplural	134, 135	\@alph	182, 185
\@Glsdescplural@	135	\@arabic	182, 185
\@Glsfirst	130	\@auxout	60,
\@Glsfirst@	130, 131		61, 95, 172, 173, 175, 178, 189, 193, 194, 269
\@Glsfirstplural	132	\@backslashchar	115, 121, 122
\@Glsfirstplural@	132	\@before	18
\@Glsname	133	\@bsphack	180
\@Glsname@	133	\@cGls	96
\@Glspl	127	\@cGls@	94, 96
\@Glspl@	95, 97, 127, 265, 266	\@cGlspl	97
\@Glsplural	131	\@cGlspl@	95, 97
\@Glsplural@	131	\@cclv	286, 287
\@Glsymbol	135	\@cgls	96
\@Glsymbol@	135	\@cgls@	94, 96
\@Glsymbolplural	136	\@cglspl	97
\@Glsymbolplural@	136	\@cglspl@	94, 97
\@Glstext	130	\@chapter	33
\@Glstext@	130	\@classoptionslist	32

<code>\@closegls</code>	171, 172	<code>\@glo@autoseehook</code>	87
<code>\@colht</code>	286	<code>\@glo@check@mkidxrangechar</code>	114, 115, 186, 322, 323
<code>\@colroom</code>	286, 287	<code>\@glo@check@sortallowed</code> ..	13–15, 174, 177
<code>\@currentlabelname</code>	8, 204	<code>\@glo@childlist</code>	195
<code>\@curroptions</code>	32	<code>\@glo@counter</code>	66, 82, 85
<code>\@declaredoptions</code>	32	<code>\@glo@counterprefix</code>	178, 183, 185–187, 213, 216
<code>\@delimN</code>	215	<code>\@glo@default@sorttype</code> ..	12, 176, 197, 198
<code>\@delimR</code>	215	<code>\@glo@defaultcounter</code>	85
<code>\@disable@onlypremakeg</code>	173	<code>\@glo@desc</code>	64, 81–83, 86
<code>\@disable@premakecs</code>	34	<code>\@glo@descaccess</code>	343–345
<code>\@disabled@gl saddxdycounters</code>	46	<code>\@glo@descplural</code>	64, 81, 82
<code>\@do@addcounter</code>	45	<code>\@glo@descpluralaccess</code>	343–345
<code>\@do@auxoutstuff</code>	193, 194	<code>\@glo@do@sortentries</code>	194
<code>\@do@glossentry</code>	208, 209, 341, 342	<code>\@glo@entry</code>	159
<code>\@do@gls@getcounterprefix</code>	183, 186	<code>\@glo@entryprefix</code>	261
<code>\@do@gls@islistofacronyms</code>	18	<code>\@glo@entryprefixfirst</code>	261
<code>\@do@glssee</code>	87	<code>\@glo@entryprefixfirstplural</code> ..	261, 262
<code>\@do@ifinlist</code>	44, 45	<code>\@glo@entryprefixplural</code>	261
<code>\@do@newglossaryentry</code>	223, 237, 239–241, 243–246, 248–253, 372–376	<code>\@glo@esclabel</code>	88, 89
<code>\@do@seeglossary</code>	175, 188	<code>\@glo@etext</code>	99–101
<code>\@do@subglossentry</code>	210, 342	<code>\@glo@first</code>	65, 82, 85, 86, 248, 376
<code>\@do@wrglossary</code>	113	<code>\@glo@firstaccess</code>	342, 344, 345
<code>\@do@writeaux@info</code>	189	<code>\@glo@firstplural</code>	65, 82, 85, 376
<code>\@ehc</code>	286	<code>\@glo@firstpluralaccess</code>	343–345
<code>\@empty</code>	11, 15, 17, 30, 32, 33, 44, 46, 49, 52, 53, 83, 84, 89, 113, 114, 124–128, 142–148, 161, 164, 166, 167, 171, 172, 179, 180, 187, 213, 238, 240, 242, 244–247, 249, 251, 253, 323, 325, 327, 359–361	<code>\@glo@grabfirst</code>	200
<code>\@end@fixbraces</code>	188	<code>\@glo@label</code>	68, 69, 75, 76, 78–80, 82–87, 93, 157, 261, 262, 317, 345
<code>\@endfortrue</code>	27, 56, 74, 269	<code>\@glo@list</code>	87
<code>\@esphack</code>	181	<code>\@glo@long</code>	57, 68, 83, 85
<code>\@expandtwoargs</code>	32	<code>\@glo@longaccess</code>	343–345
<code>\@firstofone</code>	23	<code>\@glo@longpl</code>	68, 83, 85, 237, 239, 241, 243, 246, 248, 250, 372
<code>\@firstofthree</code>	110, 123, 124, 126, 128, 142, 144, 145, 147, 359–361	<code>\@glo@longpluralaccess</code>	343–345
<code>\@firstoftwo</code>	26, 27, 73, 74, 110, 126–128, 144, 145, 147, 148	<code>\@glo@name</code>	13, 64, 69, 82, 84–86
<code>\@for</code>	27, 32, 34, 45, 46, 53, 73, 74, 115, 161–163, 173, 174, 181, 188, 194, 195, 224, 238, 240, 242, 244, 246, 249, 251, 253, 269, 270, 323	<code>\@glo@no@assign@sortkey</code>	174
<code>\@glo@@desc</code>	86	<code>\@glo@nonumberlist</code>	67
<code>\@glo@@symbol</code>	86	<code>\@glo@numfmt</code>	186, 323
<code>\@glo@access</code>	342, 344, 345, 347	<code>\@glo@parent</code>	15, 66, 82, 84, 85, 89, 195
<code>\@glo@addchildren</code>	194, 199	<code>\@glo@plural</code>	64, 82, 84, 85, 374
<code>\@glo@assign@sortkey</code>	174, 176, 205	<code>\@glo@pluralaccess</code>	343–345
<code>\@glo@autosee</code>	87	<code>\@glo@prefix</code>	9, 66, 67, 82, 89, 114, 115, 186, 322, 323
		<code>\@glo@range</code>	186, 322, 323
		<code>\@glo@see</code>	66, 82, 87
		<code>\@glo@seeautonumberlist</code>	9, 66
		<code>\@glo@short</code>	57, 58, 68, 83, 85, 375
		<code>\@glo@shortaccess</code>	343–345, 372–375

<code>\@glo@shortpl</code>	68, 83, 85,	<code>\@gls@link</code>	111
	237, 239, 241, 243, 245, 248, 250, 372, 375	<code>\@gls@Hcounter</code>	113, 114
<code>\@glo@shortpluralaccess</code>	343–345	<code>\@gls@ReturnAfterFi</code>	216
<code>\@glo@sort</code>	13, 15, 22, 23, 64, 82, 85, 89	<code>\@gls@access@display</code>	347–349
<code>\@glo@sortedinsert</code>	195, 196	<code>\@gls@actualchar</code> .	89, 118, 120, 165, 166, 326
<code>\@glo@sortentries</code>	197, 198	<code>\@gls@addpredefinedattributes</code> .	161, 170
<code>\@glo@sorthandler@case</code>	198	<code>\@gls@adjustmode</code>	158
<code>\@glo@sorthandler@letter</code>	197	<code>\@gls@automake</code>	174
<code>\@glo@sorthandler@nocase</code>	198	<code>\@gls@between</code>	270
<code>\@glo@sorthandler@word</code>	197	<code>\@gls@body</code>	150
<code>\@glo@sortinghandler</code>	194, 196	<code>\@gls@checkactual</code>	116, 168
<code>\@glo@sortinglist</code>	194, 196, 198, 199	<code>\@gls@checkbar</code>	116, 168
<code>\@glo@sorttype</code>	176, 199, 200, 205	<code>\@gls@checkedmkidx</code>	115–121, 167–169
<code>\@glo@storeentry</code>	13–15	<code>\@gls@checkescactual</code>	116, 168
<code>\@glo@suffix</code>	114, 115, 186, 323	<code>\@gls@checkescbar</code>	116, 169
<code>\@glo@symbol</code>	57, 65, 82, 86, 243, 248, 344	<code>\@gls@checkescquote</code>	116, 168, 169
<code>\@glo@symbolaccess</code>	343–345, 375	<code>\@gls@checklevel</code>	116, 169
<code>\@glo@symbolplural</code>	65, 82, 86	<code>\@gls@checkmkidxchars</code>	
<code>\@glo@symbolpluralaccess</code>	343–345	..	88, 89, 114, 168, 175, 185–187, 322, 323
<code>\@glo@text</code>	64,	<code>\@gls@checkquote</code>	116, 167, 168
	82, 85, 86, 124–129, 149–151, 243, 262, 374	<code>\@gls@classI</code>	162
<code>\@glo@textaccess</code> ...	342, 344, 345, 372–375	<code>\@gls@classII</code>	162
<code>\@glo@thislabel</code>	88	<code>\@gls@codepage</code>	193, 194
<code>\@glo@thislettergrp</code>	200–202	<code>\@gls@counter</code>	
<code>\@glo@thisvalue</code>	58, 59	109, 112–114, 158, 178, 186, 187, 323
<code>\@glo@tmp</code>	75, 76, 187	<code>\@gls@counterwithin</code>	11, 12
<code>\@glo@type</code>	8,	<code>\@gls@ctr</code>	45
	15, 65, 82, 83, 85–87, 158, 159, 173, 179,	<code>\@gls@currentlettergroup</code>	200–202
	191–195, 199, 200, 203, 204, 222, 238,	<code>\@gls@debugfalse</code>	5
	240, 242, 244, 246, 249, 251, 253, 268, 270	<code>\@gls@debugtrue</code>	5
<code>\@glo@types</code>	53,	<code>\@gls@declareoption</code>	
	54, 61, 92, 93, 158, 159, 173, 174, 259, 317	9, 10, 16, 17, 20, 21, 26, 29, 31
<code>\@glo@useri</code>	67, 82, 85	<code>\@gls@default</code>	98
<code>\@glo@userii</code>	67, 82, 85	<code>\@gls@default@value</code>	
<code>\@glo@useriii</code>	68, 82, 85	57–59, 69, 70, 82, 84–86, 247, 261
<code>\@glo@useriv</code>	68, 82, 85	<code>\@gls@deffile</code>	72, 73
<code>\@glo@userv</code>	68, 82, 85	<code>\@gls@define@glossaryentrycounter</code> .	
<code>\@glo@uservi</code>	68, 82, 85	12, 33, 205, 206
<code>\@glodesc</code>	86	<code>\@gls@define@glossarysubentrycounter</code>	
<code>\@glolist@</code>	83	33, 205, 206
<code>\@gloname</code>	86	<code>\@gls@defsort</code>	13–15, 86
<code>\@glossary@default@style</code>		<code>\@gls@defsortcount</code>	13–15, 62
.....	8, 10, 191, 213, 214, 254	<code>\@gls@do@acronymsdef</code>	16, 17, 32, 33, 63
<code>\@glossaryentryfield</code>	89	<code>\@gls@do@indexdef</code>	31–33, 63
<code>\@glossarysection</code>	41	<code>\@gls@do@numbersdef</code>	31–33, 63
<code>\@glossarystyle</code>	191, 192, 204	<code>\@gls@do@symbolsdef</code>	31, 63
<code>\@glossarysubentryfield</code>	89	<code>\@gls@do@symbolssdef</code>	32, 33
<code>\@gls</code>	123	<code>\@gls@doautomake</code>	30, 174
<code>\@gls@</code>	94, 96, 123, 264, 265	<code>\@gls@docheckquotedef</code>	167–170

