

# Documented Code For glossaries v4.32

Nicola L.C. Talbot

Dickimaw Books

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

2017-08-24

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 v4.32: L<sup>A</sup>T<sub>E</sub>X2e 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

<b>1</b>	<b>Main Package Code</b>	<b>4</b>
1.1	Package Definition	4
1.2	Package Options	5
1.3	Predefined Text	32
1.4	Xindy	42
1.5	Loops and conditionals	51
1.6	Defining new glossaries	57
1.7	Defining new entries	62
1.8	Resetting and unsetting entry flags	87
1.9	Keeping Track of How Many Times an Entry Has Been Unset	91
1.10	Loading files containing glossary entries	95
1.11	Using glossary entries in the text	96
1.12	Adding an entry to the glossary without generating text	155
1.13	Creating associated files	157
1.14	Writing information to associated files	175
1.15	Glossary Entry Cross-References	182
1.16	Displaying the glossary	184
1.17	Acronyms	213
1.18	Predefined acronym styles	217
1.19	Predefined Glossary Styles	249
1.20	Debugging Commands	250
1.21	Compatibility with version 2.07 and below	255
<b>2</b>	<b>Prefix Support (glossaries-prefix Code)</b>	<b>257</b>
<b>3</b>	<b>Glossary Styles</b>	<b>264</b>
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	264
3.2	In-line Style (glossary-inline.sty)	266
3.3	List Style (glossary-list.sty)	269
3.4	Glossary Styles using longtable (the glossary-long package)	272
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	278
3.6	Glossary Styles using longtable (the glossary-longragged package)	283
3.7	Glossary Styles using multicol (glossary-mcols.sty)	288
3.8	Glossary Styles using supertabular environment (glossary-super package)	294
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	301
3.10	Tree Styles (glossary-tree.sty)	307

<b>4 Backwards Compatibility</b>	<b>317</b>
4.1 glossaries-compatible-207 . . . . .	317
4.2 glossaries-compatible-307 . . . . .	323
<b>5 Accessibility Support (glossaries-accsupp Code)</b>	<b>337</b>
5.1 Defining Replacement Text . . . . .	338
5.2 Accessing Replacement Text . . . . .	341
5.3 Displaying the Glossary . . . . .	357
5.4 Acronyms . . . . .	358
5.5 Debugging Commands . . . . .	373
<b>6 Multi-Lingual Support</b>	<b>375</b>
6.1 Polyglossia Captions . . . . .	375
<b>Glossary</b>	<b>377</b>
<b>Change History</b>	<b>378</b>
<b>Index</b>	<b>401</b>

# 1 Main Package Code

## 1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2017/08/24 v4.32 (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]{\marginpar{\texttt{\small #1}}}

```

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

```

61 \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`

```

62 \newcommand*{\@gls@see@noindex}{%
63   \PackageError{glossaries}%
64   {'\gls@xr@key' key may only be used after \string\makeglossaries\space
65   or \string\makenoidxglossaries}%
66   {You must use \string\makeglossaries\space
67   or \string\makenoidxglossaries\space before defining
68   any entries that have a '\gls@xr@key' key}%
69 }

```

`seenoinindex`

```

70 \define@choicekey{glossaries.sty}{seenoinindex}[\val\nr]{error,warn,ignore}{%
71   \ifcase\nr
72     \renewcommand*{\@gls@see@noindex}{%
73       \PackageError{glossaries}%
74       {'\gls@xr@key' key may only be used after \string\makeglossaries\space
75       or \string\makenoidxglossaries}%
76       {You must use \string\makeglossaries\space
77       or \string\makenoidxglossaries\space before defining
78       any entries that have a '\gls@xr@key' key}%
79     }%
80   \or
81     \renewcommand*{\@gls@see@noindex}{%
82       \GlossariesWarning{'\gls@xr@key' key ignored}%
83     }%
84   \or
85     \renewcommand*{\@gls@see@noindex}{}%
86   \fi
87 }

```

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.

```
88 \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.
```

```
89 \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.
```

```
90 \ifcsundef{chapter}%
91   {\newcommand*{\@@glossarysec}{section}}%
92   {\newcommand*{\@@glossarysec}{chapter}}

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

```
93 \define@choicekey{glossaries.sty}{section}{part,chapter,section,%
94 subsection,subsubsection,paragraph,subparagraph}[section]{%
95   \renewcommand*{\@@glossarysec}{#1}}

Determine whether or not to use numbered sections.
```

glossarysecstar

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

lossaryseclabel
```

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

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

```
98 \newcommand*{\glsautoprefix}{}

numberedsection
```

```
99 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
100 false,nolabel,autolabel,nameref}[nolabel]{%
101   \ifcase\nr\relax
102     \renewcommand*{\@@glossarysecstar}{*}%
103     \renewcommand*{\@@glossaryseclabel}{}%
104   \or
105     \renewcommand*{\@@glossarysecstar}{}%
106     \renewcommand*{\@@glossaryseclabel}{}%
107   \or
108     \renewcommand*{\@@glossarysecstar}{}%
109     \renewcommand*{\@@glossaryseclabel}{%
110       \label{\glsautoprefix\@glo@type}}%
111   \or
```

```

112 \renewcommand*{\@glossarysecstar}{*}%
113 \renewcommand*{\@glossaryseclabel}{%
114 \protected@edef\@currentlabelname{\glossarytoctitle}%
115 \label{\glsautoprefix\@glo@type}}%
116 \fi
117 }

```

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`

```

118 \ifpackageloaded{classicthesis}
119 {\newcommand*{\@glossary@default@style}{index}}
120 {\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`.

```

121 \define@key{glossaries.sty}{style}{%
122 \def\@glossary@default@style{#1}%
123 }

```

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`

```

124 \newcommand*{\@gls@declareoption}[2]{%
125 \DeclareOptionX{#1}{#2}%
126 \DeclareOption{#1}{#2}%
127 }

```

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`

```

128 \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).

```

129 \@gls@declareoption{nonumberlist}{%

```



```

130 \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
131 }

savenumberlist Provide means to store the number list for entries.
132 \define@boolkey{glossaries.sty}[gls]{savenumberlist}[true]{}
133 \glssavenumberlistfalse

eautonumberlist
134 \newcommand*\@glo@seeautonumberlist{}

eautonumberlist Automatically activates number list for entries containing the see key.
135 \@gls@declareoption{seeautonumberlist}{%
136 \renewcommand*\@glo@seeautonumberlist}{%
137 \def\@glo@prefix{\glsnextpages}%
138 }%
139 }

\@gls@loadlong
140 \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.
141 \@gls@declareoption{nolong}{\renewcommand*\@gls@loadlong{}}

\@gls@loadsuper The package isn't loaded if isn't installed.
142 \IfFileExists{supertabular.sty}{%
143 \newcommand*\@gls@loadsuper{\RequirePackage{glossary-super}}}%
144 \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.
145 \@gls@declareoption{nosuper}{\renewcommand*\@gls@loadsuper{}}

\@gls@loadlist
146 \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.
147 \@gls@declareoption{nolist}{%
148 \renewcommand*\@gls@loadlist}{%
149 \ifdefstring{\@glossary@default@style}{list}%
150 {\let\@glossary@default@style\relax}%
151 }%
152 }%
153 }

```

`\@gls@loadtree`

```

154 \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.
155 \@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).
156 \@gls@declareoption{nostyles}{%
157 \renewcommand*{\@gls@loadlong}{}%
158 \renewcommand*{\@gls@loadsuper}{}%
159 \renewcommand*{\@gls@loadlist}{}%
160 \renewcommand*{\@gls@loadtree}{}%
161 \let\@glossary@default@style\relax
162 }

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.)
163 \newcommand*{\glspostdescription}{%
164 \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
165 }

nopostdot Boolean option to suppress post description dot
166 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
167 \glsnopostdotfalse

nogroupskip Boolean option to suppress vertical space between groups in the pre-defined styles.
168 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
169 \glsnogroupskipfalse

ucmark Boolean option to determine whether or not to use use upper case in definition of \gls glossarymark

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

171 \@ifclassloaded{memoir}
172 {%
173 \glsucmarktrue
174 }%
175 {%
176 \glsucmarkfalse
177 }

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.
178 \define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
179 \glsentrycounterfalse

```

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

```
180 \define@key{glossaries.sty}{counterwithin}{%
181   \renewcommand*{\@gls@counterwithin}{#1}%
182   \glsentrycountertrue
183 }
```

s@counterwithin The default value is no parent counter:

```
184 \newcommand*{\@gls@counterwithin}{}
```

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.

```
185 \define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
186 \glssubentrycounterfalse
```

default@sorttype Initialise default sort for \printnoidxglossary

```
187 \newcommand*{\@glo@default@sorttype}{standard}
```

sort Define the sort method: sort=standard (default), sort=def (order of definition) or sort=use (order of use).

```
188 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
189   \renewcommand*{\@glo@default@sorttype}{#1}%
190   \csname @gls@setupsort@#1\endcsname
191 }
```

sprestandardsort

```
\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.

```
192 \newcommand*{\glsprestandardsort}[3]{%
193   \glsdosanitizesort
194 }
```

check@sortallowed

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

upsort@standard Set up the macros for default sorting.

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

Store entry information when it's defined.

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

No count register required for standard sort.

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

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

```

199 \def\@gls@defsort##1##2{%
200   \ifx\@glo@sort\@glsdefaultsort
201     \let\@glo@sort\@glo@name
202   \fi

203   \let\glsdosanitizesort\@gls@sanitizesort
204   \glsprestandardsort{\@glo@sort}{##1}{##2}%
205   \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
206 }%
```

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

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

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

```

208 \let\@glo@check@sortallowed\@gobble
209 }
```

Set standard sort as the default:

```
210 \@gls@setupsort@standard
```

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

```

211 \newcommand*\glsortnumberfmt[1]{%
212   \ifnum#1<100000 0\fi
213   \ifnum#1<10000 0\fi
214   \ifnum#1<1000 0\fi
215   \ifnum#1<100 0\fi
216   \ifnum#1<10 0\fi
217   \number#1%
218 }
```

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

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

Store entry information when it's defined.

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

Defined count register associated with the glossary.

```

221 \def\@gls@defsortcount##1{%
222   \expandafter\global
223   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
224 }%
```

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

```
225 \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.

```
226   \ifcsundef{glossary@##1@sortcount}%
```

```

227   {\@gls@defsortcount{##1}}}%
228   {}}%
229   \expandafter\global\expandafter
230   \advance\csname glossary@##1@sortcount\endcsname by 1\relax
231   \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
232     \expandafter\glssortnumberfmt
233     {\csname glossary@##1@sortcount\endcsname}}}%
234   }%

```

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

```

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

```

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

```

236   \let\@glo@check@sortallowed\@gobble
237 }

```

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

```

238 \newcommand*{\@gls@setupsort@use}{%

```

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

```

239   \let\do@glo@storeentry\@gobble

```

Defined count register associated with the glossary.

```

240   \def\@gls@defsortcount##1{%
241     \expandafter\global
242     \expandafter\newcount\csname glossary@##1@sortcount\endcsname
243   }%

```

Initialise the sort key to empty.

```

244   \def\@gls@defsort##1##2{%
245     \expandafter\gdef\csname glo@##2@sort\endcsname{%
246   }%

```

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

```

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

```

Get the parent, if one exists

```

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

```

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

```

249     \ifx\@glo@parent\@empty
250     \else
251       \expandafter\@gls@setsort\expandafter{\@glo@parent}%
252     \fi

```

Set index information for this entry

```

253     \edef\@glo@type{\csname glo@##1@type\endcsname}%
254     \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
255     \ifx\@gls@tmp\@empty
256       \expandafter\global\expandafter
257       \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
258       \expandafter\protected@xdef\csname glo@##1@sort\endcsname{%

```

```

259         \expandafter\glssortnumberfmt
260         {\csname glossary@\@glo@type @sortcount\endcsname}}}%
261     \@glo@storeentry{##1}%
262     \fi
263 }%

```

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

```

264 \let\@glo@check@sortallowed\@gobble
265 }

```

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

```

266 \newcommand*{\@gls@setupsort@none}{%

```

Don't store entry index information.

```

267 \def\do@glo@storeentry##1{%

```

No count register required for standard sort.

```

268 \def\@gls@defsortcount##1{%

```

Don't modify sort value.

```

269 \def\@gls@defsort##1##2{%

```

```

270 \expandafter\global\expandafter\let\csname glo@##2@sort\endcsname\@glo@sort
271 }%

```

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

```

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

```

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

```

273 \renewcommand\@glo@check@sortallowed[1]{\PackageError{glossaries}
274 {Option sort=none not allowed with \string##1}%
275 {(Use sort=def instead)}}%
276 }

```

`\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.)

```

277 \newcommand*{\glsdefmain}{%
278 \if@gls@docloaded
279 \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
280 \else
281 \newglossary{main}{gls}{glo}{\glossaryname}%
282 \fi

```

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

```

283 \newcommand*{\gls@tr@set@main@toctitle}{%
284 \translatelet{\glossarytoctitle}{Glossary}%
285 }%
286 }

```

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`

```
287 \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`

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

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

```
289 \@gls@declareoption{nomain}{%
290   \let\glsdefaulttype\relax
291   \renewcommand*{\glsdefmain}{}%
292 }
```

`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.

```
293 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
294   \ifglsacronym
295     \renewcommand{\@gls@do@acronymsdef}{%
296       \DeclareAcronymList{acronym}%
297       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
298       \renewcommand*{\acronymtype}{acronym}%
299     }%
300   \else
301     \let\@gls@do@acronymsdef\relax
302   \fi
303 }
```

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

```
299   \newcommand*{\gls@tr@set@acronym@toctitle}{%
300     \translatelet{\glossarytoctitle}{Acronyms}%
301   }%
302   }%
303 \else
304   \let\@gls@do@acronymsdef\relax
305 \fi
306 }
```

`\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.

```
307 \AtBeginDocument{%
308   \ifglsacronym
309     \ifbool{glscompatible-3.07}{%
310       {}%
311     }%
312     \providecommand*{\printacronyms}[1][ ]{%
```

```

313      \printglossary[type=\acronymtype,#1]]}%
314    }%
315  \fi
316}

@do@acronymsdef  Set default value
317 \newcommand*{\@gls@do@acronymsdef}{%

acronyms  Provide a synonym for acronym=true that can be passed via the document class options.
318 \@gls@declareoption{acronyms}{%
319   \glsacronymtrue
320   \renewcommand{\@gls@do@acronymsdef}{%
321     \DeclareAcronymList{acronym}%
322     \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
323     \renewcommand*{\acronymtype}{acronym}%

    Define hook to set the toc title when translator is in use.
324     \newcommand*{\@gls@tr@set@acronym@toctitle}{%
325       \translatelet{\glossarytoctitle}{Acronyms}%
326     }%
327   }%
328}

glsacronymlists  Comma-separated list of glossary labels indicating which glossaries contain acronyms. Note
                  that \SetAcronymStyle must be used after adding labels to this macro.
329 \newcommand*{\@glsacronymlists}{%

dtoacronymlists
330 \newcommand*{\@addtoacronymlists}[1]{%
331   \ifx\@glsacronymlists\@empty
332     \protected@xdef\@glsacronymlists{#1}%
333   \else
334     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
335   \fi
336}

lareAcronymList  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.)
337 \newcommand*{\DeclareAcronymList}[1]{%
338   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
339}

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

Determines if the glossary with the given label has been identified as being a list of acronyms.
340 \newcommand{\glsIfListOfAcronyms}[1]{%

```



```

341 \edef\@do@glis@islistofacronyms{%
342   \noexpand\@glis@islistofacronyms{#1}{\@glisacronymlists}}%
343 \do@glis@islistofacronyms
344 }

```

Internal command requires label and list to be expanded:

```

345 \newcommand{\@glis@islistofacronyms}[4]{%
346   \def\glis@islistofacronyms##1,#1,##2\end@glis@islistofacronyms{%
347     \def\@before{##1}\def\@after{##2}}%
348   \glis@islistofacronyms,#2,#1,\@nil\end@glis@islistofacronyms
349   \ifx\@after\@nnil

```

Not found

```

350   #4%
351   \else

```

Found

```

352   #3%
353   \fi
354 }

```

`lsisacronymlist` Convenient boolean.

```

355 \newif\if@glisacronymlist

```

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

```

356 \newcommand*{\@glis@ckisacronymlist}[1]{%
357   \glisIfListOfAcronyms{#1}%
358   {\@glisacronymlisttrue}{\@glisacronymlistfalse}%
359 }

```

`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.)

```

360 \newcommand*{\SetAcronymLists}[1]{%
361   \renewcommand*{\@glisacronymlists}{#1}%
362 }

```

`acronymlists`

```

363 \define@key{glossaries.sty}{acronymlists}{%
364   \DeclareAcronymList{#1}%
365 }

```

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`

```

366 \newcommand{\glscounter}{page}

```

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

```

367 \define@key{glossaries.sty}{counter}{%
368   \renewcommand*{\glscounter}{#1}%
369 }

```

gls@nohyperlist

```

370 \newcommand*{\@gls@nohyperlist}{}

```

lareNoHyperList

```

371 \newcommand*{\GlsDeclareNoHyperList}[1]{%
372   \ifdefempty\@gls@nohyperlist
373   {%
374     \renewcommand*{\@gls@nohyperlist}{#1}%
375   }%
376   {%
377     \appto\@gls@nohyperlist{,#1}%
378   }%
379 }

```

nohypertypes

```

380 \define@key{glossaries.sty}{nohypertypes}{%
381   \GlsDeclareNoHyperList{#1}%
382 }

```

ossariesWarning Prints a warning message.

```

383 \newcommand*{\GlossariesWarning}[1]{%
384   \PackageWarning{glossaries}{#1}%
385 }

```

esWarningNoLine Prints a warning message without the line number.

```

386 \newcommand*{\GlossariesWarningNoLine}[1]{%
387   \PackageWarningNoLine{glossaries}{#1}%
388 }

```

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

```

389 \newcommand{\glosortentrieswarning}{%
390   \typeout{Using TeX to sort glossary entries---this may
391   take a while}%
392 }

```

nowarn Define package option to suppress warnings

```

393 \@gls@declareoption{nowarn}{%
394   \if@gls@debug
395     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
396   \else
397     \renewcommand*{\GlossariesWarning}[1]{}%
398     \renewcommand*{\GlossariesWarningNoLine}[1]{}%

```

```

399 \renewcommand*\glosortentrieswarning}{}%
400 \fi
401 }

```

nonglossdefined Issue a warning if overriding \printglossary

```

402 \newcommand*\@gls@warnonglossdefined}{%
403 \GlossariesWarning{Overriding \string\printglossary}%
404 }

```

theglossdefined Issue a warning if overriding theglossary

```

405 \newcommand*\@gls@warnontheglossdefined}{%
406 \GlossariesWarning{Overriding 'theglossary' environment}%
407 }

```

noredefwarn Suppress warning on redefinition of \printglossary

```

408 \@gls@declareoption{noredefwarn}{%
409 \renewcommand*\@gls@warnonglossdefined}{}%
410 \renewcommand*\@gls@warnontheglossdefined}{}%
411 }

```

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

```

412 \newcommand*\@gls@sanitizedesc}{%
413 }

```

lssetexpandfield \glssetexpandfield{<field>}

Sets field to always expand.

```

414 \newcommand*\glssetexpandfield[1]{%
415 \csdef{gls@assign@#1@field}##1##2{%
416 \@@gls@expand@field{##1}{#1}{##2}%
417 }%
418 }

```

setnoexpandfield \glssetnoexpandfield{<field>}

Sets field to never expand.

```

419 \newcommand*\glssetnoexpandfield[1]{%
420 \csdef{gls@assign@#1@field}##1##2{%
421 \@@gls@noexpand@field{##1}{#1}{##2}%
422 }%
423 }

```

sign@type@field The type must always be expandable.  
424 \glssetexpandfield{type}

sign@desc@field The description is not expanded by default:  
425 \glssetnoexpandfield{desc}

descplural@field  
426 \glssetnoexpandfield{descplural}

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

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

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

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

bolplural@field  
431 \glssetnoexpandfield{symbolplural}

Sanitizing stuff:

ls@sanitizesort  
432 \newcommand\*{\@gls@sanitizesort}{%  
433 \ifglssanitizesort  
434 \@gls@sanitizesort  
435 \else  
436 \@gls@nosanitizesort  
437 \fi  
438 }

ls@sanitizesort  
439 \newcommand\*\@gls@sanitizesort{%  
440 \@onelevel@sanitize\@glo@sort  
441 }

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

dx@sanitizesort Remove braces around first character (if present) before sanitizing.  
443 \newcommand\*\@gls@noidx@sanitizesort{%  
444 \ifdefvoid\@glo@sort  
445 {}%  
446 {%

```

447 \expandafter\@@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
448 }%
449 }
450 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
451 \def\@glo@sort{#1#2}%
452 \@onelevel@sanitize\@glo@sort
453 }

```

@nosanitizesort

```

454 \newcommand*\@@gls@noidx@nosanitizesort{%
455 \ifdefined\@glo@sort
456 {}%
457 {%
458 \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
459 }%
460 }
461 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
462 \bgroup
463 \glsnoidxstripaccents
464 \protected@xdef\@@glo@sort{#1#2}%
465 \egroup
466 \let\@glo@sort\@@glo@sort
467 }

```

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.

```

468 \newcommand*\glsnoidxstripaccents{%
469 \let\IeC\@firstofone
470 \let\''\@firstofone
471 \let\''\@firstofone
472 \let\~\@firstofone
473 \let\""\@firstofone
474 \let\u\@firstofone
475 \let\t\@firstofone
476 \let\d\@firstofone
477 \let\r\@firstofone
478 \let\=\@firstofone
479 \let\.\@firstofone
480 \let\~\@firstofone
481 \let\v\@firstofone
482 \let\H\@firstofone
483 \let\c\@firstofone
484 \let\b\@firstofone

485 \let\a\@secondoftwo
486 \def\AE{AE}%
487 \def\ae{ae}%
488 \def\OE{OE}%

```

```

489 \def\oe{oe}%
490 \def\AA{AA}%
491 \def\aa{aa}%
492 \def\L{L}%
493 \def\l{l}%
494 \def\O{O}%
495 \def\o{o}%
496 \def\SS{SS}%
497 \def\ss{ss}%
498 \def\th{th}%

499 \def\TH{TH}%
500 \def\dh{dh}%
501 \def\DH{DH}%
502 }

```

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.

```

503 \define@boolkey[glS]{sanitize}{description}[true]{%
504   \GlossariesWarning{sanitize={description} package option deprecated}%
505   \ifglS@sanitize@description
506     \glSsetnoexpandfield{desc}%
507     \glSsetnoexpandfield{descplural}%
508   \else
509     \glSsetexpandfield{desc}%
510     \glSsetexpandfield{descplural}%
511   \fi
512 }

513 \define@boolkey[glS]{sanitize}{name}[true]{%
514   \GlossariesWarning{sanitize={name} package option deprecated}%
515   \ifglS@sanitize@name
516     \glSsetnoexpandfield{name}%
517   \else
518     \glSsetexpandfield{name}%
519   \fi
520 }

521 \define@boolkey[glS]{sanitize}{symbol}[true]{%
522   \GlossariesWarning{sanitize={symbol} package option deprecated}%
523   \ifglS@sanitize@symbol
524     \glSsetnoexpandfield{symbol}%
525     \glSsetnoexpandfield{symbolplural}%
526   \else
527     \glSsetexpandfield{symbol}%
528     \glSsetexpandfield{symbolplural}%
529   \fi
530 }

```

sanitizesort

```

531 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
532   \ifglssanitizesort
533     \glsssetnoexpandfield{sortvalue}%
534     \renewcommand*{\@gls@noidx@setsanitizesort}{%
535       \glssanitizesorttrue
536       \glsssetnoexpandfield{sortvalue}%
537     }%
538   \else
539     \glsssetexpandfield{sortvalue}%
540     \renewcommand*{\@gls@noidx@setsanitizesort}{%
541       \glssanitizesortfalse
542       \glsssetexpandfield{sortvalue}%
543     }%
544   \fi
545 }

```

Default setting:

```

546 \glssanitizesorttrue
547 \glsssetnoexpandfield{sortvalue}%

```

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

```

548 \newcommand*{\@gls@noidx@setsanitizesort}{%
549   \glssanitizesortfalse
550   \glsssetexpandfield{sortvalue}%
551 }

552 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
553   \setbool{glssanitizesort}{#1}%
554   \ifglssanitizesort
555     \glsssetnoexpandfield{sortvalue}%
556   \else
557     \glsssetexpandfield{sortvalue}%
558   \fi
559   \GlossariesWarning{sanitize={sort} package option
560     deprecated. Use sanitizesort instead}%
561 }

```

`sanitize`

```

562 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
563   \ifthenelse{\equal{#1}{none}}{%
564     {%
565       \GlossariesWarning{sanitize package option deprecated}%
566       \glsssetexpandfield{name}%
567       \glsssetexpandfield{symbol}%
568       \glsssetexpandfield{symbolplural}%
569       \glsssetexpandfield{desc}%
570       \glsssetexpandfield{descplural}%
571     }%
572   }{%
573     \setkeys[gls]{sanitize}{#1}%

```

```

574 }%
575 }

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

notranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator
577 \newcommand*\@gls@usetranslator{%
polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
polyglossia as well.
578 \@ifpackageloaded{polyglossia}%
579 {%
580 \let\glsifusetranslator\@secondoftwo
581 }%
582 {%
583 \@ifpackageloaded{babel}%
584 {%
585 \IfFileExists{translator.sty}%
586 {%
587 \RequirePackage{translator}%
588 \let\glsifusetranslator\@firstoftwo
589 }%
590 }%
591 }%
592 {}%
593 }%
594 }

dtranslatordict Checks if given translator dictionary has been loaded.
595 \newcommand{\glsifusedtranslatordict}[3]{%
596 \glsifusetranslator
597 {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
598 {#3}%
599 }

notranslate Provide a synonym for translate=false that can be passed via the document class.
600 \@gls@declareoption{notranslate}{%
601 \glstranslatefalse
602 \let\@gls@usetranslator\relax
603 \let\glsifusetranslator\@secondoftwo
604 }

translate Define translate option. If false don't set up multi-lingual support.
605 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
606 {true,false,babel}[true]%

```



```

607 {%
608   \ifcase\nr\relax
609     \glstranslatetrue
610     \renewcommand*\@gls@usetranslator{%
611       \@ifpackageloaded{polyglossia}%
612       {%
613         \let\glsifusetranslator\@secondoftwo
614       }%
615       {%
616         \@ifpackageloaded{babel}%
617         {%
618           \IfFileExists{translator.sty}%
619           {%
620             \RequirePackage{translator}%
621             \let\glsifusetranslator\@firstoftwo
622           }%
623         }%
624       }%
625     }%
626   }%
627 }%
628 \or
629   \glstranslatefalse
630   \let\@gls@usetranslator\relax
631   \let\glsifusetranslator\@secondoftwo
632 \or
633   \glstranslatetrue
634   \let\@gls@usetranslator\relax
635   \let\glsifusetranslator\@secondoftwo
636 \fi
637 }

```

Set the default value:

```

638 \glstranslatefalse
639 \let\glsifusetranslator\@secondoftwo
640 \@ifpackageloaded{translator}%
641 {%
642   \glstranslatetrue
643   \let\glsifusetranslator\@firstoftwo
644 }%
645 {%
646   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
647   {
648     \@ifpackageloaded{\gls@thissty}%
649     {%
650       \glstranslatetrue
651       \@endfortrue
652     }%
653   }%

```

```

654 }
655 }

indexonlyfirst Set whether to only index on first use.
656 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
657 \glsindexonlyfirstfalse

hyperfirst Set whether or not terms should have a hyperlink on first use.
658 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
659 \glshyperfirsttrue

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

footnote Set the long form of the acronym in footnote on first use.
661 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}
662 \ifbool{glsacrdescription}%
663 {}%
664 {%
665 \renewcommand*{\@gls@sanitizedesc}{}%
666 }%
667 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
668 }

description Allow acronyms to have a description (needs to be set using the description key in the optional
argument of \newacronym).
669 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}
670 \renewcommand*{\@gls@sanitizesymbol}{}%
671 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
672 }

smallcaps Define \newacronym to set the short form in small capitals.
673 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}
674 \renewcommand*{\@gls@sanitizesymbol}{}%
675 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
676 }

smaller Define \newacronym to set the short form using \smaller which obviously needs to be de-
fined by loading the appropriate package.
677 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{}
678 \renewcommand*{\@gls@sanitizesymbol}{}%
679 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
680 }

dua Define \newacronym to always use the long forms (i.e. don't use acronyms)
681 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{}
682 \renewcommand*{\@gls@sanitizesymbol}{}%
683 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
684 }

```

`shortcuts` Define acronym shortcuts.  
`685 \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.  
`686 \newcommand*{\glsorder}{word}`

`\@glsorder` The ordering information is written to the auxiliary file for `makeglossaries`, so ignore the auxiliary information.  
`687 \newcommand*{\@glsorder}[1]{}`

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

`\ifglxindy` Provide boolean to determine whether `xindy` or `makeindex` will be used to sort the glossaries.  
`690 \newif\ifglxindy`  
The default is `makeindex`:  
`691 \glxindyfalse`

`makeindex` Define package option to specify that `makeindex` will be used to sort the glossaries:  
`692 \@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.  
`693 \define@boolkey[gls]{xindy}{glsnumbers}[true]{}
694 \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.)  
`695 \def\@xdy@main@language{\language}%`  
Define key to set the language  
`696 \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.  
`697 \ifcsundef{inputencodingname}{%
698 \def\gls@codepage{}}{%
699 \def\gls@codepage{\inputencodingname}
700 }`  
Define a key to set the code page.  
`701 \define@key[gls]{xindy}{codepage}{\def\gls@codepage{#1}}`

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

```
702 \define@key{glossaries.sty}{xindy}[]{%  
703   \glsxindytrue  
704   \setkeys[glx]{xindy}{#1}%  
705 }
```

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

```
706 \@glx@declareoption{xindygloss}{%  
707   \glsxindytrue  
708 }
```

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

```
709 \@glx@declareoption{xindynoglsnumbers}{%  
710   \glsxindytrue  
711   \glx{xindy}{glxnumbers}{false}  
712 }
```

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

```
713 \define@boolkey{glossaries.sty}[glx]{automake}[true]{%  
714   \ifglxautomake  
715     \renewcommand*{\@glx@doautomake}{%  
716       \PackageError{glossaries}{You must use  
717       \string\makeglossaries\space with automake=true}  
718       {%  
719         Either remove the automake=true setting or  
720         add \string\makeglossaries\space to your document preamble.%  
721       }%  
722     }%  
723   \else  
724     \renewcommand*{\@glx@doautomake}{}%  
725   \fi  
726 }  
727 \glxautomakefalse
```

**@glx@doautomake**

```
728 \newcommand*{\@glx@doautomake}{}  
729 \AtEndDocument{\@glx@doautomake}
```

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

```
730 \define@boolkey{glossaries.sty}[glx]{savewrites}[true]{%  
731   \ifglssavewrites  
732     \renewcommand*{\glxwritefiles}{\@glxwritefiles}%  
733   \else  
734     \let\glxwritefiles\@empty  
735   \fi  
736 }
```

Set default:

```
737 \glssavewritesfalse
738 \let\glswritefiles\@empty
```

compatible-3.07

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

compatible-2.07

```
741 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
```

Also set 3.07 compatibility if this option is set.

```
742 \ifbool{glscompatible-2.07}%
743 {%
744 \booltrue{glscompatible-3.07}%
745 }%
746 {}%
747 }
748 \boolfalse{glscompatible-2.07}
```

symbols Create a “symbols” glossary type

```
749 \@gls@declareoption{symbols}{%
750 \let\@gls@do@symbolsdef\@gls@symbolsdef
751 }
```

Default is not to define the symbols glossary:

```
752 \newcommand*{\@gls@do@symbolsdef}{}%
```

@gls@symbolsdef

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

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

```
756 \newcommand*{\gls@tr@set@symbols@toctitle}{%
757 \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
758 }%
759 }%
```

numbers Create a “symbols” glossary type

```
760 \@gls@declareoption{numbers}{%
761 \let\@gls@do@numbersdef\@gls@numbersdef
762 }
```

Default is not to define the numbers glossary:

```
763 \newcommand*{\@gls@do@numbersdef}{}%
```

@gls@numbersdef

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

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

```
767 \newcommand*{\gls@tr@set@numbers@toctitle}{%
768   \translatelet{\glossarytoctitle}{Numbers (glossaries)}%
769 }%
770 }%
```

index Create an “index” glossary type

```
771 \@gls@declareoption{index}{%
772   \let\@gls@do@indexdef\@gls@indexdef
773 }
```

Default is not to define index glossary:

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

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

```
775 \newcommand*{\@gls@indexdef}{%
776   \newglossary[ilg]{index}{ind}{idx}{\indexname}%
777   \newcommand*{\printindex}[1][\]{\printglossary[type=index,##1]}%
778   \newcommand*{\newterm}[2][\]{%
779     \newglossaryentry{##2}%
780     {type={index},name={##2},description={\nopostdesc},##1}%
781   }%
```

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

```
782 \@for\CurrentOption :=\@declaredoptions\do{%
783   \ifx\CurrentOption\@empty
784   \else
785     \@expandtwoargs
786     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%
787     \ifin@
788     \@use@ption
789     \expandafter \let\csname ds@\CurrentOption\endcsname\@empty
790   \fi
791 \fi
792 }
```

Now process options passed to the package:

```
793 \ProcessOptionsX
```

Load backward compatibility stuff:

```
794 \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:

```
795 \disable@keys{glossaries.sty}{compatible-2.07,%
796   xindy,xindygloss,xindynoglsnumbers,makeindex,%
797   acronym,translate,notranslate,nolong,nosuper,notree,nostyles,nomain}
```

Now define `\setupglossaries`:

```

798 \newcommand*{\setupglossaries}[1]{%
799   \renewcommand*{\@gls@setacrstyle}{}%
800   \ifglsacrshortcuts
801     \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
802   \else
803     \def\@gls@setupshortcuts{%
804       \ifglsacrshortcuts
805         \DefineAcronymSynonyms
806       \fi
807     }%
808   \fi
809   \glsacrshortcutsfalse
810   \let\@gls@do@numbersdef\relax
811   \let\@gls@do@symbolssdef\relax
812   \let\@gls@do@indexdef\relax
813   \let\@gls@do@acronymsdef\relax
814   \setkeys{glossaries.sty}{#1}%
815   \@gls@setacrstyle
816   \@gls@setupshortcuts
817   \@gls@do@acronymsdef
818   \@gls@do@numbersdef
819   \@gls@do@symbolssdef
820   \@gls@do@indexdef
821 }
```

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}`).

```

822 \ifthenelse{\equal{\glscounter}{section}}{%
823 {%
824   \ifcsundef{chapter}{}%
825   {%
826     \let\@gls@old@chapter\@chapter
827     \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
828       \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}}}%
829   }%
830 }%
831 }
```

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

```

832 \newcommand*{\@gls@onlypremakeg}{}
```

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

```
833 \newcommand*{\@onlypremakeg}[1]{%
834   \ifx\@gls@onlypremakeg\@empty
835     \def\@gls@onlypremakeg{#1}%
836   \else
837     \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
838     \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
839   \fi
840 }
```

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

```
841 \newcommand*{\@disable@onlypremakeg}{%
842   \@for\@thiscs:=\@gls@onlypremakeg\do{%
843     \expandafter\@disable@premakecs\@thiscs%
844   }}
```

`\sable@premakecs` Disables the given command.

```
845 \newcommand*{\@disable@premakecs}[1]{%
846   \def#1{\PackageError{glossaries}{\string#1\space may only be
847     used before \string\makeglossaries}{You can't use
848     \string#1\space after \string\makeglossaries}}%
849 }
```

## 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`

```
850 \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`

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

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

```
852 \newcommand*{\glssettocitle}[1]{%
853   \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`  
854 `\providecommand*{\entryname}{Notation}`

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

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

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

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

`ymbolsgroupname`  
858 `\providecommand*{\glssymbolsgroupname}{Symbols}`

`umbersgroupname`  
859 `\providecommand*{\glsnumbersgroupname}{Numbers}`

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

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

`acrpluralsuffix`  
862 `\newcommand*{\glsupacrpluralsuffix}{\glstextup{\glsacrpluralsuffix}}`

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

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

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

`eGlossariesLang`  
865 `\newcommand*{\RequireGlossariesLang}[1]{%`  
866 `\@ifundefined{ver@glossaries-#1.ldf}{\input{glossaries-#1.ldf}}{}`  
867 `}`

`sGlossariesLang`  
868 `\newcommand*{\ProvidesGlossariesLang}[1]{%`  
869 `\ProvidesFile{glossaries-#1.ldf}%`  
870 `}`

ssarytocaptions Does nothing if translator hasn't been loaded.

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

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

```
872 \ifglstranslate
```

Load tracklang

```
873 \RequirePackage{tracklang}
```

Load translator if required.

```
874 \@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.)

```
875 \@ifpackageloaded{translator}
```

```
876 {%
```

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.

```
877 \ifboolexpr
```

```
878 {
```

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

```
880 and not
```

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

```
882 }
```

```
883 {%
```

```
884 \let\glsifusetranslator\@secondoftwo
```

```
885 }%
```

```
886 {%
```

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

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

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

```
890 {%
```

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

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

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

```
894 }%
```

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

```
896 }%
```

```
897 }%
```

```
898 }%
```

```
899 }%
```

```
900 {}%
```

Check for tracked languages

```
901 \AnyTrackedLanguages
```

```

902  {%
903    \ForEachTrackedDialect{\this@dialect}{%
904      \IfTrackedLanguageFileExists{\this@dialect}%
905      {glossaries-}% prefix
906      {.ldf}%
907      {%
908        \RequireGlossariesLang{\CurrentTrackedTag}%
909      }%
910      {%
911        \PackageWarningNoLine{glossaries}%
912        {No language module detected for ‘\this@dialect’.\MessageBreak
913        Language modules need to be installed separately.\MessageBreak
914        Please check on CTAN for a bundle called\MessageBreak
915        ‘glossaries-\CurrentTrackedLanguage’ or similar}%
916      }%
917    }%
918  }%
919  {}%

```

if using translator use translator interface.

```

920  \glsifusetranslator
921  {%
922    \renewcommand*{\glssettoctitle}[1]{%
923      \ifcsdef{gls@tr@set@#1@toctitle}%
924      {%
925        \csuse{gls@tr@set@#1@toctitle}%
926      }%
927      {%
928        \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
929      }%
930    }%
931    \renewcommand*{\glossaryname}{\translate{Glossary}}}%
932    \renewcommand*{\acronymname}{\translate{Acronyms}}}%
933    \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
934    \renewcommand*{\descriptionname}{%
935      \translate{Description (glossaries)}}}%
936    \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
937    \renewcommand*{\pagelistname}{%
938      \translate{Page List (glossaries)}}}%
939    \renewcommand*{\glssymbolsgroupname}{%
940      \translate{Symbols (glossaries)}}}%
941    \renewcommand*{\glsnumbersgroupname}{%
942      \translate{Numbers (glossaries)}}}%
943  }{}%
944 \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.

```

945 \DeclareRobustCommand*{\nopostdesc}{}

```

`\@nopostdesc` Suppress next description terminator.

```
946 \newcommand*{\@nopostdesc}{%
947   \let\org@glspostdescription\glspostdescription
948   \def\glspostdescription{%
949     \let\glspostdescription\org@glspostdescription}%
950 }
```

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

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

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

```
952 \newcommand{\glspar}{\par}
```

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

```
953 \newcommand{\setStyleFile}[1]{%
954   \renewcommand*{\gls@istfilebase}{#1}%
  Just in case \istfilename has been modified.
955   \ifglsxindy
956     \def\istfilename{\gls@istfilebase.xdy}
957   \else
958     \def\istfilename{\gls@istfilebase.ist}
959   \fi
960 }
```

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

```
961 \@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`

```
962 \ifglsxindy
963   \def\istfilename{\gls@istfilebase.xdy}
964 \else
965   \def\istfilename{\gls@istfilebase.ist}
966 \fi
```

`gls@istfilebase`

```
967 \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`

```
968 \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`

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

`\glsSetCompositor` Sets the compositor.

```
970 \newcommand*{\glsSetCompositor}[1]{%
971   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
972 \@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.

`\glsAlphacompositor` 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.

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

`\glsAlphaCompositor` Sets the alpha compositor.

```
974 \ifglsxindy
975   \newcommand*\glsSetAlphaCompositor[1]{%
976     \renewcommand*\@glsAlphacompositor{#1}}
977 \else
978   \newcommand*\glsSetAlphaCompositor[1]{%
979     \glsnoxindywarning\glsSetAlphaCompositor}
980 \fi
```

Can only be used before `\makeglossaries`

```
981 \@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.

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

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

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

Only has an effect when used before `\makeglossaries`

```
985 \@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.

```

986 \newcommand*{\gls@suffixFF}{}

```

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

```

987 \newcommand*{\glsSetSuffixFF}[1]{%
988   \renewcommand*{\gls@suffixFF}{#1}%
989 }

```

`\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".

```

990 \ifcsundef{hyperlink}%
991 {%
992   \newcommand*{\glsnumberformat}[1]{#1}%
993 }%
994 {%
995   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
996 }

```

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
997 \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
998 \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`

```

999 \newcommand*{\glossarypreamble}{%
1000   \csuse{@glossarypreamble@\currentglossary}%
1001 }

```

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

Code provided by Michael Pock.

```
1002 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
1003   \ifglossaryexists{#1}{%
1004     \csgdef{@glossarypreamble@#1}{#2}%
1005   }{%
1006     \GlossariesWarning{%
1007       Glossary ‘#1’ is not defined%
1008     }%
1009   }%
1010 }
```

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{}}
```

glossary postamble

```
1011 \newcommand*{\glossarypostamble}{}%
```

glossary section 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`.

```
1012 \newcommand*{\glossarysection}[2][\@gls@title]{%
1013   \def\@gls@title{#2}%
1014   \ifcsundef{phantomsection}%
1015   {%
1016     \@glossarysection{#1}{#2}%
1017   }%
1018   {%
1019     \p@glossarysection{#1}{#2}%
1020   }%

1021   \glsglossarymark{\glossarytoctitle}%
1022 }
```

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

```
1023 \ifcsundef{glossarymark}%
1024 {%
1025   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1026 }%
1027 {%
```

```

1028 \@ifclassloaded{memoir}
1029 {%
1030   \newcommand{\glsglossarymark}[1]{%
1031     \ifglsucmark
1032       \markboth{\memUChead{#1}}{\memUChead{#1}}%
1033     \else
1034       \markboth{#1}{#1}%
1035     \fi
1036   }
1037 }%
1038 {%
1039   \newcommand{\glsglossarymark}[1]{%
1040     \ifglsucmark
1041       \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1042     \else
1043       \@mkboth{#1}{#1}%
1044     \fi
1045   }
1046 }
1047 }

```

`\glossarymark` Provided for backward compatibility:

```

1048 \providecommand{\glossarymark}[1]{%
1049   \ifglsucmark
1050     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1051   \else
1052     \@mkboth{#1}{#1}%
1053   \fi
1054 }

```

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`

```

1055 \newcommand*{\setglossarysection}[1]{%
1056 \setkeys{glossaries.sty}{section=#1}}

```

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

`\glossarysection`

```

1057 \newcommand*{\@glossarysection}[2]{%
1058   \ifdefempty\@glossarysecstar
1059   {%
1060     \csname\@glossarysec\endcsname[#1]{#2}%
1061   }%
1062   {%

```



```

1063 \csname\@glossarysec\endcsname*{#2}%
1064 \@gls@toc{#1}{\@glossarysec}%
1065 }%

```

Do automatic labelling if required

```

1066 \@@glossaryseclabel
1067 }

```

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`

```

1068 \newcommand*\@p@glossarysection[2]{%
1069 \glsclearpage
1070 \phantomsection
1071 \ifdefempty\@glossarysecstar
1072 {%
1073 \csname\@glossarysec\endcsname{#2}%
1074 }%
1075 {%
1076 \@gls@toc{#1}{\@glossarysec}%
1077 \csname\@glossarysec\endcsname*{#2}%
1078 }%

```

Do automatic labelling if required

```

1079 \@@glossaryseclabel
1080 }

```

`\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.

```

1081 \newcommand*\gls@doclearpage{%
1082 \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1083 {%
1084 \ifcsundef{cleardoublepage}%
1085 {%
1086 \clearpage
1087 }%
1088 {%
1089 \ifcsdef{if@openright}%
1090 {%
1091 \if@openright
1092 \cleardoublepage
1093 \else
1094 \clearpage
1095 \fi
1096 }%
1097 {%
1098 \cleardoublepage

```

```

1099     }%
1100   }%
1101 }%
1102 {}%
1103 }

```

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

```

1104 \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`

```

1105 \newcommand*\@gls@toc[2]{%
1106   \ifglstoc
1107     \ifglsnumberline
1108       \addcontentsline{toc}{#2}{\protect\numberline{}#1}%
1109     \else
1110       \addcontentsline{toc}{#2}{#1}%
1111     \fi
1112   \fi
1113 }

```

## 1.4 Xindy

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

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

```

1114 \newcommand*\glsnnoxindywarning[1]{%
1115   \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1116 }

```

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

```

1117 \newcommand*\glsnomakeindexwarning[1]{%
1118   \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1119 }

```

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

```

1120 \ifglsxindy
1121   \edef\@xdyattributes{\string"default\string"}%
1122 \fi

```

dyattributelist    Comma-separated list of attributes.

```
1123 \ifglxsindy
1124   \edef\@xdyattributelist{%
1125 \fi
```

\@xdylocref    Define list of markup location references.

```
1126 \ifglxsindy
1127   \def\@xdylocref{}
1128 \fi
```

\@gls@ifinlist

```
1129 \newcommand*\@gls@ifinlist[4]{%
1130   \def\@do@ifinlist##1,#1,##2\end@do@ifinlist{%
1131     \def\@gls@listsuffix{##2}%
1132     \ifx\@gls@listsuffix\@empty
1133       #4%
1134     \else
1135       #3%
1136     \fi
1137   }%
1138   \@do@ifinlist,#2,#1,\end@do@ifinlist
1139 }
```

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.

```
1140 \ifglxsindy
1141   \newcommand*\@xdycounters{\@glscounter}
1142   \newcommand*\GlsAddXdyCounters[1]{%
1143     \@for\@gls@ctr:=#1\do{%
1144       \edef\@do@addcounter{%
1145         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1146         {%
1147           \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1148             \noexpand\@gls@ctr}%
1149         }%
1150       }%
1151       \@do@addcounter
1152     }
1153   }
```

Only has an effect before \writeist:

```
1154   \@onlypremakeg\GlsAddXdyCounters
1155 \else
1156   \newcommand*\GlsAddXdyCounters[1]{%
1157     \glsnoxindywarning\GlsAddXdyAttribute
1158   }
1159 \fi
```

saddxdycounters Counters must all be identified before adding attributes.

```

1160 \newcommand*\@disabled@glssaddxdycounters{%
1161   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1162     can't be used after \string\GlsAddXdyAttribute}{Move all
1163     occurrences of \string\GlsAddXdyCounters\space before the first
1164     instance of \string\GlsAddXdyAttribute}%
1165 }

```

AddXdyAttribute Adds an attribute.

```

1166 \ifglxsindy

```

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

```

1167 \newcommand*\@glssaddxdyattribute[2]{%

```

Add to xindy attribute list

```

1168   \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1169     \string"#2#1\string"}%

```

Add to xindy markup location.

```

1170   \expandafter\toks@\expandafter{\@xdylocref}%
1171   \edef\@xdylocref{\the\toks@ ^^J%
1172     (markup-locref
1173     :open \string"glstildechar n%
1174     \expandafter\string\csname glsX#2X#1\endcsname
1175     \string" ^^J
1176     :close \string"\string" ^^J
1177     :attr \string"#2#1\string")}%

```

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

```

1178   \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1179     \setentrycounter{##1}{#2}\csname #1\endcsname{##2}%
1180   }%
1181 }

```

High-level command:

```

1182 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

1183   \ifx\@xdyattributelist\@empty
1184     \edef\@xdyattributelist{#1}%
1185   \else
1186     \edef\@xdyattributelist{\@xdyattributelist,#1}%
1187   \fi

```

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

```

1188   \@for\@this@counter:=\@xdycounters\do{%
1189     \protected@edef\gls@do@addxdyattribute{%
1190       \noexpand\@glssaddxdyattribute{#1}{\@this@counter}%
1191     }
1192     \gls@do@addxdyattribute
1193   }%

```

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

```
1194 \let\GlsAddXdyCounters\@disabled@glssaddxdycounters
1195 }
```

Only has an effect before `\writeist`:

```
1196 \@onlypremake\GlsAddXdyAttribute
1197 \else
1198 \newcommand*\GlsAddXdyAttribute[1]{%
1199 \glsnnoxindywarning\GlsAddXdyAttribute}
1200 \fi
```

`\findefattributes` Add known attributes for all defined counters

```
1201 \ifglsxindy
1202 \newcommand*\@glssaddpredefinedattributes{%
1203 \GlsAddXdyAttribute{glssnumberformat}
1204 \GlsAddXdyAttribute{textrm}
1205 \GlsAddXdyAttribute{textsf}
1206 \GlsAddXdyAttribute{texttt}
1207 \GlsAddXdyAttribute{textbf}
1208 \GlsAddXdyAttribute{textmd}
1209 \GlsAddXdyAttribute{textit}
1210 \GlsAddXdyAttribute{textup}
1211 \GlsAddXdyAttribute{textsl}
1212 \GlsAddXdyAttribute{textsc}
1213 \GlsAddXdyAttribute{emph}
1214 \GlsAddXdyAttribute{glshypernumber}
1215 \GlsAddXdyAttribute{hyperrm}
1216 \GlsAddXdyAttribute{hypersf}
1217 \GlsAddXdyAttribute{hypertt}
1218 \GlsAddXdyAttribute{hyperbf}
1219 \GlsAddXdyAttribute{hypermd}
1220 \GlsAddXdyAttribute{hyperit}
1221 \GlsAddXdyAttribute{hyperup}
1222 \GlsAddXdyAttribute{hypersl}
1223 \GlsAddXdyAttribute{hypersc}
1224 \GlsAddXdyAttribute{hyperemph}

1225 \GlsAddXdyAttribute{glssignore}
1226 }
1227 \else
1228 \let\@glssaddpredefinedattributes\relax
1229 \fi
```