\@gls@docloadedfalse	4	\@gls@link ..	111, 124–129, 142–149, 359–361
\@gls@docloadedtrue	4	\@gls@link@checkfirsthyper	124–128
\@gls@dodedeflistparser	174	\@gls@link@label	112, 239, 245
\@gls@doentrycounterdef	32, 33	\@gls@link@nocheckfirsthyper	129, 142–148
\@gls@doentrydef	108	\@gls@link@opts	112, 239, 245
\@gls@dolast	188, 189	\@gls@list	269, 270
\@gls@donext	188, 189	\@gls@listsuffix	44
\@gls@donext@def	157	\@gls@loadlist	10, 253
\@gls@dosubentrycounterdef	33	\@gls@loadlong	9, 10, 254
\@gls@dotothiswrite	171, 172	\@gls@loadsuper	10, 254
\@gls@elem	269	\@gls@loadtree	10, 254
\@gls@enablesavenonumberlist	72	\@gls@local@increment@currcount	94
\@gls@encapchar		\@gls@loclist	176, 177, 201, 202
..... 118, 119, 165, 166, 186, 188, 323, 326		\@gls@map	73, 74
\@gls@entry@count	95	\@gls@missinglang@warn	20, 36
\@gls@entry@field		\@gls@missingnumberlist	86
..... 75, 76, 93, 94, 150–156, 345–347		\@gls@noaccess	347
\@gls@escbsdq	116, 166, 327	\@gls@noexpand@fields	71
\@gls@expand@fields	70, 71	\@gls@nohyperlist	19, 63, 112
\@gls@expandonce	71	\@gls@noidx@do	200
\@gls@extramakeindexopts	173	\@gls@noidx@getgrouptitle	175
\@gls@fetchfield	58	\@gls@noidx@sanitizesort	22, 177
\@gls@field@link	76, 77, 129–141	\@gls@noidx@setsanitizesort	24, 177
\@gls@firsttok	200	\@gls@noidx@loclist@finalsep	176
\@gls@fixbraces	87	\@gls@noidx@loclist@prev	176, 202, 203
\@gls@forbidtexext	61	\@gls@noidx@loclist@sep	176, 202, 203
\@gls@get@counterprefix	187	\@gls@noref@warn	175, 200
\@gls@getbody	150	\@gls@numberlink	216
\@gls@getcounterprefix	183, 186	\@gls@numbersdef	31
\@gls@getgrouptitle	175, 212, 270	\@gls@numlist@lastsep	157
\@gls@glossary	180	\@gls@numlist@nextsep	157
\@gls@gobbleopt	60	\@gls@numlist@sep	157
\@gls@grptitle	212, 268, 270	\@gls@old@chapter	33
\@gls@hyp@opt		\@gls@oldnewglossaryentryposthook ..	344
77, 96, 97, 111, 123–148, 219–221, 263–266		\@gls@oldnewglossaryentryprehook ..	344
\@gls@hyp@opt@cs	110	\@gls@onlypremakeg	33, 34
\@gls@hypergroup	269	\@gls@order	171, 172
\@gls@ifinlist	45	\@gls@org@LT@output	286
\@gls@ifnotmeasuring	90	\@gls@org@glsnoidxdisplayloc	177
\@gls@igtype	63	\@gls@org@glsseeformat	177
\@gls@increment@currcount	94	\@gls@patchtabularx	90
\@gls@indexdef	31	\@gls@preglossaryhook	192
\@gls@initnonumberlist	67, 82	\@gls@prevlevel ..	297, 298, 317–320, 335, 336
\@gls@islistofacronyms	18	\@gls@provide@newglossary	61
\@gls@keylist	371	\@gls@quotechar	117–120, 165–169, 326
\@gls@keymap	67, 73–76, 261, 344	\@gls@reference	175, 178
\@gls@label	175, 178, 183, 186	\@gls@removespaces	216
\@gls@langmod	171	\@gls@renewglossary	171
\@gls@levelchar ..	89, 119, 120, 165, 166, 326	\@gls@replacementtext	347

\@gls@rest	150	\@glsacronymlists ..	17–19, 53, 222, 224, 238, 240, 242, 244, 246, 249, 251, 253, 258
\@gls@restoreat	72, 73	\@glsaddkey	75, 76
\@gls@roman	48, 323, 324	\@glsaddstoragekey	74, 75
\@gls@sanitized@tmp	115	\@glsaddxdyattribute	45, 46
\@gls@sanitizedesc	28	\@glsdefaultsort	13
\@gls@sanitizesort	13	\@glsdesc	133
\@gls@sanitizesymbol	28	\@glsdesc@	133, 134
\@gls@saveentrycounter	113, 158	\@glsdescplural	134
\@gls@savenonumberlist	66, 67	\@glsdescplural@	134
\@gls@see@noindex	7, 66	\@glsdisp	128
\@gls@setacrstyle	28, 32, 33	\@glsentry	92, 93, 95
\@gls@setcounter	62	\@glsentrytitlecase	153
\@gls@setdefault@glslink@opts	112	\@glsfirst	130
\@gls@setsort	13–15, 113, 158	\@glsfirst@	130
\@gls@setupshortcuts	32, 33	\@glsfirstletter	160
\@gls@sort	201	\@glsfirstplural	132
\@gls@sort@A	196, 197	\@glsfirstplural@	132
\@gls@sort@B	196, 197	\@glshypernumber	215
\@gls@startswithexpandonce	70	\@glsisacronymlistfalse	18
\@gls@storenonumberlist	67, 86	\@glsisacronymlisttrue	18
\@gls@symbolsdef	31	\@glslink	113, 123, 157, 268
\@gls@this	181	\@glslocalreset	91, 94
\@gls@thisHloc	187	\@glslocalunset	91, 94
\@gls@thisfield	58, 59	\@glslocref	178, 183, 185, 186, 322, 323
\@gls@thislabel	56, 188, 189, 198	\@glsminrange	161, 162, 324
\@gls@thislist	157	\@glsname	133
\@gls@thisloc	187	\@glsname@	133
\@gls@thisval	74	\@glsnavhypertarget	268
\@gls@title	41	\@glsnextpages	192
\@gls@tmp ...	15, 36, 49, 71, 115, 181, 269, 270	\@glsnodesc	82, 83, 86
\@gls@tmpb	116–121, 167–169	\@glsnoname	82, 84, 86
\@gls@toc	42, 43	\@glsnonextpages	192
\@gls@type	174, 224, 238, 240, 242, 244, 246, 249, 251, 253, 317	\@glsnumberformat	109, 112, 158, 178, 186, 322, 323
\@gls@updatechecked	115, 116, 168, 169	\@glsopenfile	171, 179
\@gls@usetranslator	26, 27, 36	\@glsorder	172, 173
\@gls@value	70, 153	\@glspl	126
\@gls@warnonglossdefined	21, 190	\@glspl@	94, 97, 126, 264–266
\@gls@warnonthe glossdefined	21, 208	\@glsplural	131
\@gls@write@entrycounts	95	\@glsplural@	131
\@gls@writedef	72	\@glsreset	90, 94
\@gls@writeisthook	165, 167	\@glssee	87, 188
\@gls@xdy@locationlist	162	\@glsshowtarget	5, 6, 122
\@gls@xdycheckbackslash	115	\@glsymbol	135
\@gls@xdycheckquote	115	\@glsymbol@	135
\@gls@xref	187, 188	\@glsymbolplural	136
\@gls@Alphacompositor	39, 49, 324	\@glsymbolplural@	136
\@gls@Hlocref	183, 186	\@gls@target	123, 208, 269