`\dyuseralphabets` List of additional alphabets

```
1230 \def\@xdyuseralphabets{}
```

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

```
1231 \ifglsxindy
```

```

1232 \newcommand*{\GlsAddXdyAlphabet}[2]{%
1233 \edef\xdyuseralphabets{%
1234 \xdyuseralphabets ^^J
1235 (define-alphabet "#1" (#2))}}
1236 \else
1237 \newcommand*{\GlsAddXdyAlphabet}[2]{%
1238 \glsnnoxindywarning\GlsAddXdyAlphabet}
1239 \fi

```

This code is only required for xindy:

```

1240 \ifglsxindy

```

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

```

1241 \newcommand*{\@gls\xdy@locationlist}{%
1242 roman-page-numbers,%
1243 Roman-page-numbers,%
1244 arabic-page-numbers,%
1245 alpha-page-numbers,%
1246 Alpha-page-numbers,%
1247 Appendix-page-numbers,%
1248 arabic-section-numbers%
1249 }

```

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

`\roman-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.

```

1250 \protected@edef\@gls@roman{\@roman{0}\string"
1251 \string"roman-numbers-lowercase\string" :sep \string"}}%
1252 \@onelevel@sanitize\@gls@roman
1253 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1254 :sep \string"}%
1255 \@onelevel@sanitize\@tmp
1256 \ifx\@tmp\@gls@roman
1257 \expandafter
1258 \edef\csname @gls\xdy@Lclass@roman-page-numbers\endcsname{%
1259 \string"roman-numbers-lowercase\string"%
1260 }%
1261 \else
1262 \expandafter
1263 \edef\csname @gls\xdy@Lclass@roman-page-numbers\endcsname{
1264 :sep \string"\@gls@roman\string"%
1265 }%
1266 \fi

```

`\roman-page-numbers` Upper case Roman numerals (I, II, ...).

```

1267 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1268   \string"roman-numbers-uppercase\string"%
1269 }%

ic-page-numbers  Arabic numbers (1, 2, ...).
1270 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1271   \string"arabic-numbers\string"%
1272 }%

ha-page-numbers  Lower case alphabetical (a, b, ...).
1273 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1274   \string"alpha\string"%
1275 }%

ha-page-numbers  Upper case alphabetical (A, B, ...).
1276 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1277   \string"ALPHA\string"%
1278 }%

ix-page-numbers  Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given by
\@glsAlphacompositor.
1279 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1280   \string"ALPHA\string"
1281   :sep \string"@glsAlphacompositor\string"
1282   \string"arabic-numbers\string"%
1283 }

section-numbers  Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by \glscompositor.
1284 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1285   \string"arabic-numbers\string"
1286   :sep \string"\glscompositor\string"
1287   \string"arabic-numbers\string"%
1288 }%

serlocationdefs  List of additional location definitions (separated by ^^J)
1289 \def\@xdyuserlocationdefs{}

erlocationnames  List of additional user location names
1290 \def\@xdyuserlocationnames{}

End of xindy-only block:
1291 \fi

xdycrossrefhook  Hook used after writing cross-reference class information.
1292 \ifglsxindy
1293 \newcommand\@xdycrossrefhook{}
1294 \fi

```

`\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.)

```

1295 \ifglxindy
1296   \newcommand*{\GlsAddXdyLocation}[3][{}]{%
1297     \def\@gls@tmp{#1}%
1298     \ifx\@gls@tmp\@empty
1299       \edef\@xdyuserlocationdefs{%
1300         \@xdyuserlocationdefs ^^J%
1301         (define-location-class \string"#2\string"^^J\space\space
1302         \space(:sep \string"{}\glssopenbrace\string" #3
1303         :sep \string"\glsclosebrace\string"))
1304       }%
1305     \else
1306       \edef\@xdyuserlocationdefs{%
1307         \@xdyuserlocationdefs ^^J%
1308         (define-location-class \string"#2\string"^^J\space\space
1309         \space(:sep "\glssopenbrace"
1310         #1
1311         :sep "\glsclosebrace\glssopenbrace" #3
1312         :sep "\glsclosebrace"))
1313       }%
1314     \fi

1315     \edef\@xdyuserlocationnames{%
1316       \@xdyuserlocationnames^^J\space\space\space
1317       \string"#2\string"}%
1318   }

```

Only has an effect before `\writeist`:

```

1319   \@onlypremakeg\GlsAddXdyLocation
1320 \else
1321   \newcommand*{\GlsAddXdyLocation}[2]{%
1322     \glsnnoxindywarning\GlsAddXdyLocation}
1323 \fi

```

`\locationclassorder` Define location class order

```

1324 \ifglxindy
1325   \def\@xdylocationclassorder{^^J\space\space\space
1326     \string"roman-page-numbers\string"^^J\space\space\space
1327     \string"arabic-page-numbers\string"^^J\space\space\space
1328     \string"arabic-section-numbers\string"^^J\space\space\space
1329     \string"alpha-page-numbers\string"^^J\space\space\space
1330     \string"Roman-page-numbers\string"^^J\space\space\space
1331     \string"Alpha-page-numbers\string"^^J\space\space\space
1332     \string"Appendix-page-numbers\string"
1333     \@xdyuserlocationnames^^J\space\space\space
1334     \string"see\string"
1335   }
1336 \fi

```



Change the location order.

ationClassOrder

```
1337 \ifglxindy
1338   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1339     \def\@xdylocationclassorder{#1}}
1340 \else
1341   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1342     \glsnnoxindywarning\GlsSetXdyLocationClassOrder}
1343 \fi
```

\@xdysortrules Define sort rules

```
1344 \ifglxindy
1345   \def\@xdysortrules{}
1346 \fi
```

\GlsAddSortRule Add a sort rule

```
1347 \ifglxindy
1348   \newcommand*\GlsAddSortRule[2]{%
1349     \expandafter\toks@\expandafter{\@xdysortrules}%
1350     \protected@edef\@xdysortrules{\the\toks@ ^^J
1351       (sort-rule \string"#1\string" \string"#2\string")}%
1352   }
1353 \else
1354   \newcommand*\GlsAddSortRule[2]{%
1355     \glsnnoxindywarning\GlsAddSortRule}
1356 \fi
```

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

```
1357 \ifglxindy
1358   \def\@xdyrequiredstyles{tex}
1359 \fi
```

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

```
1360 \ifglxindy
1361   \newcommand*\GlsAddXdyStyle[1]{%
1362     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1363 \else
1364   \newcommand*\GlsAddXdyStyle[1]{%
1365     \glsnnoxindywarning\GlsAddXdyStyle}
1366 \fi
```

GlsSetXdyStyles Reset the list of required styles

```
1367 \ifglxindy
1368   \newcommand*\GlsSetXdyStyles[1]{%
1369     \edef\@xdyrequiredstyles{#1}}
1370 \else
1371   \newcommand*\GlsSetXdyStyles[1]{%
1372     \glsnnoxindywarning\GlsSetXdyStyles}
1373 \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.

```
1374 \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.

```
1375 \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.

```
1376 \ifglsxindy
1377   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1378     \ifglossaryexists{#1}{%
1379       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1380     }{%
1381       \PackageError{glossaries}{Can't set language type for
1382         glossary type '#1' --- no such glossary}{%
1383         You have specified a glossary type that doesn't exist}}
1384   \else
1385     \newcommand*\GlsSetXdyLanguage[2][]{%
1386       \glsnoxywarning\GlsSetXdyLanguage}
1387 \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.

```
1388 \def\@gls@codepage#1#2{}
```

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

```
1389 \ifglsxindy
1390   \newcommand*\GlsSetXdyCodePage[1]{%
1391     \renewcommand*\@gls@codepage{#1}%
1392   }
```

Suggested by egreg:

```
1393   \AtBeginDocument{%
1394     \ifx\@gls@codepage\@empty
1395       \@ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{}%
1396     \fi
1397   }
1398 \else
1399   \newcommand*\GlsSetXdyCodePage[1]{%
1400     \glsnoxywarning\GlsSetXdyCodePage}
1401 \fi
```

`\xdylettergroups` Store letter group definitions.

```

1402 \ifglxsindy
1403   \ifglxs@xindy@glnumbers
1404     \def\xdylettergroups{(define-letter-group
1405       \string"glnumbers\string"^^J\space\space\space
1406       :prefixes (\string"0\string" \string"1\string"
1407       \string"2\string" \string"3\string" \string"4\string"
1408       \string"5\string" \string"6\string" \string"7\string"
1409       \string"8\string" \string"9\string")^^J\space\space\space
1410       :before \string"@glfirstletter\string")}
1411   \else
1412     \def\xdylettergroups{}
1413   \fi
1414 \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.

```

1415   \newcommand*\GlsAddLetterGroup[2]{%
1416     \expandafter\toks@\expandafter{\@xdylettergroups}%
1417     \protected@edef\xdylettergroups{\the\toks@^^J%
1418     (define-letter-group \string"#1\string"^^J\space\space#2)}%
1419   }%

```

## 1.5 Loops and conditionals

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

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

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

```

1420 \newcommand*\forallglossaries}[3][\@glo@types]{%
1421   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1422 }

```

`\forall acronyms`

```

1423 \newcommand*\forallacronyms}[2]{%
1424   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1425 }

```

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

```
\forall sentries[<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.

```

1426 \newcommand*{\forglentries}[3][\glsdefaulttype]{%
1427   \edef\@glo@list{\csname glolist@#1\endcsname}%
1428   \@for#2:=\@glo@list\do
1429   {%
1430     \ifdefempty{#2}{\}{#3}%
1431   }%
1432 }

```

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

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

Within `\forallglentries`, the current glossary type is given by `\@thisglo@`.

```

1433 \newcommand*{\forallglentries}[3][\@glo@types]{%
1434   \expandafter\forallglossaries\expandafter[#1]{\@thisglo@}%
1435   {%
1436     \forglentries[\@thisglo@]{#2}{#3}%
1437   }%
1438 }

```

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

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

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

```

1439 \newcommand{\ifglossaryexists}[3]{%
1440   \ifcsundef{@glo@type@#1@out}{#3}{#2}%
1441 }

```

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

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

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

`\glsdetoklabel`

```
1442 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

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

where  $\langle label \rangle$  is the entry's label.

```
1443 \newcommand{\ifglentryexists}[3]{%
1444   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1445 }
```

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

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

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

```
1446 \newcommand*{\ifglused}[3]{%
1447   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1448 }
```

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

$\backslash$ glsdoifexists 

$\backslash$ glsdoifexists $\langle label \rangle$ { $\langle code \rangle$ }

Generate an error if entry specified by  $\langle label \rangle$  doesn't exists, otherwise do  $\langle code \rangle$ .

```
1449 \newcommand{\glsdoifexists}[2]{%
1450   \ifglentryexists{#1}{#2}{%
1451     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1452       has not been defined}{You need to define a glossary entry before you
1453       can use it.}}%
1454 }
```

$\backslash$ glsdoifnoexists  $\backslash$ glsdoifnoexists $\langle label \rangle$ { $\langle code \rangle$ }

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

```
1455 \newcommand{\glsdoifnoexists}[2]{%
1456   \ifglentryexists{#1}{#2}{%
1457     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1458       been defined}{}}{#2}%
1459 }
```

$\backslash$ glsdoifexistsorwarn 

$\backslash$ glsdoifexistsorwarn $\langle label \rangle$ { $\langle code \rangle$ }

Generate a warning if entry specified by  $\langle label \rangle$  doesn't exists, otherwise do  $\langle code \rangle$ .

```
1460 \newcommand{\glsdoifexistsorwarn}[2]{%
1461   \ifglentryexists{#1}{#2}{%
1462     \GlossariesWarning{Glossary entry '\glsdetoklabel{#1}'
1463       has not been defined}%
1464   }%
1465 }
```

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

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

```

1466 \newcommand{\glsdoifexistsordo}[3]{%
1467   \ifglentryexists{#1}{#2}{%
1468     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1469     has not been defined}{You need to define a glossary entry before you
1470     can use it.}%
1471     #3%
1472   }%
1473 }
```

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

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

```

1474 \newcommand{\doifglossarynoexistsordo}[3]{%
1475   \ifglossaryexists{#1}%
1476   {%
1477     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1478     #3%
1479   }%
1480   {#2}%
1481 }
```

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

```

1482 \newcommand{\ifglshaschildren}[3]{%
1483   \glsdoifexists{#1}%
1484   {%
1485     \def\do@glshaschildren{#3}%
1486     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1487     \expandafter\forglentries\expandafter
1488     [\csname glo@\@gls@thislabel @type\endcsname]
1489     {\glo@label}%
1490     {%
1491       \letcs\glo@parent{glo@\glo@label @parent}%
1492       \ifdefequal\@gls@thislabel\glo@parent
1493       {%
1494         \def\do@glshaschildren{#2}%
1495         \@endfortrue
1496       }%
1497     }%
1498   }%
1499   \do@glshaschildren
1500 }%
1501 }
```

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

```
1502 \newcommand{\ifglshasparent}[3]{%
1503   \glsdoifexists{#1}%
1504   {%
1505     \ifcsemtyp{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1506   }%
1507 }
```

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

```
1508 \newcommand*{\ifglshasdesc}[3]{%
1509   \ifcsemtyp{glo@\glsdetoklabel{#1}@desc}%
1510   {#3}%
1511   {#2}%
1512 }
```

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

```
1513 \newcommand*{\ifglsdescsuppressed}[3]{%
1514   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1515   {#2}%
1516   {#3}%
1517 }
```

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

```
1518 \newcommand*{\ifglshassymbol}[3]{%
1519   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1520   \ifdefempty\@glo@symbol
1521   {#3}%
1522   {%
1523     \ifdefequal\@glo@symbol\@gls@default@value
1524     {#3}%
1525     {#2}%
1526   }%
1527 }
```

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

```
1528 \newcommand*{\ifglshaslong}[3]{%
1529   \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
1530   \ifdefempty\@glo@long
1531   {#3}%
1532   {%
1533     \ifdefequal\@glo@long\@gls@default@value
1534     {#3}%
1535     {#2}%
1536   }%
1537 }
```

```

\ifglshasshort \ifglshasshort{\langle label \rangle}{\langle true part \rangle}{\langle false part \rangle}
1538 \newcommand*{\ifglshasshort}[3]{%
1539   \letcs{\@glo@short}{glo@\glstdetoklabel{#1}@short}%
1540   \ifdefempty\@glo@short
1541     {#3}%
1542     {%
1543       \ifdefequal\@glo@short\@gls@default@value
1544         {#3}%
1545         {#2}%
1546     }%
1547 }

```

```

\ifglshasfield \ifglshasfield{\langle field \rangle}{\langle label \rangle}{\langle true part \rangle}{\langle false part \rangle}

```

```

1548 \newcommand*{\ifglshasfield}[4]{%
1549   \glstdoifexists{#2}%
1550   {%
1551     \letcs{\@glo@thisvalue}{glo@\glstdetoklabel{#2}@#1}%
1552     First check supplied field label is defined.
1553     \ifdef\@glo@thisvalue
1554       {%
1555         Is defined, so now check if empty.
1556         \ifdefempty\@glo@thisvalue
1557           {%
1558             Is empty, so doesn't have field set.
1559             #4%
1560           }%
1561           {%
1562             Not empty, so check if set to \@gls@default@value
1563             \ifdefequal\@glo@thisvalue\@gls@default@value
1564               {%
1565                 Value is set to the default value.
1566                 #4%
1567               }%
1568               {%
1569                 Non-empty, non-default value. Allow user to access this value through \glscurrentfieldvalue.
1570                 \let\glscurrentfieldvalue\@glo@thisvalue
1571                 #3%
1572               }%
1573             }%
1574           }%
1575         }%
1576       }%
1577     }%
1578   }%
1579 }

```



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

```
1570 \gls@fetchfield{\gls@thisfield}{#1}%
```

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

```
1571 \ifdef\gls@thisfield
1572 {%
```

Is defined, so now check if empty.

```
1573 \letcs{\glo@thisvalue}{glo\glsdetoklabel{#2}@\gls@thisfield}%
1574 \ifdefempty\glo@thisvalue
1575 {%
```

Is empty so field hasn't been set.

```
1576 #4%
1577 }%
1578 {%
```

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

```
1579 \ifdefequal\glo@thisvalue\gls@default@value
1580 {%
```

Value is set to the default value.

```
1581 #4%
1582 }%
1583 {%
```

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

```
1584 \let\glscurrentfieldvalue\glo@thisvalue
1585 #3%
1586 }%
1587 }%
1588 }%
1589 {%
```

Not defined.

```
1590 \GlossariesWarning{Unknown entry field '#1'}%
1591 #4%
1592 }%
1593 }%
1594 }%
1595 }
```

`\glscurrentfieldvalue`

```
1596 \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
1597 \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.

1598 \newcommand*\@gls@provide@newglossary{%
1599   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}}%

Only need to do this once.

1600 \let\@gls@provide@newglossary\relax
1601 }

\defglsentryfmt Allow different glossaries to have different display styles.

1602 \newcommand*\defglsentryfmt}[2][\glsdefaulttype]{%
1603   \csgdef{gls@#1@entryfmt}{#2}%
1604 }

\gls@doentryfmt
1605 \newcommand*\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}

ls@forbidtextext As a security precaution, don't allow the user to specify a 'tex' extension for any of the glossary
files. (Just in case a seriously confused novice user doesn't know what they're doing.) The
argument must be a control sequence whose replacement text is the requested extension.

1606 \newcommand*\@gls@forbidtextext}[1]{%
1607   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1608             or test {\ifdefstring{#1}{TEX}}}
1609   {%
1610     \def#1{nottex}%
1611     \PackageError{glossaries}%
1612       {Forbidden '.tex' extension replaced with '.nottex'}%
1613     {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1614       Don't use '.tex' as an extension for a temporary file.}%
1615   }%
1616   {%
1617   }%
1618 }

\gls@gobbleopt Discard optional argument.

1619 \newcommand*\gls@gobbleopt{\new@ifnextchar[\@gls@gobbleopt]}
1620 \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`

```
1621 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}
```

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

```
1622 \newcommand*{\s@newglossary}[2]{%
```

```
1623 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
```

```
1624 }
```

`\ns@newglossary` Define the unstarred version.

```
1625 \newcommand*{\ns@newglossary}[5][glg]{%
```

```
1626 \doifglossarynoexistsordo{#2}%
```

```
1627 {%
```

Check if default has been set

```
1628 \ifundef\glsdefaultttype
```

```
1629 {%
```

```
1630 \gdef\glsdefaultttype{#2}%
```

```
1631 }{}%
```

Add this to the list of glossary types:

```
1632 \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.

```
1633 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1634 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
```

```
1635 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
```

```
1636 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
```

```
1637 \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
```

```
1638 \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
```

```
1639 \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1640 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
```

```
1641 \@gls@provide@newglossary
```

```
1642 \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.

```

1643 \ifcsundef{gls@#2@entryfmt}%
1644 {%
1645     \defglsentryfmt[#2]{\glsentryfmt}%
1646 }%
1647 {}%

```

Define sort counter if required:

```

1648 \@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.)

```

1649 \@ifnextchar[{\@gls@setcounter{#2}}%
1650     {\@gls@setcounter{#2}[\glscounter]}%
1651 }%
1652 {%
1653     \gls@gobbleopt
1654 }%
1655 }

```

`\altnewglossary`

```

1656 \newcommand*{\altnewglossary}[3]{%
1657     \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1658 }

```

Only define new glossaries in the preamble:

```

1659 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1660 \@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  $\text{\LaTeX}$ , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```

1661 \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`

```

1662 \def\@gls@setcounter#1[#2]{%
1663     \expandafter\def\csname @gls@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```

1664     \ifglsxindy
1665         \GlsAddXdyCounters{#2}%
1666     \fi
1667 }

```

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

@gls@getcounter

```
1668 \newcommand*{\@gls@getcounter}[1]{%
1669 \csname @gls@#1@counter\endcsname
1670 }
```

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

```
1671 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1672 \@gls@do@acronymsdef
```

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

```
1673 \@gls@do@symbolsdef
```

```
1674 \@gls@do@numbersdef
```

```
1675 \@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.

```
1676 \newcommand*{\newignoredglossary}[1]{%
1677 \ifdefempty\@ignored@glossaries
1678 {%
1679 \edef\@ignored@glossaries{#1}%
1680 }%
1681 {%
1682 \eappto\@ignored@glossaries{,#1}%
1683 }%
1684 \csgdef{glolist@#1}{,}%
1685 \ifcsundef{gls@#1@entryfmt}%
1686 {%
1687 \defglsentryfmt[#1]{\glsentryfmt}%
1688 }%
1689 {}%
1690 \ifdefempty\@gls@nohyperlist
1691 {%
1692 \renewcommand*{\@gls@nohyperlist}{#1}%
1693 }%
1694 {%
1695 \eappto\@gls@nohyperlist{,#1}%
1696 }%
1697 }
```

`ignored@glossaries` List of ignored glossaries.

```
1698 \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.

```

1699 \newcommand*{\ifignoredglossary}[3]{%
1700   \edef\@gls@igtype{#1}%
1701   \expandafter\DTLifinlist\expandafter
1702     {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1703 }

```

## 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.

```

1704 \define@key{glossentry}{name}{%
1705 \def\@glo@name{#1}%
1706 }

```

**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`.

```

1707 \define@key{glossentry}{description}{%
1708 \def\@glo@desc{#1}%
1709 }

```

**descriptionplural**

```

1710 \define@key{glossentry}{descriptionplural}{%
1711 \def\@glo@descplural{#1}%
1712 }

```

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

```

1713 \define@key{glossentry}{sort}{%
1714 \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.

```

1715 \define@key{glossentry}{text}{%
1716 \def\@glo@text{#1}%
1717 }

```

**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.

```
1718 \define@key{glossentry}{plural}{%
1719 \def\@glo@plural{#1}%
1720 }
```

**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.

```
1721 \define@key{glossentry}{first}{%
1722 \def\@glo@first{#1}%
1723 }
```

**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.

```
1724 \define@key{glossentry}{firstplural}{%
1725 \def\@glo@firstplural{#1}%
1726 }
```

**s@default@value**

```
1727 \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).

```
1728 \define@key{glossentry}{symbol}{%
1729 \def\@glo@symbol{#1}%
1730 }
```

**symbolplural**

```
1731 \define@key{glossentry}{symbolplural}{%
1732 \def\@glo@symbolplural{#1}%
1733 }
```

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

```
1734 \define@key{glossentry}{type}{%
1735 \def\@glo@type{#1}}
```

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

```
1736 \define@key{glossentry}{counter}{%
1737 \ifcsundef{c@#1}%
```

```

1738 {%
1739   \PackageError{glossaries}%
1740   {There is no counter called ‘#1’}%
1741   {%
1742     The counter key should have the name of a valid counter
1743     as its value%
1744   }%
1745 }%
1746 {%
1747   \def\@glo@counter{#1}%
1748 }%
1749 }

```

**see** The see key specifies a list of cross-references

```

1750 \define@key{glossentry}{see}{%
1751   \gls@set@xr@key{see}{\@glo@see}{#1}%
1752 }

```

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

Assign a cross-reference key.

```

1753 \newcommand*{\gls@set@xr@key}[3]{%
1754   \renewcommand*{\gls@xr@key}{#1}%
1755   \gls@checkseeallowed
1756   \def#2{#3}%
1757   \@glo@seeautonumberlist
1758 }

```

`\gls@xr@key`

```

1759 \newcommand*{\gls@xr@key}{see}

```

`checkseeallowed`

```

1760 \newcommand*{\gls@checkseeallowed}{%
1761   \@gls@see@noindex
1762 }

```

`ed@preambleonly`

```

1763 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1764   \GlossariesWarning{glossaries}%
1765   {'\gls@xr@key' key doesn't have any effect when used in the document
1766   environment. Move the definition to the preamble
1767   after \string\makeglossaries\space
1768   or \string\makenoidxglossaries}%
1769 }

```

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

```

1770 \define@key{glossentry}{parent}{%
1771   \def\@glo@parent{#1}}

```



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

```
1772 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1773   \ifcase\nr\relax
1774     \def\@glo@prefix{\glsnonextpages}%
1775     \@gls@savenonumberlist{true}%
1776   \else
1777     \def\@glo@prefix{\glsnextpages}%
1778     \@gls@savenonumberlist{false}%
1779   \fi
1780 }
```

**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.

```
1781 \newcommand*{\@gls@savenonumberlist}[1]{}
```

**initnonumberlist**

```
1782 \newcommand*{\@gls@initnonumberlist}{}%
```

**storenonumberlist**

```
1783 \newcommand*{\@gls@storenonumberlist}[1]{}
```

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

```
1784 \newcommand*{\@gls@enablesavenonumberlist}{%
1785   \renewcommand*{\@gls@initnonumberlist}{%
1786     \undef\@glo@nonumberlist
1787   }%
1788   \renewcommand*{\@gls@savenonumberlist}[1]{%
1789     \def\@glo@nonumberlist{##1}%
1790   }%
1791   \renewcommand*{\@gls@storenonumberlist}[1]{%
1792     \ifdef\@glo@nonumberlist
1793       {%
1794         \cslet{glo@glstetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
1795       }%
1796     }%
1797   }%
1798   \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1799 }
```

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

**user1**

```
1800 \define@key{glossentry}{user1}{%
1801   \def\@glo@useri{##1}%
1802 }
```

user2

```
1803 \define@key{glossentry}{user2}{%  
1804   \def\@glo@userii{#1}%  
1805 }
```

user3

```
1806 \define@key{glossentry}{user3}{%  
1807   \def\@glo@useriii{#1}%  
1808 }
```

user4

```
1809 \define@key{glossentry}{user4}{%  
1810   \def\@glo@useriv{#1}%  
1811 }
```

user5

```
1812 \define@key{glossentry}{user5}{%  
1813   \def\@glo@userv{#1}%  
1814 }
```

user6

```
1815 \define@key{glossentry}{user6}{%  
1816   \def\@glo@uservi{#1}%  
1817 }
```

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.

```
1818 \define@key{glossentry}{short}{%  
1819   \def\@glo@short{#1}%  
1820 }
```

shortplural This key is provided for use by \newacronym.

```
1821 \define@key{glossentry}{shortplural}{%  
1822   \def\@glo@shortpl{#1}%  
1823 }
```

long This key is provided for use by \newacronym.

```
1824 \define@key{glossentry}{long}{%  
1825   \def\@glo@long{#1}%  
1826 }
```

longplural This key is provided for use by \newacronym.

```
1827 \define@key{glossentry}{longplural}{%  
1828   \def\@glo@longpl{#1}%  
1829 }
```

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

```
1830 \newcommand*{\@glsnname}{%
1831   \PackageError{glossaries}{name key required in
1832   \string\newglossaryentry\space for entry '\@glo@label'}{You
1833   haven't specified the entry name}}
```

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

```
1834 \newcommand*\@glsnodel{%
1835   \PackageError{glossaries}
1836   {%
1837     description key required in \string\newglossaryentry\space
1838     for entry '\@glo@label'%
1839   }%
1840   {%
1841     You haven't specified the entry description%
1842   }%
1843 }
```

`\glsdefaultplural` Now obsolete. Don't use.

```
1844 \newcommand*{\@glsdefaultplural}{}%
```

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

```
1845 \newcommand*{\@gls@missingnumberlist}[1]{%
1846   ??%
1847   \ifglssavenumberlist
1848     \GlossariesWarning{Missing number list for entry '#1'.
1849     Maybe makeglossaries + rerun required}%
1850   \else
1851     \PackageError{glossaries}%
1852     {Package option 'savenumberlist=true' required}%
1853     {%
1854       You must use the 'savenumberlist' package option
1855       to reference location lists.%
1856     }%
1857   \fi
1858 }
```

`\@glsdefaultsort` Define command to set default sort.

```
1859 \newcommand*{\@glsdefaultsort}{\@glo@name}
```

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

```
1860 \newcount\gls@level
```

`\@noexpand@field`

```
1861 \newcommand{\@@gls@noexpand@field}[3]{%
1862   \expandafter\global\expandafter
1863   \let\csname glo@#1@#2\endcsname#3%
1864 }
```

noexpand@fields

```
1865 \newcommand{\@gls@noexpand@fields}[4]{%
1866   \ifcsdef{gls@assign@#3@field}
1867   {%
1868     \ifdefequal{#4}{\@gls@default@value}%
1869     {%
1870       \edef\@gls@value{\expandonce{#1}}%
1871       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1872     }%
1873     {%
1874       \csuse{gls@assign@#3@field}{#2}{#4}%
1875     }%
1876   }%
1877   {%
1878     \ifdefequal{#4}{\@gls@default@value}%
1879     {%
1880       \edef\@gls@value{\expandonce{#1}}%
1881       \@gls@noexpand@field{#2}{#3}{\@gls@value}%
1882     }%
1883     {%
1884       \@gls@noexpand@field{#2}{#3}{#4}%
1885     }%
1886   }%
1887 }
```

ls@expand@field

```
1888 \newcommand{\@gls@expand@field}[3]{%
1889   \expandafter
1890   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1891 }
```

s@expand@fields

```
1892 \newcommand{\@gls@expand@fields}[4]{%
1893   \ifcsdef{gls@assign@#3@field}
1894   {%
1895     \ifdefequal{#4}{\@gls@default@value}%
1896     {%
1897       \edef\@gls@value{\expandonce{#1}}%
1898       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1899     }%
1900     {%
1901       \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
1902       {%
1903         \@gls@expand@field{#2}{#3}{#4}%
1904       }%
1905       {%
1906         \csuse{gls@assign@#3@field}{#2}{#4}%
1907       }%
1908     }%
1909   }%
1910 }
```

```

1908     }%
1909 }%
1910 {%
1911     \ifdefequal{#4}{\@gls@default@value}%
1912     {%
1913         \@gls@expand@field{#2}{#3}{#1}%
1914     }%
1915     {%
1916         \@gls@expand@field{#2}{#3}{#4}%
1917     }%
1918 }%
1919 }

```

switexpandonce

```

1920 \def\@gls@expandonce{\expandonce}
1921 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1922     \def\@gls@tmp{#1}%
1923     \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1924 }

```

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.

```
1925 \let\gls@assign@field\@gls@expand@fields
```

glsexpandfields

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

```

1926 \newcommand*{\glsexpandfields}{%
1927     \let\gls@assign@field\@gls@expand@fields
1928 }

```

snoexpandfields

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

```

1929 \newcommand*{\glsnoexpandfields}{%
1930     \let\gls@assign@field\@gls@noexpand@fields
1931 }

```

ewglossaryentry

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

```
1932 \newrobustcmd{\newglossaryentry}[2]{%
```

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

```

1933     \glsdoifnoexists{#1}%
1934     {%
1935         \gls@defglossaryentry{#1}{#2}%

```

```

1936 }%
1937 }

```

`\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.

```

1938 \newcommand*{\gls@defdocnewglossaryentry}{%
1939   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1940   \let\newglossaryentry\new@glossaryentry
1941 }

```

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

```

1942 \newrobustcmd{\provideglossaryentry}[2]{%
1943   \ifglstryexists{#1}%
1944   {%
1945     %
1946     \gls@defglossaryentry{#1}{#2}%
1947   }%
1948 }
1949 \@onlypreamble{\provideglossaryentry}

```

`\w@glossaryentry` For use in document environment.

```

1950 \newrobustcmd{\new@glossaryentry}[2]{%
1951   \ifundef\@gls@deffile
1952   {%
1953     \global\newwrite\@gls@deffile
1954     \immediate\openout\@gls@deffile=\jobname.glsdefs
1955   }%
1956   {%
1957     \ifglstryexists{#1}{%
1958       {%
1959         \gls@defglossaryentry{#1}{#2}%
1960       }%
1961       \@gls@writedef{#1}%
1962     }%
1963   }
1964   \AtBeginDocument
1965   {\@gls@enablesavenonumberlist
1966    \makeatletter
1967    \InputIfFileExists{\jobname.glsdefs}{%}{%
1968      \makeatother
1969      \gls@defdocnewglossaryentry
1970    }
1971   \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{}}

```

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

```

1972 \newcommand*{\@gls@writedef}[1]{%
1973   \immediate\write\@gls@deffile
1974   {%
1975     \string\ifglstryexists{#1}{%\glspercentchar^^J%

```

```

1976 \expandafter\@gobble\string\{\glpercentchar^^J%
1977 \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glpercentchar^^J%
1978 \expandafter\@gobble\string\{\glpercentchar%
1979 }%

```

Write key value information:

```

1980 \@for\@gls@map:=\@gls@keymap\do
1981 {%
1982 \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
1983 \ifdef\glo@value
1984 {%
1985 \@onelevel@sanitize\glo@value
1986 \immediate\write\@gls@deffile
1987 {%
1988 \expandafter\@firstoftwo\@gls@map
1989 =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
1990 \glpercentchar
1991 }%
1992 }%
1993 {}%
1994 }%

```

Provide hook:

```

1995 \glswritedefhook
1996 \immediate\write\@gls@deffile
1997 {%
1998 \glpercentchar^^J%
1999 \expandafter\@gobble\string\}\glpercentchar^^J%
2000 \expandafter\@gobble\string\}\glpercentchar%
2001 }%
2002 }

```

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

```

2003 \newcommand*{\@gls@keymap}{%
2004 {name}{name},%
2005 {sort}{sortvalue},% unescaped sort value
2006 {type}{type},%
2007 {first}{first},%
2008 {firstplural}{firstpl},%
2009 {text}{text},%
2010 {plural}{plural},%
2011 {description}{desc},%
2012 {descriptionplural}{descplural},%
2013 {symbol}{symbol},%
2014 {symbolplural}{symbolplural},%
2015 {user1}{useri},%
2016 {user2}{userii},%
2017 {user3}{useriii},%
2018 {user4}{useriv},%

```

```

2019 {user5}{userv},%
2020 {user6}{uservi},%
2021 {long}{long},%
2022 {longplural}{longpl},%
2023 {short}{short},%
2024 {shortplural}{shortpl},%
2025 {counter}{counter},%
2026 {parent}{parent}%
2027 }

```

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

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

```
2028 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
2029 \edef\@gls@thisval{#2}%
```

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

```

2030 \@for\@gls@map:=\@gls@keymap\do{%
2031 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2032 \ifdefequal{\@this@key}{\@gls@thisval}%
2033 {%

```

Found it.

```
2034 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```

2035 \@endfortrue
2036 }%
2037 {}%
2038 }%
2039 }

```

`\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.

```
2040 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```

2041 \newcommand*{\@sglsaddstoragekey}[1]{%
2042 \key@ifundefined{glossentry}{#1}%
2043 {%
2044 \expandafter\newcommand\expandafter*\expandafter
2045 {\csname gls@assign@#1@field\endcsname}[2]{%
2046 \@gls@expand@field{##1}{#1}{##2}%
2047 }%

```



```

2048 }%
2049 {}%
2050 \@glsaddstoragekey{#1}%
2051 }

```

Unstarred version doesn't override default expansion.

```

2052 \newcommand*{\@glsaddstoragekey}[3]{%

```

Check the specified key doesn't already exist.

```

2053 \key@ifundefined{glossentry}{#1}%
2054 {%

```

Set up the key.

```

2055 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2056 \appto\@gls@keymap{,{#1}{#1}}%

```

Set the default value.

```

2057 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%

```

Assignment code.

```

2058 \appto\@newglossaryentryposthook{%
2059 \letcs{\@glo@tmp}{@glo@#1}%
2060 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2061 }%

```

Define the no-link commands.

```

2062 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2063 }%
2064 {%
2065 \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2066 }%
2067 }

```

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

Allow user to add their own custom keys.

```

2068 \newcommand*{\glsaddkey}{\ifstar\@sglsaddkey\@glsaddkey}

```

Starred version switches on expansion for this key.

```

2069 \newcommand*{\@sglsaddkey}[1]{%
2070 \key@ifundefined{glossentry}{#1}%
2071 {%
2072 \expandafter\newcommand\expandafter*\expandafter
2073 {\csname gls@assign@#1@field\endcsname}[2]{%
2074 \@gls@expand@field{##1}{#1}{##2}%
2075 }%
2076 }%
2077 {}%
2078 \@glsaddkey{#1}%
2079 }

```

Unstarred version doesn't override default expansion.

```
2080 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2081 \key@ifundefined{glossentry}{#1}%
```

```
2082 {%
```

Set up the key.

```
2083 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
```

```
2084 \appto\@gls@keymap{, {#1}{#1}}%
```

Set the default value.

```
2085 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2086 \appto\@newglossaryentryposthook{%
```

```
2087 \letcs{\@glo@tmp}{@glo@#1}%
```

```
2088 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
```

```
2089 }%
```

Define the no-link commands.

```
2090 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
```

```
2091 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```
2092 \ifcsdef{@gls@user@#1@}%
```

```
2093 {%
```

```
2094 \PackageError{glossaries}%
```

```
2095 {Can't define '\string#5' as helper command
```

```
2096 '\expandafter\string\csname @gls@user@#1@\endcsname' already exists}%
```

```
2097 }%
```

```
2098 }%
```

```
2099 {%
```

```
2100 \expandafter\newcommand\expandafter*\expandafter
```

```
2101 {\csname @gls@user@#1\endcsname}[2][1]{%
```

```
2102 \new@ifnextchar[%
```

```
2103 {\csuse{@gls@user@#1@}{##1}{##2}}%
```

```
2104 {\csuse{@gls@user@#1@}{##1}{##2}[1]}}%
```

```
2105 \csdef{@gls@user@#1@}##1##2[##3]{%
```

```
2106 \@gls@field@link{##1}{##2}{#3{##2}##3}%
```

```
2107 }%
```

```
2108 \newrobustcmd*{#5}{%
```

```
2109 \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
```

```
2110 }%
```

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

```
2111 \ifcsdef{@Gls@user@#1@}%
```

```
2112 {%
```

```
2113 \PackageError{glossaries}%
```

```
2114 {Can't define '\string#6' as helper command
```

```
2115 '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
```

```

2116     {}%
2117 }%
2118 {%

2119     \expandafter\newcommand\expandafter*\expandafter
2120     {\csname @Gls@user@#1\endcsname}[2] [] {%
2121         \new@ifnextchar[%
2122             {\csuse{@Gls@user@#1@}{##1}{##2}}}%
2123             {\csuse{@Gls@user@#1@}{##1}{##2} [] }}%
2124     \csdef{@Gls@user@#1@}##1##2[##3]{%
2125         \@gls@field@link{##1}{##2}{#4{##2}##3}%
2126     }%
2127     \newrobustcmd*{#6}{%
2128         \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2129     }%

```

Finally the all caps version:

```

2130     \ifcsdef{@GLS@user@#1@}%
2131     {%
2132         \PackageError{glossaries}%
2133         {Can't define '\string#7' as helper command
2134         '\expandafter\string\csname @GLS@user@#1\endcsname' already exists}%
2135     }%
2136 }%
2137 {%

2138     \expandafter\newcommand\expandafter*\expandafter
2139     {\csname @GLS@user@#1\endcsname}[2] [] {%
2140         \new@ifnextchar[%
2141             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2142             {\csuse{@GLS@user@#1@}{##1}{##2} [] }}%
2143     \csdef{@GLS@user@#1@}##1##2[##3]{%
2144         \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2145     }%
2146     \newrobustcmd*{#7}{%
2147         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2148     }%
2149 }%
2150 {%
2151     \PackageError{glossaries}{Key '#1' already exists}{}%
2152 }%
2153 }

```

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

```

2154 \newcommand{\glsfieldxdef}[3]{%
2155     \glsdoifexists{#1}%
2156     {%

```

```

2157 \edef\@glo@label{\glstoklabel{#1}}%
2158 \ifcsdef{glo@\@glo@label @#2}%
2159 {%
2160 \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2161 }%
2162 {%
2163 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2164 }%
2165 }%
2166 }

```

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

```

2167 \newcommand{\glsfielddedf}[3]{%
2168 \glsdoifexists{#1}%
2169 {%
2170 \edef\@glo@label{\glstoklabel{#1}}%
2171 \ifcsdef{glo@\@glo@label @#2}%
2172 {%
2173 \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2174 }%
2175 {%
2176 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2177 }%
2178 }%
2179 }

```

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

```

2180 \newcommand{\glsfielddgdef}[3]{%
2181 \glsdoifexists{#1}%
2182 {%
2183 \edef\@glo@label{\glstoklabel{#1}}%
2184 \ifcsdef{glo@\@glo@label @#2}%
2185 {%
2186 \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2187 }%
2188 {%
2189 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2190 }%
2191 }%
2192 }

```

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

```

2193 \newcommand{\glsfielddef}[3]{%
2194   \glsdoifexists{#1}%
2195   {%
2196     \edef\@glo@label{\glsdetoklabel{#1}}%
2197     \ifcsdef{glo@\@glo@label @#2}%
2198     {%
2199       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2200     }%
2201     {%
2202       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2203     }%
2204   }%
2205 }

```

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

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

```

2206 \newcommand{\glsfieldfetch}[3]{%
2207   \glsdoifexists{#1}%
2208   {%
2209     \edef\@glo@label{\glsdetoklabel{#1}}%
2210     \ifcsdef{glo@\@glo@label @#2}%
2211     {%
2212       \letcs#3{glo@\@glo@label @#2}%
2213     }%
2214     {%
2215       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2216     }%
2217   }%
2218 }

```

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

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

```

2219 \newcommand{\ifglsfieldeq}[5]{%
2220   \glsdoifexists{#1}%
2221   {%
2222     \edef\@glo@label{\glsdetoklabel{#1}}%
2223     \ifcsdef{glo@\@glo@label @#2}%
2224     {%
2225       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2226     }%
2227     {%
2228       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2229     }%

```

```
2230 }%
2231 }
```

```
\ifglsfielddefeq \ifglsfielddefeq{<label>}{<field>}{<command>}{<true>}{<false>}
```

Tests if the value of the given field is equal to the replacement text of the given command.

```
2232 \newcommand{\ifglsfielddefeq}[5]{%
2233   \glsdoifexists{#1}%
2234   {%
2235     \edef\@glo@label{\glsdetoklabel{#1}}%
2236     \ifcsdef{glo@\@glo@label @#2}%
2237     {%
2238       \expandafter\ifdefstrequal
2239       \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2240     }%
2241     {%
2242       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2243     }%
2244   }%
2245 }
```

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

As above but uses \ifcsstrequal instead of \ifdefstrequal

```
2246 \newcommand{\ifglsfieldcseq}[5]{%
2247   \glsdoifexists{#1}%
2248   {%
2249     \edef\@glo@label{\glsdetoklabel{#1}}%
2250     \ifcsdef{glo@\@glo@label @#2}%
2251     {%
2252       \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2253     }%
2254     {%
2255       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2256     }%
2257   }%
2258 }
```

glswritedefhook

```
2259 \newcommand*{\glswritedefhook}{}%
```

gls@assign@desc

```
2260 \newcommand*{\gls@assign@desc}[1]{%
2261   \gls@assign@field{}{#1}{desc}{\@glo@desc}%
2262   \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2263 }
```

ewglossaryentry

```

2264 \newcommand{\longnewglossaryentry}[3]{%
2265   \glsdoifnoexists{#1}%
2266   {%
2267     \bgroup
2268     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2269     \long\def\@newglossaryentryprehook{%
2270       \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2271       \@org@newglossaryentryprehook
2272     }%
2273     \renewcommand*\@gls@assign@desc}[1]{%
2274       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
2275       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2276     }
2277     \gls@defglossaryentry{#1}{#2}%
2278   \egroup
2279 }
2280 }

```

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

```
2281 \@onlypreamble{\longnewglossaryentry}
```

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

```

2282 \newcommand{\longprovideglossaryentry}[3]{%
2283   \ifglstryexists{#1}{}%
2284   {\longnewglossaryentry{#1}{#2}{#3}}%
2285 }
2286 \@onlypreamble{\longprovideglossaryentry}

```

defglossaryentry

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

Defines a new entry without checking if it already exists.

```
2287 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of \GlsSetQuote:

```
2288 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2289 \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2290 \let\glslabel\@glo@label
```

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

```
2291 \let\@glo@name\@glsnoname
```

```
2292 \let\@glo@desc\@glsnodesc
```

```
2293 \let\@glo@descplural\@gls@default@value
```

```

2294 \let\@glo@type\@gls@default@value
2295 \let\@glo@symbol\@gls@default@value

```

```

2296 \let\@glo@symbolplural\@gls@default@value

```

```

2297 \let\@glo@text\@gls@default@value

```

```

2298 \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.)

```

2299 \let\@glo@first\@gls@default@value

```

```

2300 \let\@glo@firstplural\@gls@default@value

```

Set the default sort:

```

2301 \let\@glo@sort\@gls@default@value

```

Set the default counter:

```

2302 \let\@glo@counter\@gls@default@value

```

```

2303 \def\@glo@see{}%

```

```

2304 \def\@glo@parent{}%

```

```

2305 \def\@glo@prefix{}%

```

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

```

2306 \@gls@initnonnumberlist

```

```

2307 \def\@glo@useri{}%

```

```

2308 \def\@glo@userii{}%

```

```

2309 \def\@glo@useriii{}%

```

```

2310 \def\@glo@useriv{}%

```

```

2311 \def\@glo@userv{}%

```

```

2312 \def\@glo@uservi{}%

```

```

2313 \def\@glo@short{}%

```

```

2314 \def\@glo@shortpl{}%

```

```

2315 \def\@glo@long{}%

```

```

2316 \def\@glo@longpl{}%

```

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

```

2317 \@newglossaryentryprehook

```

Extract key-val information from third parameter:

```

2318 \setkeys{glossentry}{#2}%

```

Check there is a default glossary.

```

2319 \ifundef\glsdefaultttype

```

```

2320 {%

```

```

2321   \PackageError{glossaries}%

```

```

2322   {No default glossary type (have you used 'nomain' by mistake?)}%

```



```

2323     {If you use package option ‘nomain’ you must define
2324       a new glossary before you can define entries}%
2325   }%
2326   {}%

```

Assign type. This must be fully expandable

```

2327   \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2328   \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.

```

2329   \ifcsundef{glo@list@\@glo@type}%
2330   {%
2331     \PackageError{glossaries}%
2332     {Glossary type ‘\@glo@type’ has not been defined}%
2333     {You need to define a new glossary type, before making entries
2334       in it}%
2335   }%
2336   {%

```

Check if it's an ignored glossary

```

2337   \ifignoredglossary\@glo@type
2338   {%

```

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

```

2339     \ifx\@glo@desc\glsnodedesc
2340     \let\@glo@desc\@empty
2341     \fi
2342   }%
2343   {%
2344   }%
2345   \protected@edef\@glo@list@{\csname glo@list@\@glo@type\endcsname}%
2346   \expandafter\xdef\csname glo@list@\@glo@type\endcsname{%
2347     \@glo@list@{\@glo@label},}%
2348   }%

```

Initialise level to 0.

```

2349   \gls@level=0\relax

```

Has this entry been assigned a parent?

```

2350   \ifx\@glo@parent\@empty

```

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

```

2351     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2352   \else

```

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

```

2353     \ifdefequal\@glo@label\@glo@parent%
2354     {%
2355       \PackageError{glossaries}{Entry ‘\@glo@label’ can’t be its own parent}{}%
2356       \def\@glo@parent{}%
2357       \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%

```

```

2358 }%
2359 {%

Check the parent exists:
2360 \ifglentryexists{\@glo@parent}%
2361 {%

Parent exists. Set \glo@<label>@parent.
2362 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2363 \@glo@parent}%

Determine level.
2364 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2365 \advance\gls@level by 1\relax

If name hasn't been specified, use same as the parent name
2366 \ifx\@glo@name\@gls@name
2367 \expandafter\let\expandafter\@glo@name
2368 \csname glo@\@glo@parent @name\endcsname

If name and plural haven't been specified, use same as the parent
2369 \ifx\@glo@plural\@gls@default@value
2370 \expandafter\let\expandafter\@glo@plural
2371 \csname glo@\@glo@parent @plural\endcsname
2372 \fi
2373 \fi
2374 }%
2375 {%

Parent doesn't exist, so issue an error message and change this entry to have no parent
2376 \PackageError{glossaries}%
2377 {%
2378 Invalid parent '\@glo@parent'
2379 for entry '\@glo@label' - parent doesn't exist%
2380 }%
2381 {%
2382 Parent entries must be defined before their children%
2383 }%
2384 \def\@glo@parent{}%
2385 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2386 }%
2387 }%
2388 \fi

Set the level for this entry
2389 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

Define commands associated with this entry:
2390 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2391 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2392 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2393 \expandafter\gls@assign@field\expandafter

```

```

2394     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2395     {\@glo@label}{plural}{\@glo@plural}%
2396 \expandafter\gls@assign@field\expandafter
2397     {\csname glo@\@glo@label @text\endcsname}%
2398     {\@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.

```

2399 \ifx\@glo@first\@gls@default@value
2400   \expandafter\gls@assign@field\expandafter
2401     {\csname glo@\@glo@label @plural\endcsname}%
2402     {\@glo@label}{firstpl}{\@glo@firstplural}%
2403 \else
2404   \expandafter\gls@assign@field\expandafter
2405     {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2406     {\@glo@label}{firstpl}{\@glo@firstplural}%
2407 \fi

2408 \ifcsundef{@glotype@\@glo@type @counter}%
2409 {%
2410   \def\@glo@defaultcounter{\@glscounter}%
2411 }%
2412 {%
2413   \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
2414 }%
2415 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2416 \gls@assign@field{\@glo@label}{useri}{\@glo@useri}%
2417 \gls@assign@field{\@glo@label}{userii}{\@glo@userii}%
2418 \gls@assign@field{\@glo@label}{useriii}{\@glo@useriii}%
2419 \gls@assign@field{\@glo@label}{useriv}{\@glo@useriv}%
2420 \gls@assign@field{\@glo@label}{userv}{\@glo@userv}%
2421 \gls@assign@field{\@glo@label}{uservi}{\@glo@uservi}%
2422 \gls@assign@field{\@glo@label}{short}{\@glo@short}%
2423 \gls@assign@field{\@glo@label}{shortpl}{\@glo@shortpl}%
2424 \gls@assign@field{\@glo@label}{long}{\@glo@long}%
2425 \gls@assign@field{\@glo@label}{longpl}{\@glo@longpl}%
2426 \ifx\@glo@name\@glsnoname
2427   \@glsnoname
2428   \let\@gloname\@gls@default@value
2429 \fi
2430 \gls@assign@field{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2431 \ifcsundef{glo@\@glo@label @numberlist}%
2432 {%
2433   \csxdef{glo@\@glo@label @numberlist}{%
2434     \noexpand\@gls@missingnumberlist{\@glo@label}}%
2435 }%
2436 {}%

```

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

2437 \@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.

```
2438 \def\@glo@desc{\@glo@first}%
2439 \ifx\@glo@desc\@glo@desc
2440   \let\@glo@desc\@glo@first
2441 \fi
2442 \ifx\@glo@desc\@glsnodesc
2443   \@glsnodesc
2444   \let\@glo@desc\@gls@default@value
2445 \fi
2446 \gls@assign@desc{\@glo@label}%
```

Set the sort key for this entry:

```
2447 \@gls@defsort{\@glo@type}{\@glo@label}%

2448 \def\@glo@symbol{\@glo@text}%
2449 \ifx\@glo@symbol\@glo@symbol
2450   \let\@glo@symbol\@glo@text
2451 \fi
2452 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2453 \expandafter
2454 \gls@assign@field\expandafter
2455   {\csname glo@\@glo@label @symbol\endcsname}
2456   {\@glo@label}{symbolplural}{\@glo@symbolplural}%
```

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

```
2457 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2458   \noexpand\global
2459   \noexpand\let\expandafter\noexpand
2460   \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2461 }%
2462 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2463   \noexpand\global
2464   \noexpand\let\expandafter\noexpand
2465   \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2466 }%
2467 \csname glo@\@glo@label @flagfalse\endcsname
```

Sort out any cross-referencing if required.