\@glstext	129	\@onlypreamble ...	62, 72, 81, 95, 98, 174, 177
\@glstext@	129	\@onlypremakeg	38, 39, 45, 46, 50, 62, 167
\@glunset	91, 94	\@org@glossaryentrynumbers	191, 192
\@gluseri	137	\@org@gl@assign@descplural	
\@gluseri@	137	237, 246, 248–251, 372, 375, 376
\@gluserii	137, 138	\@org@gl@assign@firstpl	237,
\@gluserii@	138	239–241, 243, 244, 246, 248–251, 372–376	
\@gluseriii	138	\@org@gl@assign@plural	237,
\@gluseriii@	138	239–241, 243, 244, 246, 248–251, 372–376	
\@gluseriv	139	\@org@gl@assign@symbolplural ..	237,
\@gluseriv@	139	239–241, 243, 244, 249–251, 373, 374, 376	
\@gluserv	140	\@org@gl@numberformat	157
\@gluserv@	140	\@org@newglossaryentryprehook	81
\@gluservi	141	\@outputpage	286, 287
\@gluservi@	141	\@p@glossarysection	41
\@glswidestname	317–319, 335	\@pgls	263
\@glswritefiles	30	\@pgls@	263
\@glxtr@doaccsupp	341	\@pglspl	264
\@gobble	5, 13–15, 73, 90,	\@pglspl@	264
115, 159, 160, 163, 164, 175, 321, 325, 326		\@plus	273, 293, 311
\@idxitem	311	\@print@glossary	190
\@ifclassloaded	4, 11, 41	\@print@noidx@glossary	190
\@ifnextchar	62, 110	\@printgloss@setsort	174, 176, 191
\@ifpackageloaded		\@printglossary	190
.....	4, 8, 25–27, 36, 52, 90, 156, 167, 341	\@roman	48, 323
\@ifstar	61, 74, 75, 110, 160, 218	\@secondofthree	
\@ifundefined	35,	110, 123, 125, 127, 142, 144, 146, 148, 359	
269, 276, 287, 298, 305, 318, 319, 335, 349		\@secondoftwo .	23, 26, 27, 36, 73, 74, 122–
\@ignored@glossaries	63	125, 128, 142, 143, 145–147, 359–361, 379	
\@input@	193	\@set@glo@numformat	186, 323
\@istfilename	172, 173	\@sglsaddkey	75
\@makecol	286, 287	\@sglsaddstoragekey	74, 75
\@makeglossary	173	\@thirdofthree	
\@minus	273, 293, 311	110, 125, 128, 143, 145, 147, 148, 359
\@mkboth	41, 42	\@this@attr	163, 164
\@newglossary	60, 61	\@this@childlabel	195
\@newglossaryentry@defcounters ..	87, 93	\@this@counter	46
\@newglossaryentryposthook		\@this@ctr	163
.....	75, 76, 87, 261, 344	\@this@key	74
\@newglossaryentryprehook		\@this@label	194
.....	75, 76, 81, 83, 261, 344	\@thiscs	34
\@nil	18, 87, 114–116, 150, 168,	\@tmp	48, 324
169, 186, 188, 200, 201, 215, 216, 322, 323		\@use@ption	32
\@nnil	18, 188	\@warn@nomakeglossaries	172, 194
\@no@makeglossaries	174, 175	\@wrglossary@pageformat	182
\@no@post@desc	328	\@wrglossarynumberhook	182, 185
\@nopostdesc	192	\@xdy@main@language	29, 171, 193
\@onelevel@sanitize	22, 48, 73, 89,	\@xdy@attributelist	46, 163
115, 160, 161, 164, 187, 189, 200, 324, 325		\@xdy@attributes	46, 162, 321, 323

\@xdycounters	45, 46, 163	\acronymentry	223, 225–230, 232–235, 363–365, 368–371
\@xdycrossrefhook	163	\acronymfont	105, 142–145, 150, 156, 222, 223, 225–235, 239, 240, 242, 244, 245, 247–249, 356, 357, 359–361, 363–365, 367–371, 373–375
\@xdylanguage	193	\acronymname	16, 17, 37
\@xdylettergroups	53, 165, 326	\acronymsort	223, 225–230, 232–235, 363–365, 368, 369, 371
\@xdylocationclassorder	50, 163, 325	\acronymtype	16, 17, 222–224, 237–246, 248–251, 253, 372–375
\@xdylocref	46, 165, 321, 325	\acrpluralsuffix	223, 225–228, 232–234, 237–241, 243–250, 252, 253, 363, 364, 368–370, 372–376
\@xdynumbergrouporder	52, 160, 161	\Acrshort	235
\@xdyrequiredstyles	51, 161, 323	\acrshort	235
\@xdysortrules	51, 165, 326	\Acrshorttpl	235
\@xdystyle	161, 162, 323	\acrshorttpl	235
\@xdyuseralphabets	47, 162, 323	\addcontentsline	44
\@xdyuserlocationdefs	49, 50, 163, 322, 324	\addglossarytocaptions	36
\@xdyuserlocationnames	50, 322	\addtolength	319, 335
\@xfor@nextelement	188	\advance	14, 15, 84, 113, 286
\\	88, 115, 160, 166, 215, 216, 326, 327, 329–331, 339, 340	\AE	23
\{	73, 159, 166, 321, 326, 327	\ae	23
\}	73, 159, 166, 321, 327	amsgen package	4, 109
\~	23	amsmath package	90
\‘	23	\andname	189
\	116, 118, 169	\AnyTrackedLanguages	36, 379
\~	23	\appto	19, 67, 75, 76, 261, 344
A		array package	283, 287, 305
\a	23	article class	187
\AA	23	\AtBeginDocument	17, 52, 72, 90, 158, 175
\aa	23	\AtEndDocument	30, 72, 95, 174, 178, 193, 269
accsupp package	341	B	
\accsuppglossaryentryfield	341	\b	23
\accsuppglossarysubentryfield	342	babel package	25, 34, 36, 51
\acrfootnote	239, 245	\begin	164, 200, 273, 277–282, 285–311, 325
\Acrfull	236	\BeginAccSupp	347
\acrfull	236	\begingroup	5, 180, 184, 216
\ACRfullfmt	220, 223, 231, 233, 367, 369	\bfseries	278–281, 283, 284, 288–290, 292, 300–305, 307–311
\Acrfullfmt	220, 223, 231, 233, 367, 369	\bgroup	23, 81, 156, 191, 195
\acrfullfmt	219, 223, 231, 233, 367, 369	bib2gls	184
\acrfullformat	156, 219, 237, 252	booktabs package	282–285
\Acrfullpl	236	\boolean	252
\acrfullpl	236	\boolfalse	30
\ACRfullplfmt	221, 223, 231, 233, 367, 369	\booltrue	30
\Acrfullplfmt	221, 223, 231, 233, 367, 369	\bottomrule	283, 284
\acrfullplfmt	221, 223, 231, 233, 367, 369		
\acrlinkfootnote	238		
\acrlinkfullformat	219–221		
\Acrlong	236		
\acrlong	235		
\Acrlongpl	236		
\acrlongpl	235		
\acrnameformat	243, 374		

<code>\box</code>	286	<code>\CurrentTrackedTag</code>	36, 379, 380
C		<code>\CustomAcronymFields</code>	253
<code>\c</code>	23	<code>\CustomNewAcronymDef</code>	253
<code>\c@equation</code>	113	D	
<code>\c@glossaryentry</code>	11	<code>\d</code>	23
<code>\c@glossarysubentry</code>	12	datatool package	196
<code>\c@page</code>	182, 185	<code>\day</code>	161, 165, 323, 326
<code>\catcode</code>	72	<code>\DeclareAcronymList</code>	16, 17, 19, 222, 224, 238, 240, 242, 244, 246, 249, 251, 253
<code>\cGls</code>	96	<code>\DeclareListParser</code>	174
<code>\cgl</code> s	96	<code>\DeclareOption</code>	8, 261, 341
<code>\cGlsformat</code>	94	<code>\DeclareOptionX</code>	8
<code>\cgl</code> sformat	94	<code>\DeclareRobustCommand</code>	37, 188, 189, 247, 347–349
<code>\cGlspl</code>	97	<code>\def</code> ..	8, 9, 11–15, 18, 22, 23, 28, 29, 32–34, 37, 38, 41, 44, 47–52, 56, 60–62, 64–68, 71, 79, 81–86, 88, 90, 94–98, 108, 109, 112–122, 124–149, 157, 158, 161, 165, 167–169, 171–173, 175, 176, 178, 179, 182, 183, 185–188, 191, 194, 198, 200, 202–205, 212–217, 219–222, 237–244, 246, 248–251, 253, 261, 263–267, 270– 272, 297, 298, 317–320, 322, 323, 326, 328, 335, 336, 341–344, 358–361, 372–376
<code>\cgl</code> spl	97	<code>\def@gl</code> s@xdycheckbackslash	121, 122
<code>\cGlsplformat</code>	95	<code>\DefaultNewAcronymDef</code>	238
<code>\cgl</code> splformat	94	<code>\defgl</code> sentryfmt	61, 63, 108, 224, 237, 238, 241, 242, 245, 247, 250, 252
<code>\char</code>	212	<code>\define@boolkey</code>	7, 9, 11, 12, 16, 24, 27–30, 109, 205
classicthesis package	8	<code>\define@choicekey</code>	5–8, 12, 25, 26, 28, 66, 204, 205
<code>\cleardoublepage</code>	43	<code>\define@key</code>	8, 12, 19, 25, 29, 64– 68, 75, 76, 109, 158, 203–205, 261, 342, 343
<code>\clearpage</code>	43	<code>\DefineAcronymSynonyms</code>	32, 236
<code>\closeout</code>	72, 165, 167, 171, 179	<code>\delimN</code>	164, 174, 202, 216, 325
<code>\compatglossarystyle</code>	328–340	<code>\delimR</code>	164, 215, 216, 325
<code>\compatibleglossentry</code>	210	<code>\DescriptionDUANewAcronymDef</code>	242
<code>\compatiblesubglossentry</code>	210	<code>\DescriptionFootnoteNewAcronymDef</code> .	240
<code>\copy</code>	286, 287	<code>\descriptionname</code>	37, 278–281, 283, 284, 288–290, 292, 300–305, 307–311
<code>\count@</code>	200, 201	<code>\DescriptionNewAcronymDef</code>	244
<code>\csdef</code>	21, 75–77, 87, 88, 93, 94, 194, 195, 214, 224, 225, 327	<code>\DH</code>	23
<code>\csedef</code>	95, 182	<code>\dh</code>	23
<code>\csgdef</code>	40, 60, 63, 94, 95, 189, 203	<code>\dimen@</code>	226, 286, 317
<code>\cslet</code>	67, 81, 88, 199	<code>\disable@keys</code>	32
<code>\csname</code> ..	12–15, 32, 34, 36, 37, 42, 43, 46, 48, 49, 52, 53, 56, 61, 62, 69, 70, 75–80, 83– 89, 91, 92, 108, 112–114, 124–129, 142– 149, 157, 158, 162–164, 168–171, 175, 178, 179, 181, 182, 186–188, 191, 193, 195, 204, 209, 210, 213, 214, 218, 254– 262, 269, 270, 317–319, 321–323, 335, 341, 342, 345, 346, 349, 359–361, 377, 378	<code>\do</code>	27, 32, 34, 45, 46, 53, 73, 74, 115, 157, 161–163, 173,
<code>\csshow</code>	258		
<code>\csuse</code>	37, 40, 60, 70, 76, 77, 108, 171, 172, 195, 198, 199, 201, 203–205, 214, 225, 262, 328–340		
<code>\csxdef</code>	86, 95		
<code>\currentglossary</code>	11, 12, 40, 192		
<code>\currentglssubentry</code>	12, 207		
<code>\CurrentOption</code>	32, 261, 341		
<code>\CurrentTrackedLanguage</code>	36, 379, 380		