2468 \@glo@autosee

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

```
2469 \ifignoredglossary\@glo@type
2470 {%
2471   \csdef{glo@\@glo@label @index}{}%
2472 }
2473 {%
2474   \do@glo@storeentry{\@glo@label}%
2475 }%
```

Define entry counters if enabled:

```
2476 \newglossaryentry@defcounters
```

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

```
2477 \newglossaryentryposthook
```

```
2478 }
```

`\@glo@autosee` Automatically implement `\glssee`.

```
2479 \newcommand*{\@glo@autosee}{%
```

```
2480 \ifdefined\@glo@see{}%
```

```
2481 {%
```

```
2482 \protected@edef\@do@glsee{%
```

```
2483 \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
```

```
2484 \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%
```

```
2485 \@do@glsee
```

```
2486 }%
```

```
2487 \@glo@autoseehook
```

```
2488 }%
```

`glo@autoseehook`

```
2489 \newcommand*{\@glo@autoseehook}{}
```

`aryentryprehook` Allow extra information to be added to glossary entries:

```
2490 \newcommand*{\@newglossaryentryprehook}{}
```

`ryentryposthook` Allow extra information to be added to glossary entries:

```
2491 \newcommand*{\@newglossaryentryposthook}{}
```

`try@defcounters`

```
2492 \newcommand*{\@newglossaryentry@defcounters}{}
```

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

```
2493 \newcommand*{\glsmoveentry}[2]{%
```

```
2494 \edef\@glo@thislabel{\glstoklabel{#1}}%
```

```
2495 \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
```

```
2496 \def\glo@list{,}%
```

```
2497 \forglsentries[\glo@type]{\glo@label}%
```

```
2498 {%
```

```
2499 \ifdefequal\@glo@thislabel\glo@label
```

```
2500 {}{\eappto\glo@list{\glo@label,}}%
```

```
2501 }%
```

```
2502 \cslet\glolist@\glo@type{\glo@list}%
```

```
2503 \csdef{glo@\@glo@thislabel @type}{#2}%
```

```
2504 }
```

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

```
2505 \ifglxindy
2506   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2507 \else
2508   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2509 \fi
```

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

```
2510 \ifglxindy
2511   \newcommand*{\@glossarysubentryfield}{%
2512     \string\subglossentry}
2513 \else
2514   \newcommand*{\@glossarysubentryfield}{%
2515     \string\subglossentry}
2516 \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.)

```
2517 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2518   \edef\@glo@esclabel{#1}%
2519   \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2520   \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2521   \@gls@checkmkidxchars\@glo@sort
```

Same again for the name string. Escape any special characters in the prefix

```
2522   \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2523   \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2524   \ifglxindy
```

Store using xindy syntax.

```
2525     \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2526       \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2527         (\string\@glo@sort\string" %
```

```

2528     \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2529     }%
2530     \else
      Entry has a parent
2531     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2532     \csname glo@\@glo@parent @index\endcsname
2533     (\string"\@glo@sort\string" %
2534     \string"\@glo@prefix\@glossarysubentryfield
2535     {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2536     }%
2537     \fi
2538     \else
      Store using makeindex syntax.
2539     \ifx\@glo@parent\@empty
      Sanitize \@glo@prefix
2540     \@onelevel@sanitize\@glo@prefix
      Entry doesn't have a parent
2541     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2542     \@glo@sort\@gls@actualchar\@glo@prefix
2543     \@glossaryentryfield{\@glo@esclabel}%
2544     }%
2545     \else
      Entry has a parent
2546     \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2547     \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2548     \@glo@sort\@gls@actualchar\@glo@prefix
2549     \@glossarysubentryfield
2550     {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2551     }%
2552     \fi
2553     \fi
2554 }

```

## 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`

```

2555 \AtBeginDocument{%
2556   \@ifpackageloaded{amsmath}%
2557   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2558   }%

```

```

2559 }
2560 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2561   \ifmeasuring@
2562   \else
2563     #1%
2564   \fi
2565 }
2566 \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.

```

2567 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2568   \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2569 }
2570 \newcommand*\glspatchtabularx{%
2571   \ifdef\TX@trial
2572   {%
2573     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2574     \let\glspatchtabularx\relax
2575   }%
2576   {%
2577 }

```

`\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.

```

2578 \newcommand*{\glsreset}[1]{%
2579   \gls@ifnotmeasuring
2580   {%
2581     \glsdoifexists{#1}%
2582     {%
2583       \@glsreset{#1}%
2584     }%
2585   }%
2586 }

```

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

```

2587 \newcommand*{\glslocalreset}[1]{%
2588   \gls@ifnotmeasuring
2589   {%
2590     \glsdoifexists{#1}%
2591     {%
2592       \@glslocalreset{#1}%
2593     }%
2594   }%
2595 }

```

`\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.



```

2596 \newcommand*{\glsunset}[1]{%
2597   \gls@ifnotmeasuring
2598   {%
2599     \glsdoifexists{#1}%
2600     {%
2601       \@glsunset{#1}%
2602     }%
2603   }%
2604 }

```

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

```

2605 \newcommand*{\glslocalunset}[1]{%
2606   \gls@ifnotmeasuring
2607   {%
2608     \glsdoifexists{#1}%
2609     {%
2610       \@glslocalunset{#1}%
2611     }%
2612   }%
2613 }

```

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

```

2614 \newcommand*{\@glslocalunset}{\@@glslocalunset}

```

`\@@glslocalunset` Local unset without checks.

```

2615 \newcommand*{\@@glslocalunset}[1]{%
2616   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2617 }

```

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

```

2618 \newcommand*{\@glsunset}{\@@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2619 \newcommand*{\@@glsunset}[1]{%
2620   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2621 }

```

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

```

2622 \newcommand*{\@glslocalreset}{\@@glslocalreset}

```

`\@@glslocalreset` Local reset without checks.

```

2623 \newcommand*{\@@glslocalreset}[1]{%
2624   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2625 }

```

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

```

2626 \newcommand*{\@glsreset}{\@@glsreset}

```

`\@@glsreset` Global reset without checks.

```
2627 \newcommand*{\@@glsreset}[1]{%
2628   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2629 }
```

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

`\glsresetall`

```
2630 \newcommand*{\glsresetall}[1][\@glo@types]{%
2631   \forallglsentries[#1]{\@glsentry}%
2632   {%
2633     \glsreset{\@glsentry}%
2634   }%
2635 }
```

As above, but with only a local effect:

`lslocalresetall`

```
2636 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2637   \forallglsentries[#1]{\@glsentry}%
2638   {%
2639     \glslocalreset{\@glsentry}%
2640   }%
2641 }
```

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

`\glsunsetall`

```
2642 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2643   \forallglsentries[#1]{\@glsentry}%
2644   {%
2645     \glsunset{\@glsentry}%
2646   }%
2647 }
```

As above, but with only a local effect:

`lslocalunsetall`

```
2648 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2649   \forallglsentries[#1]{\@glsentry}%
2650   {%
2651     \glslocalunset{\@glsentry}%
2652   }%
2653 }
```

## 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.

`entry@defcounters` Define entry fields to keep track of how many times that entry has been marked as used.

```
2654 \newcommand*{\@newglossaryentry@defcounters}{%
2655   \csdef{glo@\@glo@label @currcount}{0}%
2656   \csdef{glo@\@glo@label @prevcount}{0}%
2657 }
```

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

```
2658 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2659 \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2660 \renewcommand*{\gls@defdocnewglossaryentry}{%
2661   \renewcommand*{\newglossaryentry}[2]{%
2662     \PackageError{glossaries}{\string\newglossaryentry\space
2663     may only be used in the preamble when entry counting has
2664     been activated}{If you use \string\glsenableentrycount\space
2665     you must place all entry definitions in the preamble not in
2666     the document environment}%
2667   }%
2668 }
```

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

```
2669 \newcommand*{\glsentrycurrcount}[1]{%
2670   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2671   {0}{\@gls@entry@field{##1}{currcount}}%
2672 }%
2673 \newcommand*{\glsentryprevcount}[1]{%
2674   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2675   {0}{\@gls@entry@field{##1}{prevcount}}%
2676 }
```

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

```
2677 \renewcommand*{\@glsunset}[1]{%
2678   \@glsunset{##1}%
2679   \@gls@increment@currcount{##1}%
2680 }
```

```

2681 \renewcommand*\@glslocalunset}[1]{%
2682   \@glslocalunset{##1}%
2683   \@gls@local@increment@currcount{##1}%
2684 }%
2685 \renewcommand*\@glsreset}[1]{%
2686   \@glsreset{##1}%
2687   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
2688 }%
2689 \renewcommand*\@glslocalreset}[1]{%
2690   \@glslocalreset{##1}%
2691   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
2692 }%

```

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.)

```

2693 \def\@cgl's@##1##2[##3]{%
2694   \ifnum\glsentryprevcount{##2}=1\relax
2695     \cgl'sformat{##2}{##3}%
2696     \glsunset{##2}%
2697   \else
2698     \@cgl's@{##1}{##2}[##3]%
2699   \fi
2700 }%

```

Similarly for the analogous commands. No case change plural:

```

2701 \def\@cgl'spl@##1##2[##3]{%
2702   \ifnum\glsentryprevcount{##2}=1\relax
2703     \cgl'splformat{##2}{##3}%
2704     \glsunset{##2}%
2705   \else
2706     \@cgl'spl@{##1}{##2}[##3]%
2707   \fi
2708 }%

```

First letter uppercase singular:

```

2709 \def\@cGls@##1##2[##3]{%
2710   \ifnum\glsentryprevcount{##2}=1\relax
2711     \cGlsformat{##2}{##3}%
2712     \glsunset{##2}%
2713   \else
2714     \@cGls@{##1}{##2}[##3]%
2715   \fi
2716 }%

```

First letter uppercase plural:

```

2717 \def\@cGlspl@##1##2[##3]{%
2718   \ifnum\glsentryprevcount{##2}=1\relax
2719     \cGlsplformat{##2}{##3}%
2720     \glsunset{##2}%
2721   \else
2722     \@cGlspl@{##1}{##2}[##3]%

```

```

2723 \fi
2724 }%

```

Write information to aux file at the end of the document

```

2725 \AtEndDocument{\@gls@write@entrycounts}%

```

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

```

2726 \renewcommand*{\@gls@entry@count}[2]{%
2727   \csxdef{glo@glstoklabel{##1}@prevcount}{##2}%
2728 }%

```

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

```

2729 \let\glsenableentrycount\relax
2730 }
2731 \@onlypreamble\glsenableentrycount

```

`ement@currcount`

```

2732 \newcommand*{\@gls@increment@currcount}[1]{%
2733   \csxdef{glo@glstoklabel{##1}@currcount}{%
2734     \number\numexpr\glsentrycurrcount{##1}+1}%
2735 }

```

`ement@currcount`

```

2736 \newcommand*{\@gls@local@increment@currcount}[1]{%
2737   \csxdef{glo@glstoklabel{##1}@currcount}{%
2738     \number\numexpr\glsentrycurrcount{##1}+1}%
2739 }

```

`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.)

```

2740 \newcommand*{\@gls@write@entrycounts}{%
2741   \immediate\write\@auxout
2742     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2743   \forallglsentries{\@glsentry}{%
2744     \ifglsused{\@glsentry}%
2745     {\immediate\write\@auxout
2746       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2747     }%
2748   }%
2749 }

```

`gls@entry@count`

Default behaviour is to ignore arguments. Activated by `\glsenableentrycount`.

```

2750 \newcommand*{\@gls@entry@count}[2]{%

```

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

```

2751 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}

```

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

```
2752 \newcommand*{\@cgl}{2} [] {%
2753   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2} []}%
2754 }
```

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

```
2755 \def\@cgl@#1#2[#3]{%
2756   \GlossariesWarning{\string\cgl\space is defaulting to
2757     \string\gl\space since you haven't enabled entry counting}%
2758   \@gl@{#1}{#2}[#3]%
2759 }
```

`\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.

```
2760 \newcommand*{\cglformat}{2} {%
2761   \ifglshaslong{#1}{\glentrylong{#1}}{\glentryfirst{#1}}#2%
2762 }
```

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

```
2763 \newrobustcmd*{\cGls}{\@gl@hyp@opt\@cGls}
```

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

```
2764 \newcommand*{\@cGls}{2} [] {%
2765   \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2} []}%
2766 }
```

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

```
2767 \def\@cGls@#1#2[#3]{%
2768   \GlossariesWarning{\string\cGls\space is defaulting to
2769     \string\Gls\space since you haven't enabled entry counting}%
2770   \@Gls@{#1}{#2}[#3]%
2771 }
```

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

```
2772 \newcommand*{\cGlsformat}{2} {%
2773   \ifglshaslong{#1}{\Glentrylong{#1}}{\Glentryfirst{#1}}#2%
2774 }
```

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

```
2775 \newrobustcmd*{\cglspl}{\@gl@hyp@opt\@cglspl}
```

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

```
2776 \newcommand*{\@cglsp1}[2] [] {%
2777   \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2} []}%
2778 }
```

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

```
2779 \def\@cglsp1@#1#2[#3] {%
2780   \GlossariesWarning{\string\cglsp1\space is defaulting to
2781     \string\glsp1\space since you haven't enabled entry counting}%
2782   \@glsp1@{#1}{#2}[#3]%
2783 }
```

`\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.

```
2784 \newcommand*{\cglsp1format}[2] {%
2785   \ifglshaslong{#1}{\gl Sentrylongpl{#1}}{\gl Sentryfirstplural{#1}}#2%
2786 }
```

`\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.)

```
2787 \newrobustcmd*{\cGlsp1}{\@glshyp@opt\@cGlsp1}
```

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

```
2788 \newcommand*{\@cGlsp1}[2] [] {%
2789   \new@ifnextchar[{\@cGlsp1@{#1}{#2}}{\@cGlsp1@{#1}{#2} []}%
2790 }
```

`\@cGlsp1@` Read in the final optional argument. This defaults to same behaviour as `\Glsp1` but issues a warning.

```
2791 \def\@cGlsp1@#1#2[#3] {%
2792   \GlossariesWarning{\string\cGlsp1\space is defaulting to
2793     \string\Glsp1\space since you haven't enabled entry counting}%
2794   \@Glsp1@{#1}{#2}[#3]%
2795 }
```

`\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.

```
2796 \newcommand*{\cGlsp1format}[2] {%
2797   \ifglshaslong{#1}{\Gl Sentrylongpl{#1}}{\Gl Sentryfirstplural{#1}}#2%
2798 }
```

## 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.<sup>1</sup>

---

<sup>1</sup>and any other valid  $\LaTeX$  code that can be used in the preamble.

`\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`

```
2799 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2800   \let\@gls@default\glsdefaulttype
2801   \def\glsdefaulttype{#1}\input{#2}%
2802   \let\glsdefaulttype\@gls@default
2803 }
```

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

```
2804 \@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`

```
2805 \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 ensure backward compatibility, the default use the old `\glsdisplay` and `\glsdisplayfirst` style of commands

```
2806 \newcommand*{\glsentryfmt}{%
2807   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2808 }
```

Format that provides backwards compatibility:

```
2809 \newcommand*{\@gls@default@entryfmt}[2]{%
2810   \ifdefempty\glscustomtext
2811     {%
2812       \glsifplural
2813       {%
```

Plural form

```
2814       \glscapscase
2815     {%
```



Don't adjust case

```
2816      \ifglused\glslabel
2817      {%
```

Subsequent use

```
2818      #2{\glsentryplural{\glslabel}}%
2819      {\glsentrydescplural{\glslabel}}%
2820      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2821      }%
2822      {%
```

First use

```
2823      #1{\glsentryfirstplural{\glslabel}}%
2824      {\glsentrydescplural{\glslabel}}%
2825      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2826      }%
2827      }%
2828      {%
```

Make first letter upper case

```
2829      \ifglused\glslabel
2830      {%
```

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.)

```
2831      \ifbool{glscompatible-3.07}%
2832      {%
2833      \protected@edef\@glo@etext{%
2834      #2{\glsentryplural{\glslabel}}%
2835      {\glsentrydescplural{\glslabel}}%
2836      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2837      \xmakefirstuc\@glo@etext
2838      }%
2839      {%
2840      #2{\Glsentryplural{\glslabel}}%
2841      {\glsentrydescplural{\glslabel}}%
2842      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2843      }%
2844      }%
2845      {%
```

First use

```
2846      \ifbool{glscompatible-3.07}%
2847      {%
2848      \protected@edef\@glo@etext{%
2849      #1{\glsentryfirstplural{\glslabel}}%
2850      {\glsentrydescplural{\glslabel}}%
2851      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2852      \xmakefirstuc\@glo@etext
2853      }%
```

```

2854         {%
2855         #1{\Glsentryfirstplural{\glslabel}}}%
2856         {\Glsentrydescplural{\glslabel}}}%
2857         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}%
2858     }%
2859 }%
2860 }%
2861 {%

    Make all upper case
2862     \ifglscased\glslabel
2863     {%

        Subsequent use
2864         \mfirstucMakeUppercase{#2{\Glsentryplural{\glslabel}}}%
2865         {\Glsentrydescplural{\glslabel}}}%
2866         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2867     }%
2868     {%

        First use
2869         \mfirstucMakeUppercase{#1{\Glsentryfirstplural{\glslabel}}}%
2870         {\Glsentrydescplural{\glslabel}}}%
2871         {\Glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2872     }%
2873 }%
2874 }%
2875 {%

    Singular form
2876     \glscapscase
2877     {%

        Don't adjust case
2878         \ifglscased\glslabel
2879         {%

            Subsequent use
2880             #2{\Glsentrytext{\glslabel}}}%
2881             {\Glsentrydesc{\glslabel}}}%
2882             {\Glsentrysymbol{\glslabel}}{\glsinsert}%
2883         }%
2884         {%

            First use
2885             #1{\Glsentryfirst{\glslabel}}}%
2886             {\Glsentrydesc{\glslabel}}}%
2887             {\Glsentrysymbol{\glslabel}}{\glsinsert}%
2888         }%
2889         }%
2890         {%

```

### Make first letter upper case

```
2891 \ifglused\glslabel
2892 {%
```

### Subsequent use

```
2893 \ifbool{glscompatible-3.07}%
2894 {%
2895 \protected@edef\@glo@etext{%
2896 #2{\glsentrytext{\glslabel}}%
2897 {\glsentrydesc{\glslabel}}%
2898 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2899 \xmakefirstuc\@glo@etext
2900 }%
2901 {%
2902 #2{\Glsentrytext{\glslabel}}%
2903 {\glsentrydesc{\glslabel}}%
2904 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2905 }%
2906 }%
2907 {%
```

### First use

```
2908 \ifbool{glscompatible-3.07}%
2909 {%
2910 \protected@edef\@glo@etext{%
2911 #1{\glsentryfirst{\glslabel}}%
2912 {\glsentrydesc{\glslabel}}%
2913 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2914 \xmakefirstuc\@glo@etext
2915 }%
2916 {%
2917 #1{\Glsentryfirst{\glslabel}}%
2918 {\glsentrydesc{\glslabel}}%
2919 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2920 }%
2921 }%
2922 }%
2923 {%
```

### Make all upper case

```
2924 \ifglused\glslabel
2925 {%
```

### Subsequent use

```
2926 \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}%
2927 {\glsentrydesc{\glslabel}}%
2928 {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2929 }%
2930 {%
```

### First use

```

2931      \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2932      {\glsentrydesc{\glslabel}}}%
2933      {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2934    }%
2935  }%
2936 }%
2937 }%
2938 {%

```

Custom text provided in \glsdisp

```

2939    \ifglsused{\glslabel}%
2940    {%

```

Subsequent use

```

2941      #2{\glscustomtext}%
2942      {\glsentrydesc{\glslabel}}}%
2943      {\glsentrysymbol{\glslabel}}{}%
2944    }%
2945    {%

```

First use

```

2946      #1{\glscustomtext}%
2947      {\glsentrydesc{\glslabel}}}%
2948      {\glsentrysymbol{\glslabel}}{}%
2949    }%
2950  }%
2951 }

```

`\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.

```

2952 \newcommand*{\glsentryfmt}{%
2953   \ifdefempty\glscustomtext
2954   {%
2955     \glsifplural
2956     {%

```

Plural form

```

2957     \glscapscase
2958     {%

```

Don't adjust case

```

2959     \ifglsused\glslabel
2960     {%

```

Subsequent use

```

2961     \glsentryplural{\glslabel}\glsinsert
2962     }%
2963     {%

```

First use

```

2964     \glsentryfirstplural{\glslabel}\glsinsert
2965     }%

```

2966 }%  
2967 {%

#### Make first letter upper case

2968 \ifglused\glslabel  
2969 {%

#### Subsequent use.

2970 \Glentryplural{\glslabel}\glinsert  
2971 }%  
2972 {%

#### First use

2973 \Glentryfirstplural{\glslabel}\glinsert  
2974 }%  
2975 }%  
2976 {%

#### Make all upper case

2977 \ifglused\glslabel  
2978 {%

#### Subsequent use

2979 \mfirstucMakeUppercase  
2980 {\glentryplural{\glslabel}\glinsert}%  
2981 }%  
2982 {%

#### First use

2983 \mfirstucMakeUppercase  
2984 {\glentryfirstplural{\glslabel}\glinsert}%  
2985 }%  
2986 }%  
2987 }%  
2988 {%

#### Singular form

2989 \glscapscase  
2990 {%

#### Don't adjust case

2991 \ifglused\glslabel  
2992 {%

#### Subsequent use

2993 \glentrytext{\glslabel}\glinsert  
2994 }%  
2995 {%

#### First use

2996 \glentryfirst{\glslabel}\glinsert  
2997 }%  
2998 }%  
2999 {%

Make first letter upper case

```
3000      \ifglused\glslabel
3001      {%
```

Subsequent use

```
3002      \Glsentrytext{\glslabel}\glsinsert
3003      }%
3004      {%
```

First use

```
3005      \Glsentryfirst{\glslabel}\glsinsert
3006      }%
3007      }%
3008      {%
```

Make all upper case

```
3009      \ifglused\glslabel
3010      {%
```

Subsequent use

```
3011      \mfirstucMakeUppercase{\glsentrytext{\glslabel}\glsinsert}%
3012      }%
3013      {%
```

First use

```
3014      \mfirstucMakeUppercase{\glsentryfirst{\glslabel}\glsinsert}%
3015      }%
3016      }%
3017      }%
3018      }%
3019      {%
```

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

```
3020      \glscustomtext\glsinsert
3021      }%
3022 }
```

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

```
3023 \newcommand*{\glsgenacfmt}{%
3024   \ifdefempty\glscustomtext
3025   {%
3026     \ifglused\glslabel
3027     {%
```

Subsequent use:

```
3028     \glsifplural
3029     {%
```

Subsequent plural form:

```
3030     \glscapscase
3031     {%
```

Subsequent plural form, don't adjust case:

```
3032      \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
3033      }%
3034      {%
```

Subsequent plural form, make first letter upper case:

```
3035      \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
3036      }%
3037      {%
```

Subsequent plural form, all caps:

```
3038      \mfirstucMakeUppercase
3039      {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
3040      }%
3041      }%
3042      {%
```

Subsequent singular form

```
3043      \glscapscase
3044      {%
```

Subsequent singular form, don't adjust case:

```
3045      \acronymfont{\glsentryshort{\glslabel}}\glsinsert
3046      }%
3047      {%
```

Subsequent singular form, make first letter upper case:

```
3048      \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
3049      }%
3050      {%
```

Subsequent singular form, all caps:

```
3051      \mfirstucMakeUppercase
3052      {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%
3053      }%
3054      }%
3055      }%
3056      {%
```

First use:

```
3057      \glsifplural
3058      {%
```

First use plural form:

```
3059      \glscapscase
3060      {%
```

First use plural form, don't adjust case:

```
3061      \genplacrformat{\glslabel}{\glsinsert}%
3062      }%
3063      {%
```

First use plural form, make first letter upper case:

```
3064      \Genplacrfullformat{\glslabel}{\glsinsert}%
3065      }%
3066      {%
```

First use plural form, all caps:

```
3067      \mfirstucMakeUppercase
3068      {\genplacrfullformat{\glslabel}{\glsinsert}}%
3069      }%
3070      }%
3071      {%
```

First use singular form

```
3072      \glscapscase
3073      {%
```

First use singular form, don't adjust case:

```
3074      \genacrfullformat{\glslabel}{\glsinsert}%
3075      }%
3076      {%
```

First use singular form, make first letter upper case:

```
3077      \Genacrfullformat{\glslabel}{\glsinsert}%
3078      }%
3079      {%
```

First use singular form, all caps:

```
3080      \mfirstucMakeUppercase
3081      {\genacrfullformat{\glslabel}{\glsinsert}}%
3082      }%
3083      }%
3084      }%
3085      }%
3086      {%
```

User supplied text.

```
3087      \glscustomtext
3088      }%
3089 }
```

genacrfullformat

```
\genacrfullformat{\langle label \rangle}{\langle insert \rangle}
```

The full format used by \gls`genacfmt` (singular).

```
3090 \newcommand*{\genacrfullformat}[2]{%
3091   \glsentrylong{#1}#2\space
3092   (\protect\firstacronymfont{\glsentryshort{#1}})%
3093 }
```

Genacrfullformat

```
\Genacrfullformat{\langle label \rangle}{\langle insert \rangle}
```



As above but makes the first letter upper case.

```
3094 \newcommand*{\Genacrfullformat}[2]{%
3095   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3096   \xmakefirstuc\gls@text
3097 }
```

`\genplacrfullformat` `\genplacrfullformat{\label}{\insert}`

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

```
3098 \newcommand*{\genplacrfullformat}[2]{%
3099   \glsentrylongpl{#1}#2\space
3100   (\protect\firstacronymfont{\glsentryshortpl{#1}})%
3101 }
```

`\Genplacrfullformat` `\Genplacrfullformat{\label}{\insert}`

As above but makes the first letter upper case.

```
3102 \newcommand*{\Genplacrfullformat}[2]{%
3103   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3104   \xmakefirstuc\gls@text
3105 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3106 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3107 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3108 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3109   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3110   Use \string\defglsentryfmt\space instead}%
3111   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3112   \edef\@gls@doentrydef{%
3113     \noexpand\defglsentryfmt[#1]{%
3114       \noexpand\ifcsdef\gls@#1@displayfirst}%
3115     {%
3116       \noexpand\@@gls@default@entryfmt
3117       {\noexpand\csuse\gls@#1@displayfirst}}%
3118     {\noexpand\csuse\gls@#1@display}}%
3119   }%
3120   {%
3121     \noexpand\@@gls@default@entryfmt
3122     {\noexpand\glsdisplayfirst}%
3123     {\noexpand\csuse\gls@#1@display}}%
3124   }%
```

```

3125 }%
3126 }%
3127 \@gls@doentrydef
3128 }

```

`glsdisplayfirst` Deprecated. Kept for backward compatibility.

```

3129 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3130   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3131   Use \string\defglsentryfmt\space instead}%
3132   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3133   \edef\@gls@doentrydef{%
3134     \noexpand\defglsentryfmt[#1]{%
3135       \noexpand\ifcsdef{gls@#1@display}%
3136       {%
3137         \noexpand\@@gls@default@entryfmt
3138         {\noexpand\csuse{gls@#1@displayfirst}}}%
3139         {\noexpand\csuse{gls@#1@display}}}%
3140     }%
3141     {%
3142       \noexpand\@@gls@default@entryfmt
3143       {\noexpand\csuse{gls@#1@displayfirst}}}%
3144       {\noexpand\glsdisplay}%
3145     }%
3146   }%
3147 }%
3148 \@gls@doentrydef
3149 }

```

## 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 `\defentryfmt`). 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.

```

3150 \define@key{glslink}{counter}{%
3151   \ifcsundef{c@#1}%
3152   {%
3153     \PackageError{glossaries}%
3154     {There is no counter called '#1'}%
3155   }%

```

```

3156      The counter key should have the name of a valid counter
3157      as its value%
3158  }%
3159 }%
3160 {%
3161   \def\@gls@counter{#1}%
3162 }%
3163 }

```

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.

```

3164 \define@key{glslink}{format}{%
3165   \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.

```

3166 \define@boolkey{glslink}{hyper}[true]{}

```

Initialise hyper key.

```

3167 \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.

```

3168 \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.

```

3169 \newcommand*{\glslinkvar}[3]{#1}

```

`\glsifhyper` Now deprecated.

```

3170 \newcommand*{\glsifhyper}[2]{%
3171   \glslinkvar{#1}{#2}{#1}%
3172   \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3173     you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3174 }

```

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

```

3175 \newcommand*{\@gls@hyp@opt}[1]{%

```

```

3176 \let\glslinkvar\@firstofthree
3177 \let\@gls@hyp@opt@cs#1\relax
3178 \@ifstar{\s@gls@hyp@opt}%
3179 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{\#1}}%
3180 }

```

\s@gls@hyp@opt Starred version

```

3181 \newcommand*{\s@gls@hyp@opt}[1] [] {%
3182 \let\glslinkvar\@secondofthree
3183 \@gls@hyp@opt@cs[hyper=false,#1]}

```

\p@gls@hyp@opt Plus version

```

3184 \newcommand*{\p@gls@hyp@opt}[1] [] {%
3185 \let\glslinkvar\@thirdofthree
3186 \@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

```

3187 \newrobustcmd*{\glslink}{%
3188 \@gls@hyp@opt\@gls@link
3189 }

```

\@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.

```

3190 \newcommand*{\@gls@link}[3] [] {%
3191 \glsdoifexistsordo{#2}%
3192 {%
3193 \let\do@gls@link@checkfirsthyper\relax
3194 \@gls@link[#1]{#2}{#3}%
3195 }%

```

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

```

3196 \glstextformat{#3}%
3197 }%

```

```

3198 \glspostlinkhook
3199 }

glspostlinkhook
3200 \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.
3201 \newcommand*{\@gls@link@checkfirsthyper}{%
3202 \ifglsused{\glslabel}%
3203 {%
3204 }%
3205 {%
3206 \gls@checkisacronymlist\glstype
3207 \ifglshyperfirst
3208 \if@glsisacronymlist
3209 \ifglsacrfootnote
3210 \KV@glslink@hyperfalse
3211 \fi
3212 \fi
3213 \else
3214 \KV@glslink@hyperfalse
3215 \fi
3216 }%

Allow user to hook into this
3217 \glslinkcheckfirsthyperhook
3218 }

kfirsthyperhook Allow used to hook into the \@gls@link@checkfirsthyper macro
3219 \newcommand*{\glslinkcheckfirsthyperhook}{}

linkpostsetkeys
3220 \newcommand*{\glslinkpostsetkeys}{}

\glsifhyperon Check the value of the hyper key:
3221 \newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}

ablehyperinlist Disable hyperlink if in the “nohyper” list.
3222 \newcommand*{\do@glsdisablehyperinlist}{%
3223 \expandafter\DTLifinlist\expandafter{\glstype}{\@gls@nohyperlist}%
3224 {\KV@glslink@hyperfalse}}%
3225 }

lt@glslink@opts Hook to set default options for \@glslink.
3226 \newcommand*{\@gls@setdefault@glslink@opts}{}

```

`\@gls@link`

```
3227 \def\@gls@link[#1]#2#3{%
```

Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with tabularx).

```
3228 \leavevmode
```

```
3229 \edef\glslabel{\glsdetoklabel{#2}}%
```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

```
3230 \def\@gls@link@opts{#1}%
```

```
3231 \let\@gls@link@label\glslabel
```

```
3232 \def\@glsnumberformat{glsnumberformat}%
```

```
3233 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
```

If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default

```
3234 \edef\glstype{\csname glo@\glslabel @type\endcsname}%
```

Save original setting

```
3235 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Set defaults:

```
3236 \@gls@setdefault@glslink@opts
```

Switch off hyper setting if the glossary type has been identified in nohyperlist.

```
3237 \do@gl:disablehyperinlist
```

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`.

```
3238 \do@gls@link@checkfirsthyper
```

```
3239 \setkeys{glslink}{#1}%
```

Add a hook for the user to customise things after the keys have been set.

```
3240 \glslinkpostsetkeys
```

Store the entry’s counter in `\theglentrycounter`

```
3241 \@gls@saveentrycounter
```

Define sort key if necessary:

```
3242 \@gls@setsort{\glslabel}%
```

(De-tok’ing done by `\@do@wrglossary`)

```
3243 \@do@wrglossary{#2}%
```

```
3244 \ifKV@glslink@hyper
```

```
3245 \glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3246 \else
```

```
3247 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
```

```
3248 \fi
```

Restore original setting

```
3249 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
```

```
3250 }
```

```

\glolinkprefix
3251 \newcommand*{\glolinkprefix}{glo:}

glentrycounter Set default value of entry counter
3252 \def\glentrycounter{\glscounter}%

saveentrycounter Need to check if using equation counter in align environment:
3253 \newcommand*{\@gls@saveentrycounter}{%
3254   \def\@gls@Hcounter{}%

   Are we using equation counter?
3255   \ifthenelse{\equal{\@gls@counter}{equation}}{%
3256     {

       If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as
       may be inside an inner environment.)
3257     \ifcsundef{xatlevel@}%
3258     {%
3259       \edef\theglentrycounter{\expandafter\noexpand
3260         \csname the\@gls@counter\endcsname}%
3261     }%
3262     {%
3263       \ifx\xatlevel@\@empty
3264         \edef\theglentrycounter{\expandafter\noexpand
3265           \csname the\@gls@counter\endcsname}%
3266       \else
3267         \savecounters@
3268         \advance\c@equation by 1\relax
3269         \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

       Check if hyperref version of this counter
3270         \ifcsundef{theH\@gls@counter}%
3271         {%
3272           \def\@gls@Hcounter{\theglentrycounter}%
3273         }%
3274         {%
3275           \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3276         }%
3277         \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3278         \restorecounters@
3279       \fi
3280     }%
3281   }%
3282   {%

       Not using equation counter so no special measures:
3283     \edef\theglentrycounter{\expandafter\noexpand
3284       \csname the\@gls@counter\endcsname}%
3285   }%

```

Check if hyperref version of this counter

```

3286 \ifx\@gls@Hcounter\@empty
3287 \ifcsundef{theH\@gls@counter}%
3288 {%
3289 \def\theHglentrycounter{\theglentrycounter}%
3290 }%
3291 {%
3292 \protected@edef\theHglentrycounter{\expandafter\noexpand
3293 \csname theH\@gls@counter\endcsname}%
3294 }%
3295 \fi
3296 }

```

`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.

```

3297 \def\@set@glo@numformat#1#2#3#4{%
3298 \expandafter\@glo@check@mkidxrangechar#3\@nil
3299 \protected@edef#1{%
3300 \@glo@prefix setentrycounter[#4]{#2}%
3301 \expandafter\string\csname\@glo@suffix\endcsname
3302 }%
3303 \@gls@checkmkidxchars#1%
3304 }

```

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.

```

3305 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3306 \if#1(\relax
3307 \def\@glo@prefix{(%}
3308 \if\relax#2\relax
3309 \def\@glo@suffix{glsnumberformat}%
3310 \else
3311 \def\@glo@suffix{#2}%
3312 \fi
3313 \else
3314 \if#1)\relax
3315 \def\@glo@prefix{)}%
3316 \if\relax#2\relax
3317 \def\@glo@suffix{glsnumberformat}%
3318 \else
3319 \def\@glo@suffix{#2}%
3320 \fi
3321 \else
3322 \def\@glo@prefix{}\def\@glo@suffix{#1#2}%

```



```

3323 \fi
3324 \fi}

```

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

```

3325 \newcommand*{\@gls@escbsdq}[1]{%
3326   \def\@gls@checkedmkidx{%
3327     \let\gls@xdyststring=#1\relax
3328     \@onelevel@sanitize\gls@xdyststring
3329     \edef\do@gls@xdycheckbackslash{%
3330       \noexpand\@gls@xdycheckbackslash\gls@xdyststring\noexpand\@nil
3331       \@backslashchar\@backslashchar\noexpand\null}%
3332     \do@gls@xdycheckbackslash
3333     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdyststring}%
3334     \def\@gls@checkedmkidx{%
3335       \expandafter\@gls@xdycheckquote\gls@xdyststring\@nil""\null
3336       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdyststring}%

```

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

```

3337 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3338 {%
3339   \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3340   \@onelevel@sanitize\@gls@sanitized@tmp
3341   \edef\gls@dosubst{%
3342     \noexpand\DTLsubstituteall\noexpand\gls@xdyststring
3343     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3344   }%
3345   \gls@dosubst
3346 }%

```

Assign to required control sequence

```

3347 \let#1=\gls@xdyststring
3348 }

```

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

`checkmkidxchars`

```

3349 \newcommand{\@gls@checkmkidxchars}[1]{%
3350   \ifglxsindy
3351     \@gls@escbsdq{#1}%
3352   \else
3353     \def\@gls@checkedmkidx{%
3354       \expandafter\@gls@checkquote#1\@nil""\null
3355       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3356     \def\@gls@checkedmkidx{%
3357       \expandafter\@gls@checkescquote#1\@nil\""\null
3358       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3359     \def\@gls@checkedmkidx{%
3360       \expandafter\@gls@checkescactual#1\@nil\?\?\null
3361       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%

```

```

3362 \def\@gls@checkedmkidx{}%
3363 \expandafter\@gls@checkactual#1\@nil??\null
3364 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3365 \def\@gls@checkedmkidx{}%
3366 \expandafter\@gls@checkbar#1\@nil||\null
3367 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3368 \def\@gls@checkedmkidx{}%
3369 \expandafter\@gls@checkescbar#1\@nil|||\null
3370 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3371 \def\@gls@checkedmkidx{}%
3372 \expandafter\@gls@checklevel#1\@nil!!\null
3373 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3374 \fi
3375 }

```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```

3376 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gls@tmpb Define temporary token

```

3377 \newtoks\@gls@tmpb

```

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

```

3378 \def\@gls@checkquote#1"#2"#3\null{%
3379 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3380 \toks@={#1}%
3381 \ifx\null#2\null
3382 \ifx\null#3\null
3383 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3384 \def\@gls@checkquote{\relax}%
3385 \else
3386 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3387 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3388 \def\@gls@checkquote{\@gls@checkquote#3\null}%
3389 \fi
3390 \else
3391 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3392 \@gls@quotechar\@gls@quotechar}%
3393 \ifx\null#3\null
3394 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
3395 \else
3396 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
3397 \fi
3398 \fi
3399 \@gls@checkquote
3400 }

```

s@checkescquote Do the same for \:

```

3401 \def\@gls@checkescquote#1\"#2\"#3\null{%
3402   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3403   \toks@={#1}%
3404   \ifx\null#2\null
3405     \ifx\null#3\null
3406       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3407       \def\@@gls@checkescquote{\relax}%
3408     \else
3409       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3410         \@gls@quotechar\string\"@gls@quotechar
3411         \@gls@quotechar\string\"@gls@quotechar}%
3412       \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3413     \fi
3414   \else
3415     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3416       \@gls@quotechar\string\"@gls@quotechar}%
3417     \ifx\null#3\null
3418       \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3419     \else
3420       \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3421     \fi
3422   \fi
3423 \@@gls@checkescquote
3424 }

```

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

```

3425 \def\@gls@checkescactual#1\?#2\?#3\null{%
3426   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3427   \toks@={#1}%
3428   \ifx\null#2\null
3429     \ifx\null#3\null
3430       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3431       \def\@@gls@checkescactual{\relax}%
3432     \else
3433       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3434         \@gls@quotechar\string\"@gls@actualchar
3435         \@gls@quotechar\string\"@gls@actualchar}%
3436       \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
3437     \fi
3438   \else
3439     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3440       \@gls@quotechar\string\"@gls@actualchar}%
3441     \ifx\null#3\null
3442       \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3443     \else
3444       \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3445     \fi
3446   \fi
3447 \@@gls@checkescactual

```

3448 }

gls@checkeschar Similarly for \|:

```
3449 \def\@gls@checkeschar#1\|#2\|#3\null{%
3450   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3451   \toks@={#1}%
3452   \ifx\null#2\null
3453   \ifx\null#3\null
3454     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3455     \def\@gls@checkeschar{\relax}%
3456   \else
3457     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3458       \@gls@quotechar\string"\@gls@encapchar
3459       \@gls@quotechar\string"\@gls@encapchar}%
3460     \def\@gls@checkeschar{\@gls@checkeschar#3\null}%
3461   \fi
3462 \else
3463   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3464     \@gls@quotechar\string"\@gls@encapchar}%
3465   \ifx\null#3\null
3466     \def\@gls@checkeschar{\@gls@checkeschar#2\|\|\null}%
3467   \else
3468     \def\@gls@checkeschar{\@gls@checkeschar#2\|#3\null}%
3469   \fi
3470 \fi
3471 \@gls@checkeschar
3472 }
```

s@checkesclevel Similarly for \!:

```
3473 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3474   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3475   \toks@={#1}%
3476   \ifx\null#2\null
3477   \ifx\null#3\null
3478     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3479     \def\@gls@checkesclevel{\relax}%
3480   \else
3481     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3482       \@gls@quotechar\string"\@gls@levelchar
3483       \@gls@quotechar\string"\@gls@levelchar}%
3484     \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3485   \fi
3486 \else
3487   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3488     \@gls@quotechar\string"\@gls@levelchar}%
3489   \ifx\null#3\null
3490     \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
3491   \else
3492     \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%

```

```

3493 \fi
3494 \fi
3495 \@gls@checkesclevel
3496 }

```

\gls@checkbar and for |:

```

3497 \def\@gls@checkbar#1|#2|#3\null{%
3498 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3499 \toks@={#1}%
3500 \ifx\null#2\null
3501 \ifx\null#3\null
3502 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3503 \def\@gls@checkbar{\relax}%
3504 \else
3505 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3506 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3507 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3508 \fi
3509 \else
3510 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3511 \@gls@quotechar\@gls@encapchar}%
3512 \ifx\null#3\null
3513 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3514 \else
3515 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3516 \fi
3517 \fi
3518 \@gls@checkbar
3519 }

```

@gls@checklevel and for !:

```

3520 \def\@gls@checklevel#1!#2!#3\null{%
3521 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3522 \toks@={#1}%
3523 \ifx\null#2\null
3524 \ifx\null#3\null
3525 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3526 \def\@gls@checklevel{\relax}%
3527 \else
3528 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3529 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3530 \def\@gls@checklevel{\@gls@checklevel#3\null}%
3531 \fi
3532 \else
3533 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3534 \@gls@quotechar\@gls@levelchar}%
3535 \ifx\null#3\null
3536 \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3537 \else

```

```

3538     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3539     \fi
3540 \fi
3541 \@gls@checklevel
3542 }

```

gls@checkactual and for ?:

```

3543 \def\@gls@checkactual#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@checkactual{\relax}%
3550     \else
3551       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3552         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3553       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3554     \fi
3555   \else
3556     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3557       \@gls@quotechar\@gls@actualchar}%
3558     \ifx\null#3\null
3559       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3560     \else
3561       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3562     \fi
3563   \fi
3564   \@gls@checkactual
3565 }

```

s@xdycheckquote As before but for use with xindy

```

3566 \def\@gls@xdycheckquote#1"#2"#3\null{%
3567   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3568   \toks@={#1}%
3569   \ifx\null#2\null
3570     \ifx\null#3\null
3571       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3572       \def\@gls@xdycheckquote{\relax}%
3573     \else
3574       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3575         \string\string\}%
3576       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3577     \fi
3578   \else
3579     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3580       \string\string\}%
3581     \ifx\null#3\null
3582       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%

```

```

3583 \else
3584 \def\@@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3585 \fi
3586 \fi
3587 \@@gls@xdycheckquote
3588 }

```

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

```

3589 \edef\def@gls@xdycheckbackslash{%
3590 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3591 ##2\@backslashchar##3\noexpand\null{%
3592 \noexpand\@gls@tmpb=\noexpand\expandafter
3593 {\noexpand\@gls@checkedmkidx}%
3594 \noexpand\toks@={##1}%
3595 \noexpand\ifx\noexpand\null##2\noexpand\null
3596 \noexpand\ifx\noexpand\null##3\noexpand\null
3597 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3598 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3599 \noexpand\def\noexpand\@@gls@xdycheckbackslash{\relax}%
3600 \noexpand\else
3601 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3602 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3603 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3604 \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
3605 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3606 \noexpand\fi
3607 \noexpand\else
3608 \noexpand\edef\noexpand\@gls@checkedmkidx{%
3609 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3610 \@backslashchar\@backslashchar}%
3611 \noexpand\ifx\noexpand\null##3\noexpand\null
3612 \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
3613 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3614 \@backslashchar\noexpand\null}%
3615 \noexpand\else
3616 \noexpand\def\noexpand\@@gls@xdycheckbackslash{%
3617 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3618 ##3\noexpand\null}%
3619 \noexpand\fi
3620 \noexpand\fi
3621 \noexpand\@@gls@xdycheckbackslash
3622 }%
3623 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3624 \def@gls@xdycheckbackslash

```

lsdohypertarget

```

3625 \newlength\gls@tmplen
3626 \newcommand*{\glsdohypertarget}[2]{%

```

```

3627 \@glsshowtarget{#1}%
3628 \settoheight{\gls@tmplen}{#2}%
3629 \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3630 }

```

`\glsdohyperlink`

```

3631 \newcommand*{\glsdohyperlink}[2]{%
3632 \@glsshowtarget{#1}%
3633 \hyperlink{#1}{#2}%
3634 }

```

`\glsdonohyperlink`

```

3635 \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`.

```

3636 \ifcsundef{hyperlink}%
3637 {%
3638 \let\@glslink\glsdonohyperlink
3639 }%
3640 {%
3641 \let\@glslink\glsdohyperlink
3642 }

```

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

```

3643 \ifcsundef{hypertarget}%
3644 {%
3645 \let\@glstarget\@secondoftwo
3646 }%
3647 {%
3648 \let\@glstarget\glsdohypertarget
3649 }

```

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

`\glsdisablehyper`

```

3650 \newcommand{\glsdisablehyper}{%
3651 \KV@glslink@hyperfalse
3652 \let\@glslink\glsdonohyperlink
3653 \let\@glstarget\@secondoftwo
3654 }

```

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

`\glsenablehyper`

```

3655 \newcommand{\glsenablehyper}{%
3656 \KV@glslink@hypertrue

```



```

3657 \let\@glslink\glsdohyperlink
3658 \let\@glstarget\glsdohypertarget
3659 }

```

Provide some convenience commands if not already defined:

```

3660 \providecommand{\@firstofthree}[3]{#1}
3661 \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`

```

3662 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}

```

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

`\@gls`

```

3663 \newcommand*{\@gls}[2][ ]{%
3664   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[ ]}%
3665 }

```

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

```

3666 \def\@gls@#1#2[#3]{%
3667   \glsdoifexists{#2}%
3668   {%
3669     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3670     \let\glsifplural\@secondoftwo
3671     \let\gls caps case\@firstofthree
3672     \let\gls custom text\@empty
3673     \def\gls insert{#3}%

```

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

```

3674   \def\@gls@text{\csname gls@\gls type @entryfmt\endcsname}%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronym type`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

3675   \@gls@link[#1]{#2}{\@gls@text}%

```

Indicate that this entry has now been used

```
3676 \ifKV@glslink@local
3677 \glsllocalunset{#2}%
3678 \else
3679 \glunset{#2}%
3680 \fi
3681 }%

3682 \glspostlinkhook
3683 }
```

`\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`

```
3684 \newrobustcmd*{\Gls}{\@Gls@hyp@opt\@Gls}
```

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

```
3685 \newcommand*{\@Gls}[2][{}]{%
3686 \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[]}%
3687 }
```

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

```
3688 \def\@Gls@#1#2[#3]{%
3689 \glstoifexists{#2}%
3690 {%
3691 \let\do@glslink@checkfirsthyper\@glslink@checkfirsthyper

3692 \let\glslifplural\@secondoftwo
3693 \let\glscapscase\@secondofthree
3694 \let\glscustomtext\@empty
3695 \def\glinsert{#3}%
```

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

```
3696 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@glslink` 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.

```
3697 \@glslink[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3698 \ifKV@glslink@local
3699 \glsllocalunset{#2}%
3700 \else
3701 \glunset{#2}%
3702 \fi
3703 }%
```

```

3704 \glspostlinkhook
3705 }

```

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

\GLS

```

3706 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}

```

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

```

3707 \newcommand*{\@GLS}[2] [] {%
3708   \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2} []}%
3709 }

```

\@GLS@ Read in the final optional argument:

```

3710 \def\@GLS@#1#2[#3]{%
3711   \glsdoifexists{#2}%
3712   {%
3713     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3714     \let\glsifplural\@secondoftwo
3715     \let\glsupcase\@thirdofthree
3716     \let\glscustomtext\@empty
3717     \def\glsinsert{#3}%

```

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

```

3718   \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.