174, 181, 188, 194, 195, 224, 238, 240, 242, 244, 246, 249, 251, 253, 269, 270, 323	\endcsname 12–
\do@glo@storeentry 13–15, 87	15, 32, 34, 36, 37, 42, 43, 46, 48, 49, 52, 53, 56, 61, 62, 69, 70, 75–80, 83– 89, 91, 92, 108, 112–114, 124–129, 142– 149, 157, 158, 162–164, 168–171, 175, 178, 179, 181, 182, 186–188, 191, 193, 195, 204, 209, 210, 213, 214, 218, 254– 262, 269, 270, 317–319, 321–323, 335, 341, 342, 345, 346, 349, 359–361, 377, 378
\do@gl@link@checkfirsthyper	\endfoot . 277–279, 281, 283, 284, 288–290, 292
. . . . 111, 112, 124–129, 142–148, 359, 360	\endgroup 5, 181, 185, 216
\do@gl@xdycheckbackslash 115	\endhead . 277–279, 281, 283, 284, 288–290, 292
\do@gl@disablehyperinlist 112	\endtheglossary 5
\do@gl@haschildren 56	\entryname 37, 278–281, 283, 284, 288–290, 292, 300–305, 307–311
doc package 4, 5, 16	\equal 25, 33, 43, 113, 173, 213, 269
\doifglossarynoexistsordo 61	equation (counter) 113, 114
\dtl@ifsingle 212	etoolbox package 4
\dtl@insertinto 196	\expandafter . . . 13–15, 22, 32–34, 36, 46, 48, 49, 51–54, 56, 61–63, 69, 70, 73–80, 83–87, 89–92, 108, 112–121, 150, 157, 159, 160, 163, 164, 167–171, 179, 181, 183, 185, 186, 189, 191, 195, 200, 201, 209, 210, 214, 216, 218, 239, 245, 254– 259, 261, 262, 269, 270, 317, 321–323, 325, 326, 341, 342, 345, 347, 371, 377, 378
\dtl@sortresult 196, 197	\expandonce 70, 71, 115, 168, 169, 182, 196, 197, 209, 210, 223, 237, 239, 241, 243, 245, 246, 248, 250, 341, 342
\dtlcompare 196	
\dtlicompare 197	
\DTLifinlist 63, 112	
\DTLifint 212	
\dtlletterindexcompare 196	
\DTLsubstituteall 115	
\dtlwordindexcompare 196	
\DUANewAcronymDef 251	
E	
\eappto 63, 88, 182	
\edef 15, 18, 34, 36, 44–51, 53, 56, 61, 63, 70, 72, 74, 78–80, 82, 83, 88, 89, 108, 112–121, 157–160, 165, 167–169, 171, 172, 174, 175, 179, 183, 186, 187, 189, 193–197, 201, 207, 212, 216, 218, 237, 239, 241, 243, 245, 248, 250, 268, 321, 322, 324, 326, 371–375	
\egroup 23, 81, 157, 192, 195	
\else 6, 10–12, 15–18, 20, 22, 24, 25, 30, 32, 33, 38, 39, 41–48, 50–53, 66, 69, 84, 85, 88–90, 94, 95, 111–113, 115–122, 124–129, 150, 160, 161, 164– 170, 172, 179, 181–183, 185–189, 191, 200, 205, 207, 213, 215, 216, 226, 240, 242, 244, 247, 249–252, 254, 269, 274, 277, 279, 280, 283, 284, 286, 288, 290, 291, 299, 301, 303, 306, 308, 310, 313, 314, 316, 318, 319, 322–328, 333–336, 347	
\emph 188, 217	
\empty 216, 341	
\end 164, 200, 273, 277–282, 285–311, 325	
\end@doifinlist 44, 45	
\end@getprefix 187	
\end@gl@islistofacronyms 18	
\EndAccSupp 347	
	F
	\fi 5–8, 10–18, 20, 22, 24, 25, 27, 30, 32–34, 37–39, 41–53, 62, 67, 69, 83–86, 88–90, 94, 95, 111–122, 124–129, 151, 158, 160–162, 164–172, 174, 179–183, 185–189, 191, 194, 201, 204–208, 213– 216, 226, 236, 238, 240–242, 244, 246, 247, 249–254, 260, 269, 274, 277, 279, 280, 283, 284, 286–288, 290, 291, 299, 301, 303, 306, 308, 310, 313, 314, 316– 319, 321–325, 327, 328, 333–336, 341, 347
	file types
	.aux 193
	.glo 88
	.ist 159, 170
	.toc 43
	.xdy 38
	glo 259
	\firstacronymfont 107, 225–227, 232, 233, 239, 243, 245, 248, 358, 362–364, 368, 369

`\footnote` 232, 233, 238, 369
`\FootnoteNewAcronymDef` 246
`\forallallglossaries` 54, 179, 190, 317
`\forallallglsentries` 92, 93, 95, 159
`\ForEachTrackedDialect` 36, 379, 380
`\forglsentries` 54, 56, 88, 198, 317
`\forlistcsloop` 194, 200
`\forlistloop` 176, 177, 202

G

`garamondx` package 218
`\gdef` 14, 46, 61, 79, 84, 85, 180, 206, 269
`\Genacrfullformat`
 106, 223, 225–227, 232, 358, 362, 363, 369
`\genacrfullformat` 106, 107,
 223, 225, 226, 232, 357, 358, 362, 363, 368
`\GenericAcronymFields` .. 223, 225, 226,
 228–232, 234, 235, 362–365, 367, 368, 371
`\Genplacrfullformat`
 106, 223, 225–227, 233, 357, 363, 369
`\genplacrfullformat`
 106, 107, 223, 225–227, 233, 357, 363, 369
`\glo@desc` 328
`\glo@do@compare` 196, 197
`\glo@grabfirst` 201
`\glo@label` 56, 88
`\glo@list` 88
`\glo@name` 209
`\glo@parent` 56
`\glo@type` 88
`\glo@value` 73
`\global` 14, 15, 69, 72, 81,
 86, 91, 92, 181, 192, 201, 202, 206, 286, 287
`\glolinkprefix` 113, 157, 208
`\glosortentrieswarning` 20, 194
`glossaries` package
 ... 31, 51, 52, 161, 253, 261, 273, 321, 341
`glossaries-accsupp` package 88, 341
`glossaries-extra` package 72, 163, 341
`\GlossariesWarning`
 5, 7, 20, 24, 25, 40, 44, 55, 59, 66,
 69, 96, 97, 108, 110, 156, 172, 175–178,
 181, 187, 191, 192, 210, 211, 213, 321, 341
`\GlossariesWarningNoLine`
 5, 6, 20, 173, 175, 179, 194, 269
`\glossary` 322, 323
`glossary` package 1, 217
`glossary` styles:
 `altlist` 274, 275, 329

`altlistgroup` 275, 329
`altlisthypergroup` 275, 329
`altlong4col` 281, 282, 290, 331
`altlong4col-booktabs` 284, 286
`altlong4colborder` 282, 331
`altlong4colheader` 282, 284, 331
`altlong4colheaderborder` 282, 331
`altlongragged4col` ... 285, 290–292, 332
`altlongragged4col-booktabs` 285
`altlongragged4colborder` 292, 332
`altlongragged4colheader` 291, 332
`altlongragged4colheaderborder` 292, 333
`altsuper4col` 304, 309, 340
`altsuper4colborder` 304, 340
`altsuper4colheader` 304, 340
`altsuper4colheaderborder` ... 304, 340
`altsuperragged4col` 309, 310, 338
`altsuperragged4colborder` ... 310, 338
`altsuperragged4colheader` ... 310, 338
`altsuperragged4colheaderborder` .
 310, 338
`alttree` 297, 312, 317, 319, 335
`alttreegroup` 320, 336
`alttreehypergroup` 320, 336
`index` 8, 293, 311–314, 333
`indexgroup` 313, 333
`indexhypergroup` 313, 333
`inline` 328
`list` 8, 10, 273–275, 328
`listdotted` 275, 276, 329
`listgroup` 274, 328
`listhypergroup` 274, 329
`long` 277, 278, 283, 287, 329, 331
`long-booktabs` 283, 285
`long3col` 278, 279, 283, 330
`long3col-booktabs` 283, 285
`long3colborder` 279, 330
`long3colheader` 279, 283, 330
`long3colheaderborder` 279, 330
`long4col` 280, 281, 284, 330
`long4col-booktabs` 284
`long4colborder` 281, 331
`long4colheader` 280, 284, 331
`long4colheaderborder` 281, 331
`longborder` 277, 330
`longheader` 277, 283, 330
`longheaderborder` 278, 330
`longragged` 285, 287–289
`longragged-booktabs` 285