```

3719   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3720   \ifKV@glslink@local
3721     \glslocalunset{#2}%
3722   \else
3723     \glsunset{#2}%
3724   \fi
3725 }%

```

```

3726 \glspostlinkhook
3727 }

```

\glspl behaves in the same way as \gls except it uses the plural form.

\glspl

```

3728 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}

```

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

```

3729 \newcommand*{\@glspl}[2] [] {%
3730   \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2} []}%
3731 }

```

`\@glsp1@` Read in the final optional argument:

```
3732 \def\@glsp1@#1#2[#3]{%
3733   \glsdoidexists{#2}%
3734   {%
3735     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper

3736     \let\gl@ifplural\@firstoftwo
3737     \let\gl@scapscase\@firstofthree
3738     \let\glscustomtext\@empty
3739     \def\glinsert{#3}%
```

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

```
3740   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@gl@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.

```
3741   \@gl@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3742   \ifKV@gl@link@local
3743     \gl@localunset{#2}%
3744   \else
3745     \gl@unset{#2}%
3746   \fi
3747 }%

3748 \glspostlinkhook
3749 }
```

`\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`

```
3750 \newrobustcmd*{\Glspl}{\@gl@hyp@opt\@Glspl}
```

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

```
3751 \newcommand*{\@Glspl}[2][ ]{%
3752   \new@ifnextchar[\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[ ]}%
3753 }
```

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

```
3754 \def\@Glspl@#1#2[#3]{%
3755   \glsdoidexists{#2}%
3756   {%
3757     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper

3758     \let\gl@ifplural\@firstoftwo
3759     \let\gl@scapscase\@secondofthree
3760     \let\glscustomtext\@empty
3761     \def\glinsert{#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.

```
3762 \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.

```
3763 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3764 \ifKV@glslink@local
```

```
3765 \glsllocalunset{#2}%
```

```
3766 \else
```

```
3767 \glsunset{#2}%
```

```
3768 \fi
```

```
3769 }%
```

```
3770 \glspostlinkhook
```

```
3771 }
```

\GLSp1 behaves like \glsp1 except that all the link text is converted to uppercase.

\GLSp1

```
3772 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3773 \newcommand*{\@GLSp1}[2] []{%
```

```
3774 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
```

```
3775 }
```

\@GLSp1 Read in the final optional argument:

```
3776 \def\@GLSp1@#1#2[#3]{%
```

```
3777 \glsdofexists{#2}%
```

```
3778 {%
```

```
3779 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```
3780 \let\glsifplural\@firstoftwo
```

```
3781 \let\glscapscase\@thirdofthree
```

```
3782 \let\glscustomtext\@empty
```

```
3783 \def\glsinsert{#3}%
```

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

```
3784 \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.

```
3785 \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```

3786 \ifKV@glslink@local
3787 \glsllocalunset{#2}%
3788 \else
3789 \glssunset{#2}%
3790 \fi
3791 }%

3792 \glspostlinkhook
3793 }

```

`\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:

```

3794 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3795 \newcommand*{\@glsdisp}[3][ ]{%
3796 \glsdofexists{#2}%

3797 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

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

```

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

```

3802 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

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

```

Indicate that this entry has now been used

```

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

3810 \glspostlinkhook
3811 }

```

checkfirsthyper Instead of just setting \do@gl@link@checkfirsthyper to \relax in \@gl@field@link, set it to \@gl@link@nocheckfirsthyper in case some other action needs to take place.

```
3812 \newcommand*{\@gl@link@nocheckfirsthyper}{}
```

@gl@field@link

```
3813 \newcommand{\@gl@field@link}[3]{%
3814   \glstoifexists{#2}%
3815   {%
3816     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3817     \@gl@link[#1]{#2}{#3}%
3818   }%

3819   \glspostlinkhook
3820 }
```

\glstext behaves like \gl except it always uses the value given by the text key and it doesn't mark the entry as used.

\glstext

```
3821 \newrobustcmd*{\glstext}{\@gl@hyp@opt\@glstext}
```

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

```
3822 \newcommand*{\@glstext}[2][{}]{%
3823   \new@ifnextchar[\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[ ]}}
```

Read in the final optional argument:

```
3824 \def\@glstext@#1#2[#3]{%
3825   \@gl@field@link{#1}{#2}{\glstentrytext{#2}{#3}}%
3826 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

\GLStext

```
3827 \newrobustcmd*{\GLStext}{\@gl@hyp@opt\@GLStext}
```

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

```
3828 \newcommand*{\@GLStext}[2][{}]{%
3829   \new@ifnextchar[\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}[ ]}}
```

Read in the final optional argument:

```
3830 \def\@GLStext@#1#2[#3]{%
3831   \@gl@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}{#3}}}%
3832 }
```

\Glstext behaves like \glstext except that the first letter of the text is converted to uppercase.

\Glstext

```
3833 \newrobustcmd*{\Glstext}{\@gl@hyp@opt\@Glstext}
```

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

```
3834 \newcommand*{\@Glstext}[2] [] {%  
3835   \new@ifnextchar[{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3836 \def\@Glstext@#1#2[#3] {%  
3837   \@gls@field@link{#1}{#2}{\glstrytext{#2}#3}%  
3838 }
```

`\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`

```
3839 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3840 \newcommand*{\@glsfirst}[2] [] {%  
3841   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3842 \def\@glsfirst@#1#2[#3] {%  
3843   \@gls@field@link{#1}{#2}{\glstryfirst{#2}#3}%  
3844 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
3845 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3846 \newcommand*{\@Glsfirst}[2] [] {%  
3847   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3848 \def\@Glsfirst@#1#2[#3] {%  
3849   \@gls@field@link{#1}{#2}{\glstryfirst{#2}#3}%  
3850 }
```

`\GLSfirst` behaves like `\Glsfirst` except it displays the text in uppercase.

`\GLSfirst`

```
3851 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3852 \newcommand*{\@GLSfirst}[2] [] {%  
3853   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3854 \def\@GLSfirst@#1#2[#3] {%  
3855   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstryfirst{#2}#3}}%  
3856 }
```

`\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`

```
3857 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3858 \newcommand*{\@glsplural}[2] [] {%
```

```
3859   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3860 \def\@glsplural@#1#2[#3]{%
```

```
3861   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}%
```

```
3862 }
```

`\Glsplural` behaves like `\glsplural` except that the first letter is converted to uppercase.

`\Glsplural`

```
3863 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3864 \newcommand*{\@Glsplural}[2] [] {%
```

```
3865   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3866 \def\@Glsplural@#1#2[#3]{%
```

```
3867   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}%
```

```
3868 }
```

`\GLSplural` behaves like `\glsplural` except that the text is converted to uppercase.

`\GLSplural`

```
3869 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
3870 \newcommand*{\@GLSplural}[2] [] {%
```

```
3871   \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3872 \def\@GLSplural@#1#2[#3]{%
```

```
3873   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
```

```
3874 }
```

`\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`

```
3875 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3876 \newcommand*{\@glsfirstplural}[2] [] {%
```

```
3877   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3878 \def\@glsfirstplural@#1#2[#3]{%
```

```
3879   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
```

```
3880 }
```

`\Glsfirstplural` behaves like `\glsfirstplural` except that the first letter is converted to uppercase.

`\Glsfirstplural`

```
3881 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3882 \newcommand*{\@Glsfirstplural}[2] [] {%
```

```
3883   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3884 \def\@Glsfirstplural@#1#2[#3] {%
```

```
3885   \@gls@field@link{#1}{#2}{\glstryfirstplural{#2}#3}%
```

```
3886 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
3887 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3888 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3889   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3890 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3891   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstryfirstplural{#2}#3}}%
```

```
3892 }
```

`\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`

```
3893 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3894 \newcommand*{\@glsname}[2] [] {%
```

```
3895   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3896 \def\@glsname@#1#2[#3] {%
```

```
3897   \@gls@field@link{#1}{#2}{\glstryname{#2}#3}%
```

```
3898 }
```

`\Glsname` behaves like `\glsname` except that the first letter is converted to uppercase.

`\Glsname`

```
3899 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3900 \newcommand*{\@Glsname}[2] [] {%
```

```
3901   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3902 \def\@Glsname#1#2[#3]{%
3903   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
3904 }
```

\GLSname behaves like \glsname except that the link text is converted to uppercase.

\GLSname

```
3905 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3906 \newcommand*{\@GLSname}[2][{}]{%
3907   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3908 \def\@GLSname#1#2[#3]{%
3909   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryname{#2}#3}}%
3910 }
```

\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

```
3911 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3912 \newcommand*{\@glsdesc}[2][{}]{%
3913   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3914 \def\@glsdesc#1#2[#3]{%
3915   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3916 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3917 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3918 \newcommand*{\@Glsdesc}[2][{}]{%
3919   \new@ifnextchar[{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2}[]}]}
```

Read in the final optional argument:

```
3920 \def\@Glsdesc#1#2[#3]{%
3921   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3922 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3923 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3924 \newcommand*{\@GLSdesc}[2] [] {%
3925   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3926 \def\@GLSdesc@#1#2[#3] {%
3927   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydesc{#2}#3}}%
3928 }
```

`\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`

```
3929 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3930 \newcommand*{\@glsdescplural}[2] [] {%
3931   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3932 \def\@glsdescplural@#1#2[#3] {%
3933   \@gls@field@link{#1}{#2}{\glsentrydescplural{#2}#3}%
3934 }
```

`\Glsdescplural` behaves like `\glsdescplural` except that the first letter is converted to uppercase.

`\Glsdescplural`

```
3935 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3936 \newcommand*{\@Glsdescplural}[2] [] {%
3937   \new@ifnextchar [{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3938 \def\@Glsdescplural@#1#2[#3] {%
3939   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3940 }
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
3941 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3942 \newcommand*{\@GLSdescplural}[2] [] {%
3943   \new@ifnextchar [{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3944 \def\@GLSdescplural@#1#2[#3] {%
3945   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3946 }
```

`\glssymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glssymbol`

```
3947 \newrobustcmd*{\glssymbol}{\@gls@hyp@opt\@glssymbol}
```

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

```
3948 \newcommand*{\@glssymbol}[2] [] {%
```

```
3949   \new@ifnextchar[{\@glssymbol@{#1}{#2}}{\@glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3950 \def\@glssymbol@#1#2[#3] {%
```

```
3951   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
3952 }
```

`\Glssymbol` behaves like `\glssymbol` except that the first letter is converted to uppercase.

`\Glssymbol`

```
3953 \newrobustcmd*{\Glssymbol}{\@gls@hyp@opt\@Glssymbol}
```

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

```
3954 \newcommand*{\@Glssymbol}[2] [] {%
```

```
3955   \new@ifnextchar[{\@Glssymbol@{#1}{#2}}{\@Glssymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3956 \def\@Glssymbol@#1#2[#3] {%
```

```
3957   \@gls@field@link{#1}{#2}{\glstentrysymbol{#2}#3}%
```

```
3958 }
```

`\GLSsymbol` behaves like `\glssymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3959 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3960 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3961   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3962 \def\@GLSsymbol@#1#2[#3] {%
```

```
3963   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrysymbol{#2}#3}}%
```

```
3964 }
```

`\glsymbolplural` behaves like `\gls` except it always uses the value given by the symbol-plural key and it doesn't mark the entry as used.

`\glsymbolplural`

```
3965 \newrobustcmd*{\glsymbolplural}{\@gls@hyp@opt\@glsymbolplural}
```

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

```
3966 \newcommand*{\@glsymbolplural}[2] [] {%
```

```
3967   \new@ifnextchar[{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3968 \def\@glssymbolplural@#1#2[#3]{%
3969   \@gls@field@link{#1}{#2}{\glstrysymbolplural{#2}#3}%
3970 }
```

`\Glssymbolplural` behaves like `\glssymbolplural` except that the first letter is converted to uppercase.

`Glssymbolplural`

```
3971 \newrobustcmd*{\Glssymbolplural}{\@gls@hyp@opt\@Glssymbolplural}
```

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

```
3972 \newcommand*{\@Glssymbolplural}[2] [] {%
3973   \new@ifnextchar[{\@Glssymbolplural@{#1}{#2}}{\@Glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3974 \def\@Glssymbolplural@#1#2[#3]{%
3975   \@gls@field@link{#1}{#2}{\glstrysymbolplural{#2}#3}%
3976 }
```

`\GLSsymbolplural` behaves like `\glssymbolplural` except that the link text is converted to uppercase.

`GLSsymbolplural`

```
3977 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
3978 \newcommand*{\@GLSsymbolplural}[2] [] {%
3979   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3980 \def\@GLSsymbolplural@#1#2[#3]{%
3981   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstrysymbolplural{#2}#3}}%
3982 }
```

`\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`

```
3983 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3984 \newcommand*{\@glsuseri}[2] [] {%
3985   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3986 \def\@glsuseri@#1#2[#3]{%
3987   \@gls@field@link{#1}{#2}{\glstryuseri{#2}#3}%
3988 }
```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

\Glsuseri

```
3989 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3990 \newcommand*{\@Glsuseri}[2] [] {%
```

```
3991   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3992 \def\@Glsuseri@#1#2[#3]{%
```

```
3993   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
```

```
3994 }
```

\Glsuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
3995 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3996 \newcommand*{\@GLSuseri}[2] [] {%
```

```
3997   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3998 \def\@GLSuseri@#1#2[#3]{%
```

```
3999   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseri{#2}#3}}%
```

```
4000 }
```

\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

```
4001 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
4002 \newcommand*{\@glsuserii}[2] [] {%
```

```
4003   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4004 \def\@glsuserii@#1#2[#3]{%
```

```
4005   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
4006 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
4007 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
4008 \newcommand*{\@Glsuserii}[2] [] {%
```

```
4009   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4010 \def\@Glsuserii@#1#2[#3]{%
```

```
4011   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
4012 }
```

`\GLSuserii` behaves like `\glsuserii` except that the link text is converted to uppercase.

`\GLSuserii`

```
4013 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
4014 \newcommand*{\@GLSuserii}[2] [] {%
```

```
4015   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4016 \def\@GLSuserii@#1#2[#3] {%
```

```
4017   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}%
```

```
4018 }
```

`\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`

```
4019 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
4020 \newcommand*{\@glsuseriii}[2] [] {%
```

```
4021   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4022 \def\@glsuseriii@#1#2[#3] {%
```

```
4023   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}}%
```

```
4024 }
```

`\Glsuseriii` behaves like `\glsuseriii` except that the first letter is converted to uppercase.

`\Glsuseriii`

```
4025 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
4026 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
4027   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4028 \def\@Glsuseriii@#1#2[#3] {%
```

```
4029   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}}%
```

```
4030 }
```

`\GLSuseriii` behaves like `\glsuseriii` except that the link text is converted to uppercase.

`\GLSuseriii`

```
4031 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
4032 \newcommand*{\@GLSuseriii}[2] [] {%
```

```
4033   \new@ifnextchar[{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}]}
```



Read in the final optional argument:

```
4034 \def\@GLSuseriii@#1#2[#3]{%
4035   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}%
4036 }
```

\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

```
4037 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
4038 \newcommand*{\@glsuseriv}[2][\%]
4039   \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4040 \def\@glsuseriv@#1#2[#3]{%
4041   \@gls@field@link{#1}{#2}{\glsentryuseriv{#2}#3}%
4042 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
4043 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
4044 \newcommand*{\@Glsuseriv}[2][\%]
4045   \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4046 \def\@Glsuseriv@#1#2[#3]{%
4047   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
4048 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
4049 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```

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

```
4050 \newcommand*{\@GLSuseriv}[2][\%]
4051   \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4052 \def\@GLSuseriv@#1#2[#3]{%
4053   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
4054 }
```

\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

```
4055 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
4056 \newcommand*{\@glsuserv}[2] [] {%
4057   \new@ifnextchar [{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4058 \def\@glsuserv@#1#2[#3] {%
4059   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}%
4060 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
4061 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
4062 \newcommand*{\@GLsuserv}[2] [] {%
4063   \new@ifnextchar [{\@GLsuserv@{#1}{#2}}{\@GLsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4064 \def\@GLsuserv@#1#2[#3] {%
4065   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}%
4066 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4067 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
4068 \newcommand*{\@GLSuserv}[2] [] {%
4069   \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4070 \def\@GLSuserv@#1#2[#3] {%
4071   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4072 }
```

\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

```
4073 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4074 \newcommand*{\@glsuservi}[2] [] {%
4075   \new@ifnextchar [{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4076 \def\@glsuservi@#1#2[#3] {%
4077   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
4078 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4079 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4080 \newcommand*{\@Glsuservi}[2][\@Glsuservi]
```

```
4081 \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4082 \def\@Glsuservi@#1#2[#3]{%
```

```
4083 \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
4084 }
```

\Glsuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4085 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
4086 \newcommand*{\@GLSuservi}[2][\@GLSuservi]
```

```
4087 \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2}[]}]
```

Read in the final optional argument:

```
4088 \def\@GLSuservi@#1#2[#3]{%
```

```
4089 \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuservi{#2}#3}}%
```

```
4090 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4091 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
4092 \newcommand*{\@ns@acrshort}[2][\@ns@acrshort]
```

```
4093 \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2}[]}]
```

```
4094 }
```

Read in the final optional argument:

```
4095 \def\@acrshort#1#2[#3]{%
```

```
4096 \glsdoifexists{#2}%
```

```
4097 {%
```

```
4098 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
4099 \let\glsifplural\@secondoftwo
```

```
4100 \let\glsapscase\@firstofthree
```

```
4101 \let\glsinsert\@empty
```

```
4102 \def\glscustomtext{%
```

```
4103 \acronymfont{\Glsentryshort{#2}#3}%
```

```
4104 }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4105 \@gls@link{#1}{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
4106 }%
```

```

4107 \glspostlinkhook
4108 }

```

\Acrshort

```

4109 \newrobustcmd*{\Acrshort}{\@gls@hyp@opt\@ns@Acrshort}

```

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

```

4110 \newcommand*{\ns@Acrshort}[2][\%
4111 \new@ifnextchar[\@Acrshort{#1}{#2}]{\@Acrshort{#1}{#2}[]}%
4112 }

```

Read in the final optional argument:

```

4113 \def\@Acrshort#1#2[#3]{%
4114 \glsdoifexists{#2}%
4115 {%
4116 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4117 \def\glslabel{#2}%
4118 \let\glsifplural\@secondoftwo
4119 \let\glsapscase\@secondofthree
4120 \let\glsinsert\@empty
4121 \def\glscustomtext{%
4122 \acronymfont{\Glsentryshort{#2}}#3%
4123 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

4124 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4125 }%
4126 \glspostlinkhook
4127 }

```

\ACRshort

```

4128 \newrobustcmd*{\ACRshort}{\@gls@hyp@opt\@ns@ACRshort}

```

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

```

4129 \newcommand*{\ns@ACRshort}[2][\%
4130 \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
4131 }

```

Read in the final optional argument:

```

4132 \def\@ACRshort#1#2[#3]{%
4133 \glsdoifexists{#2}%
4134 {%
4135 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4136 \def\glslabel{#2}%
4137 \let\glsifplural\@secondoftwo
4138 \let\glsapscase\@thirdofthree
4139 \let\glsinsert\@empty
4140 \def\glscustomtext{%
4141 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4142 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

4143 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4144 }%

4145 \glspostlinkhook
4146 }

```

Short plural:

\acrshortpl

```

4147 \newrobustcmd*{\acrshortpl}{\@gls@hyp@opt\@ns@acrshortpl}

```

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

```

4148 \newcommand*{\ns@acrshortpl}[2] [] {%
4149 \new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} []}%
4150 }

```

Read in the final optional argument:

```

4151 \def\@acrshortpl#1#2[#3]{%
4152 \glsdoifexists{#2}%
4153 {%

4154 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4155 \def\glslabel{#2}%
4156 \let\glsifplural\@firstoftwo
4157 \let\glsapscase\@firstofthree
4158 \let\glsinsert\@empty
4159 \def\glscustomtext{%
4160 \acronymfont{\glsentryshortpl{#2}}#3%
4161 }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

4162 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4163 }%

4164 \glspostlinkhook
4165 }

```

\Acrshortpl

```

4166 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\@ns@Acrshortpl}

```

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

```
4167 \newcommand*{\ns@Acrshortpl}[2] [] {%
4168   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
4169 }
```

Read in the final optional argument:

```
4170 \def\@Acrshortpl#1#2[#3] {%
4171   \glsdoifexists{#2}%
4172   {%
4173     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4174     \def\glslabel{#2}%
4175     \let\glsifplural\@firstoftwo
4176     \let\glscapscase\@secondofthree
4177     \let\glsinsert\@empty
4178     \def\glscustomtext{%
4179       \acronymfont{\Glsentryshortpl{#2}}#3%
4180     }%
```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```
4181   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4182   }%
4183   \glspostlinkhook
4184 }
```

\ACRshortpl

```
4185 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}
```

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

```
4186 \newcommand*{\ns@ACRshortpl}[2] [] {%
4187   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
4188 }
```

Read in the final optional argument:

```
4189 \def\@ACRshortpl#1#2[#3] {%
4190   \glsdoifexists{#2}%
4191   {%
4192     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
4193     \def\glslabel{#2}%
4194     \let\glsifplural\@firstoftwo
4195     \let\glsapscase\@thirdofthree
4196     \let\glsinsert\@empty
4197     \def\glscustomtext{%
4198       \mfirstucMakeUppercase{\acronymfont{\Glsentryshortpl{#2}}#3}%
4199     }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4200   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%  
4201 }%  
  
4202 \glspostlinkhook  
4203 }
```

\acrlong

```
4204 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\@ns@acrlong}
```

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

```
4205 \newcommand*{\ns@acrlong}[2][ ]{%  
4206   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2}[ ]}%  
4207 }
```

Read in the final optional argument:

```
4208 \def\@acrlong#1#2[#3]{%  
4209   \glsdoifexists{#2}%  
4210   {%  
  
4211     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4212     \def\glslabel{#2}%  
4213     \let\glsifplural\@secondoftwo  
4214     \let\glscapscase\@firstofthree  
4215     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4216   \def\glscustomtext{%  
4217     \glsentrylong{#2}#3%  
4218   }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4219   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%  
4220 }%  
  
4221 \glspostlinkhook  
4222 }
```

\Acrlong

```
4223 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\@ns@Acrlong}
```

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

```
4224 \newcommand*{\ns@Acrlong}[2][ ]{%  
4225   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[ ]}%  
4226 }
```

Read in the final optional argument:

```
4227 \def\@Acrlong#1#2[#3]{%  
4228   \glsdoifexists{#2}%  
4229   {%
```

```

4230 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

4231 \def\glslabel{#2}%
4232 \let\gl@ifplural\@secondoftwo
4233 \let\gl@scapscase\@secondofthree
4234 \let\gl@insert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4235 \def\glscustomtext{%
4236 \gl@entrylong{#2}#3%
4237 }%

```

Call \@gl@link. Note that \@gl@link sets \glstype.

```

4238 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4239 }%

4240 \glspostlinkhook
4241 }

```

\ACRlong

```

4242 \newrobustcmd*{\ACRlong}{\@gl@hyp@opt\ns@ACRlong}

```

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

```

4243 \newcommand*{\ns@ACRlong}[2][\gl@entrylong]{%
4244 \new@ifnextchar{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4245 }

```

Read in the final optional argument:

```

4246 \def\@ACRlong#1#2[#3]{%
4247 \gl@doifexists{#2}%
4248 {%

```

```

4249 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

```

```

4250 \def\glslabel{#2}%
4251 \let\gl@ifplural\@secondoftwo
4252 \let\gl@scapscase\@thirdofthree
4253 \let\gl@insert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4254 \def\glscustomtext{%
4255 \mfirstucMakeUppercase{\gl@entrylong{#2}#3}%
4256 }%

```

Call \@gl@link. Note that \@gl@link sets \glstype.

```

4257 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4258 }%

4259 \glspostlinkhook
4260 }

```



Short plural:

`\acrlongpl`

```
4261 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}
```

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

```
4262 \newcommand*{\ns@acrlongpl}[2][\%  
4263   \new@ifnextchar[\@acrlongpl{#1}{#2}]{\@acrlongpl{#1}{#2}[]}%  
4264 }
```

Read in the final optional argument:

```
4265 \def\@acrlongpl#1#2[#3]{%  
4266   \glsdoifexists{#2}%  
4267   {%  
  
4268     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4269     \def\glslabel{#2}%  
4270     \let\glsifplural\@firstoftwo  
4271     \let\glsupcase\@firstofthree  
4272     \let\glsinsert\@empty
```

Bug fix v4.02 removed `\acronymfont` from `\glscustomtext` (`\acronymfont` only designed for short form).