longragged3col 285, 289, 290, 332
longragged3col-booktabs 285
longragged3colborder 290, 332
longragged3colheader 290, 332
longragged3colheaderborder . 290, 332
longraggedborder 288, 331
longraggedheader 288, 332
longraggedheaderborder 289, 332
mcolalmtree 297, 337
mcolalmtreegroup 297, 337
mcolalmtreehypergroup ... 297, 298, 337
mcolindex 293, 336
mcolindexgroup 293, 336
mcolindexhypergroup 293, 294, 336
mcoltree 294, 336
mcoltreegroup 336
mcoltreehypergroup 295, 336
mcoltreenoname 296, 337
mcoltreenonamegroup 296, 337
mcoltreenonamehypergroup ... 296, 337
sublistdotted 329
super 299, 300, 307, 339
super3col 300–302, 339
super3colborder 301, 339
super3colheader 301, 339
super3colheaderborder 302, 339
super4col 302–304, 340
super4colborder 303, 340
super4colheader 303, 340
super4colheaderborder 303, 340
superborder 299, 339
superheader 300, 339
superheaderborder 300, 339
superragged 305, 307, 337
superragged3col 307–309, 338
superragged3colborder 308, 338
superragged3colheader 308, 338
superragged3colheaderborder 309, 338
superraggedborder 306, 337
superraggedheader 307, 337
superraggedheaderborder 307, 338
tree 294, 314, 315, 317, 333
treegroup 295, 315, 334
treehypergroup 315, 334
treenoname 295, 312, 315, 316, 334
treenonamegroup 316, 335
treenonamehypergroup 316, 335
glossary-hypernav package 159
glossary-list package 8, 10, 273
glossary-long package 9, 276, 290, 298, 299
glossary-longragged package 287
glossary-mcols package 292
glossary-super package .. 10, 276, 298, 305, 309
glossary-superragged package 305
glossary-tree package 10, 311
\glossaryentry 186, 188, 323
glossaryentry (counter) 11, 206, 207
\glossaryentryfield
..... 208, 328–335, 337–340, 362
\glossaryentrynumbers
.. 9, 164, 191, 192, 201, 202, 205, 206, 325
\glossaryheader
..... 164, 200, 271, 273–275, 277–
281, 283, 284, 287–293, 295–297, 299,
300, 302, 306, 307, 309, 312–317, 320, 325
\glossarymark 41
\glossaryname 16, 36, 37
\glossarypostamble 164, 200, 325
\glossarypreamble 163, 199, 325
\glossarysection 163, 199, 325
glossarysubentry (counter) 12, 206–208
\glossarysubentryfield
..... 210, 328–335, 337–340, 362
\glossarytitle 163, 191, 199, 203, 325
\glossarytoctitle 8,
16, 17, 31, 34, 37, 41, 163, 191, 199, 204, 325
\glossentry 88, 192, 202, 210,
271, 273–278, 280, 288, 289, 291, 299,
301, 302, 306, 307, 309, 312, 314, 315, 318
\Glossentrydesc 361
\glossentrydesc
. 271–278, 280, 288, 289, 291, 299, 301,
302, 306–309, 312–314, 316, 318, 319, 361
\glossentryname
. 271–278, 280, 288, 289, 291, 299, 301,
302, 306, 307, 309, 312–315, 318, 319, 361
\Glossentrysymbol 362
\glossentrysymbol 271, 272, 280,
291, 302, 309, 312–314, 316, 318, 319, 362
\Gls 96, 218, 236
\gls 96, 175, 207, 218, 236
\gls@Alphpage 181, 185
\gls@alphpage 181, 185
\gls@arabicpage 181, 185
\gls@assign@desc 81, 86
\gls@assign@descplural
..... 237, 246, 248–251, 372, 375, 376

<code>\gls@assign@field</code>	<code>\gls@Romanpage</code>	181, 185
..... 71, 75, 76, 81, 83, 85, 86, 261, 262	<code>\gls@romanpage</code>	181, 185
<code>\gls@assign@firstpl</code>	<code>\gls@save@numberlist</code>	9
239–241, 243, 244, 246, 248–251, 372–376	<code>\gls@set@xr@key</code>	66
<code>\gls@assign@plural</code>	<code>\gls@suffixF</code>	39, 164, 166, 325, 327
239–241, 243, 244, 246, 248–251, 372–376	<code>\gls@suffixFF</code>	39, 164–167, 325, 327
<code>\gls@assign@symbolplural</code>	<code>\gls@text</code>	107
239–241, 243, 244, 249–251, 373, 374, 376	<code>\gls@the</code>	185
<code>\gls@begindocdefs</code>	<code>\gls@thissty</code>	27
<code>\gls@checkisacronymlist</code>	<code>\gls@tmp</code>	179, 247
<code>\gls@checkseeallowed</code>	<code>\gls@tmplen</code>	122, 317–319, 335, 336
66, 72, 173, 175	<code>\gls@tr@set@acronym@toctitle</code>	17
<code>\gls@checkseeallowed@preambleonly</code> ..	<code>\gls@tr@set@main@toctitle</code>	16
72	<code>\gls@tr@set@numbers@toctitle</code>	31
<code>\gls@codepage</code>	<code>\gls@tr@set@symbols@toctitle</code>	31
52, 171, 194	<code>\gls@wrglossary</code>	180
<code>\gls@defdocnewglossaryentry</code>	<code>\gls@xdyststring</code>	115
73, 93	<code>\gls@xindy@glsnumbersfalse</code>	29
<code>\gls@defglossaryentry</code>	<code>\gls@xindy@glsnumberstrue</code>	29
71–73, 81	<code>\gls@xr@key</code>	6, 7, 66
<code>\gls@disablepagerefexpansion</code> ..	<code>\glsaccsupp</code>	347
181, 185	<code>\glsacronymtrue</code>	17
<code>\gls@do@addxdyattribute</code>	<code>\glsacrpluralsuffix</code>	
46 35, 218, 227, 228, 232–234, 238	
<code>\gls@docclearpage</code>	<code>\glsacrshortcutsfalse</code>	32
43	<code>\glsacrshortcutstrue</code>	32
<code>\gls@dosubst</code>	<code>\glsacspace</code>	226, 228
115	<code>\glsadd</code>	159
<code>\gls@dotocitle</code>	<code>\glsadd options</code>	
191, 192, 203, 204	counter	158
<code>\gls@end@sanitizesort</code>	format	158, 215
22, 23	<code>\glsaddall options</code>	
<code>\gls@endcheck</code>	types	158
70, 71	<code>\GlsAddXdyAttribute</code>	45, 47, 321, 322
<code>\gls@glossary</code>	<code>\GlsAddXdyCounters</code>	45, 46, 62
179, 186–188	<code>\glsautomakefalse</code>	30
<code>\gls@gobbleopt</code>	<code>\glsautoprefix</code>	8, 204
62	<code>\glsapscase</code>	99, 100, 103–106,
<code>\gls@grplabel</code>	124–128, 142–148, 230, 231, 243, 248,	
268	349, 351, 353, 354, 356, 357, 359–361, 366	
<code>\gls@hypergroup prerun</code>	<code>\glscclearpage</code>	42
269	<code>\glscclosebrace</code>	49, 50, 164, 165, 325, 326
<code>\gls@ifnotmeasuring</code>	<code>\glscompositor</code>	38, 39, 49, 166, 324, 327
90, 91	<code>\glscounter</code>	19, 33, 45, 62, 85, 113, 321
<code>\gls@inlinepostchild</code>	<code>\glscurrententrylabel</code>	189, 192
271, 272, 328	<code>\glscurrentfieldvalue</code>	58, 59
<code>\gls@inlinesep</code>	<code>\glscustomtext</code>	99,
271, 328	102, 104, 105, 107, 124–128, 142–149,	
<code>\gls@inlinesubsep</code>		
271, 272, 328		
<code>\gls@islistofacronyms</code>		
18		
<code>\gls@istfilebase</code>		
38, 171		
<code>\gls@label</code>		
217		
<code>\gls@level</code>		
84, 85, 201		
<code>\gls@noidxglossary</code>		
175		
<code>\gls@nosetquote</code>		
82, 165, 167, 170		
<code>\gls@number</code>		
185		
<code>\gls@numberpage</code>		
181, 185		
<code>\gls@org@glossaryentryfield</code>		
192		
<code>\gls@org@glossarysubentryfield</code>		
192		
<code>\gls@org@insert</code>		
242, 245, 247		
<code>\gls@orgAlph</code>		
185		
<code>\gls@orgalph</code>		
185		
<code>\gls@orgarabic</code>		
185		
<code>\gls@orgnumber</code>		
185		
<code>\gls@orgRoman</code>		
185		
<code>\gls@orgromannumeral</code>		
185		
<code>\gls@orgthe</code>		
185		
<code>\gls@protected@pagefmts</code>		
115, 181, 182		

230, 231, 238, 239, 242, 243, 245, 247, 248, 349, 353, 355, 356, 358–361, 366, 367	\glstryfirstplural .. 97, 99, 100, 103, 132, 349, 351, 353, 354
\GlsDeclareNoHyperList 19	\glstryfirstpluralaccess 348
\glstdefaulttype 16, 40, 52, 53, 60, 61, 83, 98, 108, 179, 190, 191	\glstryfmt 61, 63
\glstdefmain 16, 63	\Glsentryfull 223, 232, 233, 368, 370
\glstdescriptionaccessdisplay 351–353, 361, 362	\glstryfull 223, 232, 233, 367, 370
\glstdescriptionpluralaccessdisplay 349–351	\Glsentryfullpl 223, 232, 233, 368, 370
\glstdescwidth 277– 279, 281, 282, 285–292, 299–302, 304–311	\glstryfullpl 223, 232, 233, 368, 370
\glstetoklabel 54–59, 67, 73, 78–82, 88, 91–95, 112, 149, 150, 157, 158, 175– 177, 183, 186, 192, 195–197, 201, 203, 207, 209, 254–258, 317, 341, 342, 377, 378	\glstryitem 271, 273–278, 280, 288, 289, 291, 299, 301, 302, 306, 307, 309, 312, 314, 315, 318, 328–335, 337–340
\glstdisplay 99, 108	\Glsentrylong 97, 146, 150, 156, 225, 226, 231, 232, 358, 360, 363, 366–368
\glstdisplayfirst 99, 108	\glstrylong 96, 107, 145, 147, 150, 156, 225–235, 245, 358, 360–371
\glstdisplaynumberlist 176	\glstrylongaccess 348
\glstdohyperlink 122, 123	\Glsentrylongpl 98, 148, 156, 225, 226, 230–232, 358, 363, 366–368
\glstdohypertarget 123	\glstrylongpl 97, 107, 147, 149, 156, 225–227, 230–233, 245, 252, 358, 363, 364, 366–370
\glstdoifexists 56, 58, 78–80, 90, 91, 124–129, 142– 148, 156, 158, 176, 177, 263–267, 358–362	\glstrylongpluralaccess 349
\glstdoifexistsordo 111, 149	\Glsentryname 133, 209, 361
\glstdoifexistsorwarn 203, 209, 210	\glstryname 133, 317, 361
\glstdoifnoexists 71, 81	\glstrynumberlist 156, 176
\glstdonohyperlink 113, 122, 123	\Glsentryplural 100, 103, 131, 350, 354
\glstdosanitizesort 13	\glstryplural 99, 100, 103, 131, 132, 349, 350, 353, 354
\glstentryaccess 347	\glstrypluralaccess 348
\glstentrycounter 213, 216	\Glsentryprefix 265
\glstentrycounterfalse 11	\glstryprefix 263, 266
\glstentrycounterlabel 207	\Glsentryprefixfirst 265
\GlsEntryCounterLabelPrefix 207	\glstryprefixfirst 264, 266
\glstentrycountertrue 12	\Glsentryprefixfirstplural 266
\glstentrycurrcount 93, 95	\glstryprefixfirstplural 264, 267
\Glsentrydesc 134, 209, 361	\Glsentryprefixplural 265
\glstentrydesc 101, 102, 134, 209, 351–353, 361	\glstryprefixplural 264, 267
\glstentrydescaccess 348	\glstentryprevcount 93–95
\Glsentrydescplural 135	\Glsentryshort 105, 142, 150, 227, 233, 357–359, 363, 369, 370
\glstentrydescplural 99, 100, 134, 135, 349–351	\glstryshort 105, 107, 142, 143, 150, 156, 223, 225–235, 356–359, 362–365, 367–371
\glstentrydescpluralaccess 348	\glstryshortaccess 348
\Glsentryfirst ... 97, 101, 104, 131, 352, 355	\Glsentryshortpl 105, 144, 227, 233, 356, 363, 369, 370
\glstryfirst 96, 101, 102, 104, 130, 131, 351, 352, 355	\glstryshortpl 105, 107, 144, 145, 156, 225– 227, 231–233, 252, 356, 358, 363, 367–370
\glstryfirstaccess 348	\glstryshortpluralaccess 348
\Glsentryfirstplural 98, 100, 103, 132, 350, 354	

<code>\Glsentrysymbol</code>	135, 210, 362	<code>\glsifhyperon</code>	110
<code>\glsentrysymbol</code>	101, 102, 135, 136, 209, 239, 243, 248, 351–353, 362	<code>\glsIfListOfAcronyms</code>	18
<code>\glsentrysymbolaccess</code>	348	<code>\glsifplural</code>	99, 102, 105, 106, 124–128, 142–148, 230, 239, 243, 245, 247, 349, 353, 356, 357, 359–361, 366
<code>\Glsentrysymbolplural</code>	136	<code>\glsifusetranslator</code>	26, 27, 36, 37, 379
<code>\glsentrysymbolplural</code> 99, 100, 136, 137, 239, 243, 248, 349–351	<code>\glsindexonlyfirstfalse</code>	27
<code>\glsentrysymbolpluralaccess</code>	348	<code>\glsinlinedescformat</code>	271, 328
<code>\Glsentrytext</code>	101, 104, 130, 351, 355	<code>\glsinlinedopostchild</code>	271, 328
<code>\glsentrytext</code>	101, 102, 104, 129, 130, 157, 189, 351, 352, 354, 355	<code>\glsinlineemptydescformat</code>	271, 328
<code>\glsentrytextaccess</code>	347	<code>\glsinlineformat</code>	271, 328
<code>\glsentrytype</code>	83	<code>\glsinlineparentchildseparator</code>	271, 328
<code>\Glsentryuseri</code>	137	<code>\glsinlinepostchild</code>	271, 328
<code>\glsentryuseri</code>	137	<code>\glsinlineseparator</code>	271, 328
<code>\Glsentryuserii</code>	138	<code>\glsinlinesubdescformat</code>	272, 328
<code>\glsentryuserii</code>	138	<code>\glsinlinesubnameformat</code>	271, 328
<code>\Glsentryuseriii</code>	139	<code>\glsinlinesubseparator</code>	272, 328
<code>\glsentryuseriii</code>	138, 139	<code>\glsinsert</code>	99– 106, 124–128, 142–148, 230, 231, 239, 242, 243, 245, 247, 248, 349–361, 366, 367
<code>\Glsentryuseriv</code>	139	<code>\glskeylisttok</code>	222, 223, 237–244, 246, 248–251, 253, 371–376
<code>\glsentryuseriv</code>	139, 140	<code>\glslabel</code> ..	82, 99–106, 111–113, 142–148, 225, 226, 230–232, 238, 239, 242, 243, 245, 247, 248, 349–358, 362, 363, 366–368
<code>\Glsentryuserv</code>	140	<code>\glslabeltok</code> 222, 223, 237–246, 248–253, 372–375
<code>\glsentryuserv</code>	140, 141	<code>\glslink</code>	223, 231, 233, 238, 367, 369
<code>\Glsentryuservi</code>	141	<code>\glslink options</code>	
<code>\glsentryuservi</code>	141	counter	109, 123, 260
<code>\glsesclocationstrue</code>	9	format	109, 123, 215
<code>\glsfieldfetch</code>	153	hyper	109, 111, 112, 123
<code>\glsfirstaccessdisplay</code>	351, 352, 355	local	109
<code>\glsfirstpluralaccessdisplay</code> 349, 350, 353, 354	<code>\glslinkcheckfirsthyperhook</code>	111
<code>\glsfirstpluralacessdisplay</code>	354	<code>\glslinkpostsetkeys</code>	112
<code>\glsгенacfmt</code>	225, 226, 232, 362, 363, 368	<code>\glslinkvar</code>	110
<code>\glsгенentryfmt</code> 225, 226, 231, 232, 237, 239, 241– 243, 245, 247, 250, 252, 362, 363, 367, 368	<code>\glslistdottedwidth</code>	275, 276, 329
<code>\glsgetgrouptitle</code> 270, 274, 275, 293–298, 313–316, 320	<code>\glslistgroupheaderfmt</code>	274, 275
<code>\glsгlossarymark</code>	41	<code>\glslistnavigationitem</code>	274, 275
<code>\glsgroupheading</code>	165, 201, 271, 273–275, 277, 278, 280, 288, 289, 291, 293–300, 302, 306, 307, 309, 312–317, 319, 320, 326	<code>\glslocalreset</code>	92
<code>\glsgroupskip</code>	164, 165, 201, 272, 274, 277, 279, 280, 283, 284, 288–291, 299, 301, 303, 306, 308, 310, 313, 314, 316, 319, 325	<code>\glslocalunset</code>	93, 124–129
<code>\glshyperfirstfalse</code>	232, 368	<code>\glslongaccessdisplay</code>	358, 360–372
<code>\glshyperfirsttrue</code>	27	<code>\glslongkey</code>	376
<code>\glshyperlink</code>	189	<code>\glslongpluralaccessdisplay</code> 358, 363, 364, 366–370, 372
<code>\glshypernavsep</code>	270	<code>\glslongpluralkey</code>	376
<code>\glshypernumber</code>	40, 216, 217	<code>\glslongtok</code> 222, 223, 225, 226, 231, 232, 237– 246, 248–253, 362, 363, 367, 368, 371–376

<code>\glsLTpenaltycheck</code>	286, 287	<code>\glsprestandardsort</code>	13
<code>\glsmcols</code>	293–298	<code>\glsreset</code>	92
<code>\glsnameaccessdisplay</code>	361, 362	<code>\glsresetentrycounter</code>	206
<code>\glsnamefont</code>	209, 210, 341, 342, 361	<code>\glsresetentrylist</code>	164, 200, 206, 325
<code>\glsnavhyperlink</code>	270	<code>\glsresetsubentrycounter</code> ...	208, 271, 328
<code>\glsnavhyperlinkname</code>	268, 269	<code>\glssanitizesortfalse</code>	24, 25
<code>\glsnavhypertarget</code>		<code>\glssanitizesorttrue</code>	24, 25
.....	274, 275, 294–298, 314–316, 320	<code>\glssavenumberlistfalse</code>	9
<code>\glsnavigation</code>		<code>\glssavewritesfalse</code>	30
.....	274, 275, 293–298, 313, 315, 316, 320	<code>\glssееformat</code>	163, 175, 177, 325
<code>\glsnextpages</code>	9, 67, 192	<code>\glssееitem</code>	189
<code>\glsnogroupskipfalse</code>	11	<code>\glssееitemformat</code>	189
<code>\glsnoidxdisplayloc</code>	177, 178	<code>\glssееlastsep</code>	189
<code>\glsnoidxdisplayloclisthandler</code> ...	176	<code>\glssееlist</code>	188
<code>\glsnoidxloclist</code>	176, 201, 202	<code>\glssееsep</code>	189
<code>\glsnoidxloclisthandler</code>	202	<code>\glsssetexpandfield</code>	21, 24, 25
<code>\glsnoidxnumberlistloophandler</code> ...	177	<code>\glsssetnoexpandfield</code>	21, 22, 24, 25
<code>\glsnoidxstripaccents</code>	23	<code>\GlsSetQuote</code>	82, 165
<code>\glsnomakeindexwarning</code>	167	<code>\glssettoctitle</code>	37, 191
<code>\glsnonextpages</code>	66, 192	<code>\glssshortaccessdisplay</code>	
<code>\glsnopostdotfalse</code>	11	356–359, 362–365, 367–372
<code>\glsnoxindywarning</code> 39, 45–47, 50–52, 160, 161		<code>\glssshortkey</code>	376
<code>\glsnumberformat</code>	157	<code>\glssshortpluralaccessdisplay</code>	
<code>\glsnumberlistloop</code>	177	356, 358, 363, 367–370, 372
<code>\glsnumbersgroupname</code>	31, 37, 213	<code>\glssshortpluralkey</code>	376
<code>\glsnumlistlastsep</code>	157, 176	<code>\glssshorttok</code>	
<code>\glsnumlistparser</code>	157, 174	222, 223, 237–246, 248–253, 372–376
<code>\glsnumlistsep</code>	157, 176	<code>\glssshowtarget</code>	6
<code>\glsopenbrace</code>	49, 50, 164, 165, 325, 326	<code>\glssortnumberfmt</code>	14, 15
<code>\glsorder</code>	28, 171–173, 197, 198	<code>\glsspace</code>	219
<code>\glsorg@endtheglossary</code>	5	<code>\glssstepentry</code>	207
<code>\glsorg@PrintChanges</code>	5	<code>\glssstepsubentry</code>	208
<code>\glsorg@theglossary</code>	5	<code>\glsssubentrycounterfalse</code>	12
<code>\glspagelistwidth</code> 278, 279, 281, 282, 285,		<code>\glsssubentrycounterlabel</code>	208
286, 289–292, 300–302, 304, 305, 307–311		<code>\glsssubentryitem</code> 272, 273, 275–278, 280,	
<code>\glspatchLToutput</code>	283–285	288, 289, 291, 299, 301, 302, 306, 308,	
<code>\glspenaltygroupskip</code>	283, 284	309, 313, 314, 316, 318, 328–335, 337–340	
<code>\glsspercentchar</code>	73, 164, 165	<code>\glsssymbolaccessdisplay</code> ...	351–353, 362
<code>\Glspl</code>	97, 236	<code>\glssymbolpluralaccessdisplay</code> .	349–351
<code>\glspl</code>	97, 236	<code>\glssymbolsgroupname</code>	31, 37, 213
<code>\glspluralaccessdisplay</code> 349, 350, 353, 354		<code>\glstarget</code>	210, 211,
<code>\glspluralsuffix</code>		272–278, 280, 288, 289, 291, 299, 301,	
.....	35, 85, 225–227, 363, 364, 368–370	302, 306–309, 312–316, 318, 319, 328–340	
<code>\glspostdescription</code>		<code>\glstextaccessdisplay</code> ..	351, 352, 354, 355
. 37, 272–275, 277, 288, 299, 306, 312–		<code>\glstextformat</code>	111, 113
314, 316, 318, 319, 328–331, 333–337, 339		<code>\glstextup</code>	35, 370
<code>\glspostinline</code>	271	<code>\glstildechar</code>	46, 164, 165
<code>\glspostlinkhook</code>		<code>\glstranslatefalse</code>	26, 27
.....	111, 124–129, 142–149, 359–361	<code>\glstranslatetrue</code>	26, 27

<code>\glstreechildpredesc</code>	313, 314	<code>\if@gl@debug</code>	5, 20, 180
<code>\glstreegroupheaderfmt</code>		<code>\if@gl@docloaded</code>	4, 16, 179
.....	293–298, 313, 315, 316, 320	<code>\if@gl@isacronymlist</code>	111
<code>\glstreeindent</code> ..	314, 316, 318, 319, 334–336	<code>\if@openright</code>	43
<code>\glstreeitem</code>	293, 294, 312	<code>\ifbool</code>	17, 28, 30, 55, 99–101
<code>\glstreenamebox</code>	318, 319	<code>\ifboolexpr</code>	36, 60, 212
<code>\glstreenamefmt</code>	311–315, 317–319	<code>\ifcase</code>	5, 6, 8, 26, 66, 204, 313, 333
<code>\glstreenavigationfmt</code>		<code>\ifcsdef</code>	26, 37, 43, 69, 70,
.....	293–298, 313, 315, 316, 320	76–80, 108, 181, 194, 195, 197, 199, 214, 224	
<code>\glstreepredesc</code>	312, 314, 316	<code>\ifcsequal</code>	56, 57, 263
<code>\glstreesubitem</code>	293, 312	<code>\ifcsstring</code>	80
<code>\glstreesubsubitem</code>	293, 312	<code>\ifcsundef</code>	7, 14, 29, 33, 36, 40, 41,
<code>\glstype</code> ..	111, 112, 124–129, 142–149, 359–361	43, 54, 61, 63, 65, 83, 85, 86, 93, 94, 109,	
<code>\glsucmarkfalse</code>	11	113, 114, 122, 171, 179, 190, 193, 195,	
<code>\glsucmarktrue</code>	11	203, 204, 208, 212–215, 217, 224, 270, 327	
<code>\glsunset</code>	90, 92, 94, 95, 124–129	<code>\ifdef</code>	58, 67, 72,
<code>\glsupacrpluralsuffix</code>		73, 90, 109, 149, 153, 176, 177, 218, 311, 312	
.....	227, 228, 234, 240, 244, 247, 249	<code>\ifdefempty</code>	19, 42, 43, 53, 57–59,
<code>\GlsUseAcrEntryDispStyle</code> ...	224, 227–	63, 99, 102, 105, 174, 200, 201, 222, 224,	
230, 232, 234, 235, 364, 365, 368, 370, 371		230, 238, 242, 245, 247, 349, 353, 356, 366	
<code>\GlsUseAcrStyleDefs</code>	224, 227–	<code>\ifdefequal</code>	56–59, 69–71, 74, 84, 88, 201
230, 232, 234, 235, 364, 365, 368, 370, 371		<code>\ifdefstrequal</code>	80
<code>\glswrallowprimitivemodstrue</code>	183	<code>\ifdefstring</code>	
<code>\glswrite</code> ...	161–167, 172, 178, 179, 323–327	... 10, 36, 60, 171–173, 197, 198, 200, 202	
<code>\glswritedefhook</code>	73	<code>\ifdefvoid</code>	22, 87, 201, 202
<code>\glswriteentry</code>	181	<code>\ifdim</code>	226, 286, 317
<code>\glswritefiles</code>	30, 178	<code>\iffalse</code>	86, 92
<code>\glsxindyfalse</code>	29	<code>\ifFileExists</code>	10, 26, 193
<code>\glsxindytrue</code>	29	<code>\ifglossaryexists</code> ..	40, 52, 56, 170–172, 191
H			
<code>\H</code>	23	<code>\ifgl@sanitize@description</code>	24
<code>\hangindent</code>		<code>\ifgl@sanitize@name</code>	24
297, 298, 311, 314–316, 318–320, 333–336		<code>\ifgl@sanitize@symbol</code>	24
<code>\hbox</code>	90, 275, 276, 329	<code>\ifgl@xindy@gl@numbers</code>	52
<code>\hfill</code>	275, 276, 329	<code>\ifgl@acr@description</code>	251
<code>\hline</code> ...	277–279, 281, 288–290, 292, 299–311	<code>\ifgl@acr@dua</code>	241, 247, 250–252
<code>\hsize</code>	276, 287, 298, 305	<code>\ifgl@acr@footnote</code>	111, 251, 252
<code>\hspace</code>	312	<code>\ifgl@acr@synonym</code>	16, 17
<code>\hss</code>	275, 276, 329	<code>\ifgl@acr@shortcuts</code>	32, 236
<code>\ht</code>	286	<code>\ifgl@acr@smallcaps</code>	
<code>\hyperdef</code>	33	240, 242, 244, 246, 249, 250
<code>\hyperlink</code>	109, 122, 216	<code>\ifgl@acr@smaller</code>	240, 242, 244, 247
<code>hyperref</code> package	187, 190, 215, 260	<code>\ifgl@automake</code>	30, 174
<code>\hypertarget</code>	122	<code>\ifgl@desc@suppressed</code>	271
I			
<code>\IeC</code>	23	<code>\ifgl@sentry@counter</code>	11, 12, 32, 206, 207
<code>\if</code>	115, 186, 322	<code>\ifgl@sentry@exists</code>	55, 72, 73, 81, 84
<code>\if@endfor</code>	269	<code>\ifgl@esc@locations</code>	183
		<code>\ifgl@has@children</code>	271, 328

longtable package	276, 283, 287	\newacronymhook	223, 238, 240, 242, 244, 246, 249, 251, 253, 371
\LT@end@pen	286	\newacronymstyle	225–230, 232, 234, 235
\LT@err	286	\newcommand	6–23, 25– 28, 30–35, 37–47, 49–63, 65–81, 87, 88, 90–93, 95–99, 102, 105, 107, 108, 110– 113, 115, 116, 122–161, 167, 170–172, 174, 177, 178, 180–184, 186–192, 194– 199, 201–203, 205–226, 235, 237–259, 262–266, 268–270, 272, 273, 286, 293, 311, 312, 317, 327, 345–347, 362, 376–378
\LT@foot	286, 287	\newcount	14, 69
\LT@head	286, 287	\newcounter	11, 12
\LT@lastfoot	286	\newenvironment	208
\LT@output	286	\newglossary	16, 17, 31, 62, 173
M			
\makeatletter	72, 193	\newglossaryentry 6, 32, 68, 69, 72, 93, 223, 237, 239, 241, 243, 245, 248, 250, 252, 372–375
\makebox	275, 276, 317–319, 329, 335, 336	\newglossaryentry options	
makeglossaries	28, 38, 51, 52, 61, 167, 173, 193	access	344, 345
\makeglossaries 6, 7, 30, 34, 66, 174–176, 178, 194	counter	65
\makeglossary	171, 173	description 28, 64, 69, 71, 82, 133, 151, 218, 246, 343
makeindex	381	descriptionaccess	346, 348
makeindex	9, 13, 29, 30, 35, 38– 40, 44, 60–62, 64, 89, 114, 117, 159, 163, 165, 167, 170, 179, 184–186, 211, 322, 323	descriptionplural	134, 343
delim_n	40	descriptionpluralaccess	346, 348
delim_r	40	entrycounter	205
page_compositor	38	first	65, 85, 123, 130, 152, 244, 249, 342
special characters	116, 159	firstaccess	346, 348
\makenoidxglossaries	6, 7, 66, 174, 177, 178	firstplural	65, 132, 152, 343
\MakeTextUppercase	4	firstpluralaccess	346, 348
\MakeUppercase	350, 352, 359, 361	format	162
\marginpar	6	long	105, 155, 343
\markboth	41	longaccess	347, 348
\mbox	158, 274, 297, 298, 317, 329	longplural	155, 343
memoir class	180	longpluralaccess	347, 349
\memUHead	41	name	64, 68, 71, 82, 133, 150, 189, 342
\MessageBreak	20, 60, 191, 341, 379, 380	nonumberlist	66, 67
mfistuc package	1	parent	66, 71
\mfistucMakeUppercase 4, 41, 42, 77, 100, 102–106, 130–141, 143, 145, 147, 149, 223, 230–233, 243, 248, 266, 267, 354–358, 366, 367, 369, 370	plural	64, 85, 131, 343
\midrule	283, 284	pluralaccess	346, 348
\month	161, 165, 323, 326	prefix	261
multicol package	292	prefixfirst	261
N			
\n	166, 326	prefixfirstplural	262
\NeedsTeXFormat	4, 261, 321, 327, 341, 379	prefixplural	262
\new@glossaryentry	72, 176	see	6, 9, 66, 71, 173, 175
\new@ifnextchar	60, 76, 77, 96, 97, 123–127, 129–148, 219–221, 263–266	short	105, 155, 343
\newacronym	217, 222, 238, 240, 242, 244, 246, 249, 251, 253	shortaccess	346, 348
		shortplural	155, 343

shortpluralaccess	346, 348	\ns@Acrlong	146
sort	64, 153, 180, 211	\ns@acrlong	145
symbol	64, 65,	\ns@ACRlongpl	148
135, 240, 242, 244, 249, 280, 302, 342–344		\ns@Acrlongpl	148
symbolaccess	346, 348	\ns@acrlongpl	147
symbolplural	136, 343	\ns@ACRshort	143
symbolpluralaccess	346, 348	\ns@Acrshort	142
text	64, 65, 123, 129, 151, 240, 244, 342	\ns@acrshort	141, 142
textaccess	346, 347	\ns@ACRshortpl	144
type	16, 65, 98, 153	\ns@Acrshortpl	144
user1	137, 153, 344	\ns@acrshortpl	143
user2	137, 154	\ns@newglossary	61
user3	138, 154	\null	115–122, 167–170, 193
user4	139, 154	\number	14, 72, 85, 95, 182, 185, 210, 342
user5	140, 154	\numberline	44
user6	141, 155, 344	\numexpr	95
\newglossarystyle		O	
270, 273–285, 287–310, 312–317, 319, 320		\O	23
\newif	4, 5, 18, 25, 29, 183	\o	23
\newlength	122, 276, 287, 298, 305, 315	\OE	23
\newrobustcmd		\oe	23
71, 72, 77, 96, 97, 111, 123–148,		\openout	72, 161, 165, 171, 323, 326
150–156, 158, 159, 218–221, 262–266, 317		\OR	252
\newterm	32	\or	5, 7, 8, 27, 204, 313, 333
\newtoks	116, 170, 222	\org@glossaryentrynumbers	192, 206
\newwrite	72, 161, 165, 170, 172	\org@glossarytitle	191
\nfss@text	6	\org@glspostdescription	37
ngerman package	167	\org@ifKV@glslink@hyper	112, 113
\noalign	286	\outputpenalty	286
\nobreak	274, 287, 329	P	
\noexpand	18,	\p@	273, 293, 311, 312
34, 45, 46, 72, 86, 87, 108, 113–115,		\p@glsl@hyp@opt	110
121, 122, 157, 167–172, 174, 182, 183,		package options:	
186, 189, 193, 194, 196, 197, 208–210,		acronym	16, 17, 34, 190, 218
218, 223, 237, 239, 241, 243, 245, 246,		true	17
248, 250, 252, 253, 321, 341, 342, 372–376		counter	19
\nohyperpage	215	debug	
\noindent	210, 294–296, 298, 315, 316	showtargets	6
\noist	326, 327	description	244, 245
\nopostdesc	32, 37, 81, 192, 328	dua	242, 244, 245
\normalbaselineskip	286	entrycounter	11, 205, 206
\nr	5, 6, 8, 26, 66, 204	true	11
\ns@ACRfull	220	esclocations	404
\ns@Acrfull	219, 220	false	9
\ns@acrfull	219	footnote	124–129, 240, 242, 244, 246
\ns@ACRfullpl	221	hyperfirst	
\ns@Acrfullpl	221	false	124–129
\ns@ACRfullpl	220		
\ns@ACRlong	146		

indexonlyfirst	388	\phantomsection	43
makeindex	163, 260	polyglossia package	25, 36
nogroupskip	277, 279, 280, 283, 284, 288, 289, 291, 299, 301, 303, 306, 308, 310	\printglossaries	173
nolist	253	\printglossary	17, 20, 31, 173, 190, 205
nolong	254, 276	\printglossary options	
nomain	16	entrycounter	205
nonumberlist	9	nogroupskip	204
nosuper	254	nonumberlist	205
notree	254	nopostdot	204
nowarn	5	numberedsection	204
numberline	7	style	204
sanitize	24, 64, 150, 151	subentrycounter	205
sanitizesort	21	title	203
savewrites	30, 385	toctitle	204
false	170	type	16, 189, 203
true	172, 178	\printindex	31
section	7, 42	\printnoidxglossaries	175
sort		\printnoidxglossary	
def	12, 13	175, 178, 190, 197, 198, 205
none	12	\printnoidxglossary options	
standard	12	sort	205
use	12, 13, 404	\printnumbers	31
style	8, 253, 254	\printsymbols	31
subentrycounter	12, 205, 206	\ProcessOptions	261, 341
toc	7	\ProcessOptionsX	32
true	7	\protect	44, 107, 225–227, 232, 233, 239, 243, 245, 358, 362, 363, 368, 369
translate	26	\protected@edef	8, 46, 48, 51, 53, 83, 87, 89, 99–101, 107, 114, 157, 181, 183, 186, 204, 208, 210, 213, 214, 223, 247, 252, 262, 268, 322, 323, 341, 342, 347
false	26	\protected@write	60, 61, 162, 163, 172, 173, 175, 178, 181, 189, 269, 323
translator	25	\protected@xdef	13–15, 18, 23, 70, 89, 185, 345
xindy	29, 163, 260	\providecommand	17, 34, 35, 42, 60, 95, 123, 163, 172, 175, 193, 208, 210, 273, 293, 311
\PackageError	6, 7, 15, 30, 34, 45, 52, 55, 56, 60, 65, 68, 69, 75–80, 83, 84, 93, 109, 149, 170, 171, 173, 176–178, 197–199, 204, 205, 213, 214, 224, 225, 241, 242, 247, 250, 327, 349	\ProvidesFile	35
\PackageInfo	5, 6, 171, 180	\ProvidesPackage	
\PackageWarning	5, 19	4, 261, 268, 270, 273, 276, 282, 287, 292, 298, 305, 311, 321, 327, 341, 379
\PackageWarningNoLine ...	5, 6, 20, 379, 380		
\pagegoal	286		
\pagelistname	37, 279, 281, 283, 284, 290, 292, 301–305, 308–311		
\par	37, 210, 211, 273–275, 293, 295–298, 311, 312, 314–320, 329, 334–336		
\parindent			
.....	293–298, 312, 314–316, 318–320, 334–336		
\parskip	293–296, 312, 314, 315		
\PassOptionsToPackage	261, 341		
\penalty	286		

R

\r	23
\raggedright	285–292, 306–311
\raisebox	122
\ref	207
\refstepcounter	207
\relax	5, 8, 10, 14–17, 26, 27, 32, 33, 47, 60, 65, 66, 70, 72, 84, 86,

<code>\tablehead</code>	299–311	<code>\ttfamily</code>	6
<code>\tabletail</code>	299–311	<code>\TX@trial</code>	90
<code>\tabularnewline</code>	277–281, 283, 284, 288–292, 299–311, 331, 332, 337, 338	<code>\typeout</code>	20
<code>\texorpdfstring</code>	153	U	
<code>\textbar</code>	270	<code>\u</code>	23
<code>\textbf</code>	210, 217, 311, 333–336	<code>\uccode</code>	200
textcase package	4	<code>\undef</code>	67, 73, 190
<code>\textit</code>	217	<code>\unskip</code>	81, 275, 276, 329
<code>\textmd</code>	217	<code>\unvbox</code>	286, 287
<code>\textrm</code>	216	<code>\usedictionary</code>	36
<code>\textsc</code>	217, 227, 228, 234, 240, 244, 247, 249, 370	<code>\usepackage</code>	197, 198
<code>\textsf</code>	216	V	
<code>\textsl</code>	217	<code>\v</code>	23
<code>\textsmaller</code>	227, 228, 234, 240, 244, 247, 249, 370	<code>\val</code>	5, 6, 8, 26, 66, 204
<code>\texttt</code>	6, 217	<code>\vbox</code>	286, 287
<code>\textulc</code>	218	<code>\vsize</code>	286, 287
<code>\textup</code>	217, 218	<code>\vskip</code>	273, 286, 293, 311
<code>\TH</code>	23	<code>\vss</code>	286, 287
<code>\th</code>	23	W	
<code>\the</code>	34, 36, 46, 51, 53, 61, 116–121, 161, 165, 167–169, 179, 181, 185, 189, 201, 209, 210, 216, 223, 225, 226, 231, 232, 237, 239, 241, 243, 245, 246, 248, 250, 252, 253, 321, 323, 326, 342, 362, 363, 367, 368, 371–376	<code>\warn@nomakeglossaries</code>	173–175
<code>\the@numberlist</code>	157	<code>\warn@noprintglossary</code>	173–175, 192
<code>\theglossary</code>	5	<code>\write</code>	73, 95, 161– 167, 171, 172, 175, 179, 193, 194, 323–327
<code>\theglossaryentry</code>	11, 207	<code>\writeist</code>	170, 171, 327
<code>\theglossarysubentry</code>	12, 207	X	
<code>\theglentrycounter</code>	113, 114, 183, 185, 322, 323	<code>\x</code>	216
<code>\theH</code>	187	<code>\xatlevel@</code>	113
<code>\theHglossaryentry</code>	11	<code>\xcapitalisewords</code>	153
<code>\theHglossarysubentry</code>	12	<code>\xdef</code>	78, 83–86, 192, 269
<code>\theHglentrycounter</code> ...	114, 183, 185, 186	<code>\xglaccsupp</code>	347
<code>\thesection</code>	33	<code>\xifinlistcs</code>	194, 195, 198
<code>\this@dialect</code>	36, 379, 380	xindy	381
<code>\toks@</code> ..	33, 34, 36, 46, 51, 53, 61, 116–121, 167–169, 189, 208–210, 216, 321, 341, 342	xindy	9, 13, 29, 30, 38, 39, 44, 47, 49, 51–53, 89, 120, 121, 160, 161, 163, 179, 183, 184, 186, 193, 211, 260, 322
<code>\toprule</code>	283, 284	<code>\xmakefirstuc</code>	99–101, 107, 149, 151, 262
tracklang package	35, 379	<code>\xspace</code>	218
<code>\trans@languages</code>	36	xspace package	4, 217
<code>\translate</code>	36, 37	Y	
<code>\translatelet</code>	16, 17, 31	<code>\year</code>	161, 165, 323, 326
translator package	16, 17, 26, 31, 35–37, 190	Z	
		<code>\z@</code>	286