```
4273   \def\glscustomtext{%  
4274     \glsentrylongpl{#2}#3%  
4275   }%
```

Call `\@gls@link`. Note that `\@gls@link` sets `\glsstyle`.

```
4276   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4277   }%  
  
4278   \glspostlinkhook  
4279 }
```

`\Acrlongpl`

```
4280 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\@ns@Acrlongpl}
```

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

```
4281 \newcommand*{\ns@Acrlongpl}[2][\%  
4282   \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%  
4283 }
```

Read in the final optional argument:

```
4284 \def\@Acrlongpl#1#2[#3]{%  
4285   \glsdoifexists{#2}%  
4286   {%  
  
4287     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```

4288 \def\glslabel{#2}%
4289 \let\glsifplural\@firstoftwo
4290 \let\glsupcase\@secondofthree
4291 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4292 \def\glscustomtext{%
4293 \Glsentrylongpl{#2}#3%
4294 }%

```

Call \@gls@link. Note that \@gls@link sets \glsstyle.

```

4295 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4296 }%

```

```

4297 \glspostlinkhook
4298 }

```

\ACRlongpl

```

4299 \newrobustcmd*{\ACRlongpl}{\@gls@hyp@opt\@ns@ACRlongpl}

```

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

```

4300 \newcommand*{\@ns@ACRlongpl}[2][{}]{%
4301 \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2}[]}%
4302 }

```

Read in the final optional argument:

```

4303 \def\@ACRlongpl#1#2[#3]{%
4304 \glsdoifexists{#2}%
4305 {%

```

```

4306 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

4307 \def\glslabel{#2}%
4308 \let\glsifplural\@firstoftwo
4309 \let\glsupcase\@thirdofthree
4310 \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4311 \def\glscustomtext{%
4312 \mfirstucMakeUppercase{\Glsentrylongpl{#2}#3}%
4313 }%

```

Call \@gls@link. Note that \@gls@link sets \glsstyle.

```

4314 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4315 }%

```

```

4316 \glspostlinkhook
4317 }

```

## 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>}
```

```
4318 \newcommand*{\@gls@entry@field}[2]{%
4319   \csname glo@glsdetoklabel{#1}@#2\endcsname
4320 }
```

`glsletentryfield`

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

```
4321 \newcommand*{\glsletentryfield}[3]{%
4322   \letcs{#1}{glo@glsdetoklabel{#2}@#3}%
4323 }
```

`Gls@entry@field` Generic first letter uppercase version.

```
\@Gls@entry@field{\<label>}{\<field>}
```

```
4324 \newcommand*{\@Gls@entry@field}[2]{%
4325   \glsdoifexistsordo{#1}%
4326   {%
4327     \letcs\@glo@text{glo@glsdetoklabel{#1}@#2}%
4328     \ifdef\@glo@text
4329     {%
4330       \xmakefirstuc{\@glo@text}%
4331     }%
4332     {%
4333       ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4334       entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry
4335       label and the field name}%
4336     }%
4337   }%
4338   {%
4339     ??%
4340   }%
4341 }
```

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
4342 \newcommand*{\glsentryname}[1]{\@Gls@entry@field{#1}{name}}

\Glsentryname
4343 \newrobustcmd*{\Glsentryname}[1]{%
4344   \@Gls@entryname{#1}%
4345 }

\@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:
4346 \newcommand*{\@Gls@entryname}[1]{%
4347   \@Gls@entry@field{#1}{name}%
4348 }

ls@acrentryname Now the behaviour when \setacronymstyle is used:
4349 \newcommand*{\@Gls@acrentryname}[1]{%
4350   \ifglshaslong{#1}%
4351   {%
4352     \letcs\@glo@text{glo@\glsdetoklabel{#1}@name}%
4353     \expandafter\@gls@getbody\@glo@text{}\@nil
4354     \expandafter\ifx\@gls@body\glsentrylong\relax
4355       \expandafter\Glsentrylong\@gls@rest
4356     \else
4357       \expandafter\ifx\@gls@body\glsentryshort\relax
4358         \expandafter\Glsentryshort\@gls@rest
4359       \else
4360         \expandafter\ifx\@gls@body\acronymfont\relax
          Temporarily make \glsentryshort behave like \Glsentryshort. (This is on the assump-
          tion that the argument of \acronymfont is \glsentryshort{\langle label \rangle}, as that's the behaviour
          of the predefined acronym styles.) This is scoped to localise the effect of the assignment.
4361         {%
4362           \let\glsentryshort\Glsentryshort
4363           \@glo@text
4364         }%
4365       \else
4366         \xmakefirstuc{\@glo@text}%
4367       \fi
4368     \fi
4369   \fi
4370 }%
4371 {%
  Not an acronym
4372   \@Gls@entry@field{#1}{name}%
4373 }%
4374 }

```

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`

```
4375 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}
```

`\Glsentrydesc`

```
4376 \newrobustcmd*{\Glsentrydesc}[1]{%
4377   \@Gls@entry@field{#1}{desc}%
4378 }
```

Plural form:

`entrydescplural`

```
4379 \newcommand*{\glsentrydescplural}[1]{%
4380   \@gls@entry@field{#1}{descplural}%
4381 }
```

`entrydescplural`

```
4382 \newrobustcmd*{\Glsentrydescplural}[1]{%
4383   \@Gls@entry@field{#1}{descplural}%
4384 }
```

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`

```
4385 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}
```

`\Glsentrytext`

```
4386 \newrobustcmd*{\Glsentrytext}[1]{%
4387   \@Gls@entry@field{#1}{text}%
4388 }
```

Get the plural form:

`\glsentryplural`

```
4389 \newcommand*{\glsentryplural}[1]{%
4390   \@gls@entry@field{#1}{plural}%
4391 }
```

`\Glsentryplural`

```
4392 \newrobustcmd*{\Glsentryplural}[1]{%
4393   \@Gls@entry@field{#1}{plural}%
4394 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

`\glsentrysymbol`

```
4395 \newcommand*{\glsentrysymbol}[1]{%
4396   \@gls@entry@field{#1}{symbol}%
4397 }
```

`\Glsentrysymbol`

```
4398 \newrobustcmd*{\Glsentrysymbol}[1]{%
4399   \@Gls@entry@field{#1}{symbol}%
4400 }
```

Plural form:

`trysymbolplural`

```
4401 \newcommand*{\glsentrysymbolplural}[1]{%
4402   \@gls@entry@field{#1}{symbolplural}%
4403 }
```

`trysymbolplural`

```
4404 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4405   \@Gls@entry@field{#1}{symbolplural}%
4406 }
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

`\glsentryfirst`

```
4407 \newcommand*{\glsentryfirst}[1]{%
4408   \@gls@entry@field{#1}{first}%
4409 }
```

`\Glsentryfirst`

```
4410 \newrobustcmd*{\Glsentryfirst}[1]{%
4411   \@Gls@entry@field{#1}{first}%
4412 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

`ntryfirstplural`

```
4413 \newcommand*{\glsentryfirstplural}[1]{%
4414   \@gls@entry@field{#1}{firstpl}%
4415 }
```

`ntryfirstplural`

```
4416 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4417   \@Gls@entry@field{#1}{firstpl}%
4418 }
```

sentrytitlecase

```
4419 \newrobustcmd*{\@glsentrytitlecase}[2]{%
4420   \glsfieldfetch{#1}{#2}{\@gls@value}%
4421   \xcapitalisewords{\@gls@value}%
4422 }
4423 \ifdef\texorpdfstring
4424 {
4425   \newcommand*{\glsentrytitlecase}[2]{%
4426     \texorpdfstring
4427       {\@glsentrytitlecase{#1}{#2}}%
4428       {\@gls@entry@field{#1}{#2}}%
4429   }
4430 }
4431 {
4432   \newcommand*{\glsentrytitlecase}[2]{\@glsentrytitlecase{#1}{#2}}
4433 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
4434 \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

```
4435 \newcommand*{\glsentrysort}[1]{%
4436   \@gls@entry@field{#1}{sort}%
4437 }
```

\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.

```
4438 \newcommand*{\glsentryuseri}[1]{%
4439   \@gls@entry@field{#1}{useri}%
4440 }
```

\Glsentryuseri

```
4441 \newrobustcmd*{\Glsentryuseri}[1]{%
4442   \@Gls@entry@field{#1}{useri}%
4443 }
```

\glsentryuserii Get the second user key (as specified by the user2 when the entry was defined). The argument is the label associated with the entry.

```
4444 \newcommand*{\glsentryuserii}[1]{%
4445   \@gls@entry@field{#1}{userii}%
4446 }
```

```

\Glsentryuserii
4447 \newrobustcmd*{\Glsentryuserii}[1]{%
4448   \@Gls@entry@field{#1}{userii}%
4449 }

\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.
4450 \newcommand*{\glentryuseriii}[1]{%
4451   \@Gls@entry@field{#1}{useriii}%
4452 }

Glsentryuseriii
4453 \newrobustcmd*{\Glsentryuseriii}[1]{%
4454   \@Gls@entry@field{#1}{useriii}%
4455 }

\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.
4456 \newcommand*{\glentryuseriv}[1]{%
4457   \@Gls@entry@field{#1}{useriv}%
4458 }

\Glsentryuseriv
4459 \newrobustcmd*{\Glsentryuseriv}[1]{%
4460   \@Gls@entry@field{#1}{useriv}%
4461 }

\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.
4462 \newcommand*{\glentryuserv}[1]{%
4463   \@Gls@entry@field{#1}{userv}%
4464 }

\Glsentryuserv
4465 \newrobustcmd*{\Glsentryuserv}[1]{%
4466   \@Gls@entry@field{#1}{userv}%
4467 }

\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.
4468 \newcommand*{\glentryuservi}[1]{%
4469   \@Gls@entry@field{#1}{uservi}%
4470 }

\Glsentryuservi
4471 \newrobustcmd*{\Glsentryuservi}[1]{%
4472   \@Gls@entry@field{#1}{uservi}%
4473 }

```



`\glsentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4474 \newcommand*{\glsentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4475 \newrobustcmd*{\Glsentryshort}[1]{%
4476   \@Gls@entry@field{#1}{short}%
4477 }
```

`\glsentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4478 \newcommand*{\glsentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`\Glsentryshortpl`

```
4479 \newrobustcmd*{\Glsentryshortpl}[1]{%
4480   \@Gls@entry@field{#1}{shortpl}%
4481 }
```

`\glsentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4482 \newcommand*{\glsentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4483 \newrobustcmd*{\Glsentrylong}[1]{%
4484   \@Gls@entry@field{#1}{long}%
4485 }
```

`\glsentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4486 \newcommand*{\glsentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4487 \newrobustcmd*{\Glsentrylongpl}[1]{%
4488   \@Gls@entry@field{#1}{longpl}%
4489 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4490 \newcommand*{\glsentryfull}[1]{%
4491   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4492 }
```

`\Glsentryfull`

```
4493 \newrobustcmd*{\Glsentryfull}[1]{%
4494   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4495 }
```

`\glsentryfullpl`

```
4496 \newcommand*{\glsentryfullpl}[1]{%
4497   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4498 }
```

`\Glsentryfullpl`

```
4499 \newrobustcmd*{\Glsentryfullpl}[1]{%
4500   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4501 }
```

`entrynumberlist` Displays the number list as is.

```
4502 \newcommand*{\glsentrynumberlist}[1]{%
4503   \glsdoifexists{#1}%
4504   {%
4505     \@gls@entry@field{#1}{numberlist}%
4506   }%
4507 }
```

`splaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4508 \@ifpackageloaded{hyperref} {%
4509   \newcommand*{\glsdisplaynumberlist}[1]{%
4510     \GlossariesWarning
4511     {%
4512       \string\glsdisplaynumberlist\space
4513       doesn't work with hyperref.^^JUsing
4514       \string\glsentrynumberlist\space instead%
4515     }%
4516     \glsentrynumberlist{#1}%
4517   }%
4518 }%
4519 {%
4520   \newcommand*{\glsdisplaynumberlist}[1]{%
4521     \glsdoifexists{#1}%
4522     {%
4523       \bgroup
4524
4525       \edef\@glo@label{\glsdetoklabel{#1}}%
4526       \let\@org@glsglsnumberformat\glsglsnumberformat
4527       \def\glsglsnumberformat##1{##1}%
4528       \protected@edef\the@numberlist{%
4529         \csname glo@\@glo@label @numberlist\endcsname}%
4530       \def\@gls@numlist@sep{}%
4531       \def\@gls@numlist@nextsep{}%
4532       \def\@gls@numlist@lastsep{}%
4533       \def\@gls@thislist{}%
4534       \def\@gls@donext@def{}%
4535       \renewcommand\do[1]{%
4536         \protected@edef\@gls@thislist{%
4537           \@gls@thislist
```

```

4537         \noexpand\@gls@numlist@sep
4538         ##1%
4539     }%
4540     \let\@gls@numlist@sep\@gls@numlist@nextsep
4541     \def\@gls@numlist@nextsep{\glsnumlistsep}%
4542     \@gls@donext@def
4543     \def\@gls@donext@def{%
4544         \def\@gls@numlist@lastsep{\glsnumlistlastsep}%
4545     }%
4546 }%
4547 \expandafter \glsnumlistparser \expandafter{\the@numberlist}%
4548 \let\@gls@numlist@sep\@gls@numlist@lastsep
4549 \@gls@thislist
4550 \egroup
4551 }%
4552 }
4553 }

```

`\glsnumlistsep`

```

4554 \newcommand*{\glsnumlistsep}{, }

```

`\glsnumlistlastsep`

```

4555 \newcommand*{\glsnumlistlastsep}{ \& }

```

`\gls hyperlink` Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\gls link` or `\gls add` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```

4556 \newcommand*{\gls hyperlink}[2][\glsentrytext{\@glo@label}]{%
4557   \def\@glo@label{#2}%
4558   \@gls link{\glo link prefix\glsdetoklabel{#2}}{#1}}

```

## 1.12 Adding an entry to the glossary without generating text

The following keys are provided for `\gls add` and `\gls add all`:

```

4559 \define@key{gloss add}{counter}{\def\@gls@counter{#1}}
4560 \define@key{gloss add}{format}{\def\@gls@number format{#1}}

```

This key is only used by `\gls add all`:

```

4561 \define@key{gloss add}{types}{\def\@glo@type{#1}}

```

`\gls add[<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`

```
4562 \newrobustcmd*{\glsadd}[2] [] {%
```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```
4563 \@gls@adjustmode
```

```
4564 \glsdoifexists{#2}%
```

```
4565 {%
```

```
4566 \def\@glsnumberformat{glsnumberformat}%
```

```
4567 \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
```

```
4568 \setkeys{glossadd}{#1}%
```

Store the entry's counter in `\theglsentrycounter`

```
4569 \@gls@saveentrycounter
```

This should use `\@@do@wrglossary` rather than `\do@wrglossary` since the whole point of `\glsadd` is to add a line to the glossary.

```
4570 \@@do@wrglossary{#2}%
```

```
4571 }%
```

```
4572 }
```

`@gls@adjustmode`

```
4573 \newcommand*{\@gls@adjustmode}{}%
```

```
4574 \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`

```
4575 \newrobustcmd*{\glsaddall}[1] [] {%
```

```
4576 \edef\@glo@type{\@glo@types}%
```

```
4577 \setkeys{glossadd}{#1}%
```

```
4578 \forallglsentries[\@glo@type]{\@glo@entry}{%
```

```
4579 \glsadd[#1]{\@glo@entry}%
```

```
4580 }%
```

```
4581 }
```

`\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.

```
4582 \newrobustcmd*{\glsaddallunused}[1] [\@glo@types] {%
```

```
4583 \forallglsentries[#1]{\@glo@entry}%
```

```
4584 {%
```

```
4585 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
```

```
4586 }%
4587 }
```

`\glsignore`

```
4588 \newcommand*{\glsignore}[1]{}
```

## 1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The makeindex actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about makeindex special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
4589 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
4590 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsbackslash` Define `\glsbackslash` to make it easier to write a backslash to a file.

```
4591 \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.

```
4592 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4593 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4594 \edef\glstildechar{\string~}
```

`@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for `xindy`.

```
4595 \ifglsxindy
```

```
4596 \newcommand*{\@glsfirstletter}{A}
```

```
4597 \fi
```

letterAfterDigits Sets the first letter to come after the digits 0,...,9.

```

4598 \ifglsxindy
4599   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4600     \renewcommand*{\@glsfirstletter}{#1}}
4601 \else
4602   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4603     \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4604 \fi

```

\@glsminrange Define the minimum number of successive location references to merge into a range.

```

4605 \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.

```

4606 \ifglsxindy
4607   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4608     \renewcommand*{\@glsminrange}{#1}}
4609 \else
4610   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4611     \glsnoxywarning\GlsSetXdyMinRangeLength}
4612 \fi

```

\writeist

```

4613 \ifglsxindy
    Code to use if xindy is required.
4614   \def\writeist{%
    Define write register if not already defined
4615     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4616     \@gls@addpredefinedattributes
    Open the file.
4617     \openout\glswrite=\istfilename
    Write header comment at the start of the file
4618     \write\glswrite{;; xindy style file created by the glossaries
4619       package}%
4620     \write\glswrite{;; for document '\jobname' on
4621       \the\year-\the\month-\the\day}%
    Specify the required styles
4622     \write\glswrite{^^J; required styles^^J}
4623     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4624       \ifx\@xdystyle\@empty
4625       \else
4626         \protected@write\glswrite{{(require
4627           \string"\@xdystyle.xdy\string")}}%
4628       \fi
4629     }%

```

List the allowed attributes (possible values used by the format key)

```
4630 \write\glswrite{^^J%
4631 ; list of allowed attributes (number formats)^^J}%
4632 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4633 \write\glswrite{^^J; user defined alphabets^^J}%
4634 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4635 \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.

```
4636 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case were  $\langle Hprefix \rangle$  is empty:

```
4637 \protected@write\glswrite{}\{(define-location-class
4638 \string"\@gls@classI\string"^^J\space\space\space
4639 (
4640 :sep "{"{"
4641 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4642 :sep "}"
4643 )
4644 ^^J\space\space\space
4645 :min-range-length \@glsminrange^^J%
4646 )
4647 }%
```

Nested iteration over all classes:

```
4648 {%
4649 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4650 \protected@write\glswrite{}\{(define-location-class
4651 \string"\@gls@classII-\@gls@classI\string"
4652 ^^J\space\space\space
4653 (
4654 :sep "{"
4655 \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4656 :sep "}"{"
4657 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4658 :sep "}"
4659 )
4660 ^^J\space\space\space
4661 :min-range-length \@glsminrange^^J%
4662 )
4663 }%
4664 }%
4665 }%
4666 }%
```

User defined location classes (needs checking for new location format).

```

4667 \write\glswrite{^^J; user defined location classes}%
4668 \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.)

```

4669 \write\glswrite{^^J; define cross-reference class^^J}%
4670 \write\glswrite{(define-crossref-class \string"see\string"
4671 :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.)

```

4672 \write\glswrite{(markup-crossref-list
4673 :class \string"see\string"^^J\space\space\space
4674 :open \string"\string\glsseeformat\string"
4675 :close \string"{\string")}%

```

Provide hook to write extra material here (used by glossaries-extra to define a seealso class).

```

4676 \@xdycrossrefhook

```

List the order to sort the classes.

```

4677 \write\glswrite{^^J; define the order of the location classes}%
4678 \write\glswrite{(define-location-class-order
4679 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4680 \write\glswrite{^^J; define the glossary markup^^J}%

4681 \write\glswrite{(markup-index^^J\space\space\space
4682 :open \string"\string
4683 \glossarysection[\string\glossarytoctitle]{\string
4684 \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)

```

4685 \@for\@this@ctr:=\@xdycounters\do{%
4686   {%
4687     \@for\@this@attr:=\@xdyattributelist\do{%
4688       \protected@write\glswrite{}\{\string\providecommand*%
4689         \expandafter\string
4690         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4691         {%
4692           \string\setentrycounter
4693           [\expandafter\@gobble\string\#1]{\@this@ctr}%
4694           \expandafter\string
4695           \csname\@this@attr\endcsname
4696           {\expandafter\@gobble\string\#2}%
4697         }%
4698       }%

```



```

4699     }%
4700 }%
4701 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4702 \write\glswrite{%
4703   \string\begin
4704   {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4705   \space\space:close \string"\glpercentchar\glstildechar n\string
4706   \end{theglossary}\string\glossarypostamble
4707   \glstildechar n\string" ^^J\space\space\space
4708   :tree)}}%

```

Specify what to put between letter groups

```

4709 \write\glswrite{(markup-letter-group-list
4710   :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4711 \write\glswrite{(markup-indexentry
4712   :open \string"\string\relax \string\glresetentrylist
4713   \glstildechar n\string")}%

```

Specify how to format entries

```

4714 \write\glswrite{(markup-locclass-list :open
4715   \string"\glsoopenbrace\string\glossaryentrynumbers
4716   \glsoopenbrace\string\relax\space \string"^^J\space\space\space
4717   :sep \string", \string"
4718   :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

4719 \write\glswrite{(markup-locref-list
4720   :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

4721 \write\glswrite{(markup-range
4722   :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.

```

4723 \@onelevel@sanitize\gls@suffiF
4724 \@onelevel@sanitize\gls@suffiFF
4725 \ifx\gls@suffiF\@empty
4726 \else
4727   \write\glswrite{(markup-range
4728     :close "\gls@suffiF" :length 1 :ignore-end)}}%
4729 \fi
4730 \ifx\gls@suffiFF\@empty
4731 \else
4732   \write\glswrite{(markup-range
4733     :close "\gls@suffiFF" :length 2 :ignore-end)}}%
4734 \fi

```

Specify how to format locations.

```
4735 \write\glswrite{^^J; define format to use for locations^^J}%  
4736 \write\glswrite{\@xdylocref}%
```

Specify how to separate letter groups.

```
4737 \write\glswrite{^^J; define letter group list format^^J}%  
4738 \write\glswrite{(markup-letter-group-list  
4739 :sep \string"\string\glsgroupskip\glstildechar n\string")}%
```

Define letter group headings.

```
4740 \write\glswrite{^^J; letter group headings^^J}%  
4741 \write\glswrite{(markup-letter-group  
4742 :open-head \string"\string\glsgroupheading  
4743 \glsoopenbrace\string"^^J\space\space\space  
4744 :close-head \string"\glsclosebrace\string")}%
```

Define additional letter groups.

```
4745 \write\glswrite{^^J; additional letter groups^^J}%  
4746 \write\glswrite{\@xdylettergroups}%
```

Define additional sort rules

```
4747 \write\glswrite{^^J; additional sort rules^^J}  
4748 \write\glswrite{\@xdysortrules}%
```

Hook for any additional information:

```
4749 \@gls@writeisthook
```

Close the style file

```
4750 \closeout\glswrite
```

Suppress any further calls.

```
4751 \let\writeist\relax  
4752 }  
4753 \else
```

Code to use if makeindex is required.

```
4754 \edef\@gls@actualchar{\string?}  
4755 \edef\@gls@encapchar{\string!}  
4756 \edef\@gls@levelchar{\string!}  
4757 \edef\@gls@quotechar{\string"}%  
4758 \let\GlsSetQuote\gls@nosetquote  
4759 \def\writeist{\relax  
4760 \ifundef{\glswrite}{\newwrite\glswrite}{\relax  
4761 \openout\glswrite=\istfilename  
4762 \write\glswrite{\glspersentchar\space makeindex style file  
4763 created by the glossaries package}  
4764 \write\glswrite{\glspersentchar\space for document  
4765 '\jobname' on \the\year-\the\month-\the\day}  
4766 \write\glswrite{actual '\@gls@actualchar'}  
4767 \write\glswrite{encap '\@gls@encapchar'}  
4768 \write\glswrite{level '\@gls@levelchar'}  
4769 \write\glswrite{quote '\@gls@quotechar'}}
```

```

4770 \write\glswrite{keyword \string\string\glossaryentry\string}
4771 \write\glswrite{preamble \string\string\glossarysection[\string
4772 \glossarytoctitle]{\string\glossarytitle}\string
4773 \glossarypreamble\string\n\string\begin{theglossary}\string
4774 \glossaryheader\string\n\string}
4775 \write\glswrite{postamble \string\string%\string\n\string
4776 \end{theglossary}\string\glossarypostamble\string\n
4777 \string}
4778 \write\glswrite{group_skip \string\string\glsgroupskip\string\n
4779 \string}
4780 \write\glswrite{item_0 \string\string%\string\n\string}
4781 \write\glswrite{item_1 \string\string%\string\n\string}
4782 \write\glswrite{item_2 \string\string%\string\n\string}
4783 \write\glswrite{item_01 \string\string%\string\n\string}
4784 \write\glswrite{item_x1
4785 \string\string\relax \string\glssresetentrylist\string\n
4786 \string}
4787 \write\glswrite{item_12 \string\string%\string\n\string}
4788 \write\glswrite{item_x2
4789 \string\string\relax \string\glssresetentrylist\string\n
4790 \string}

4791 \write\glswrite{delim_0 \string\string{\string
4792 \glossaryentrynumbers\string{\string\relax \string}
4793 \write\glswrite{delim_1 \string\string{\string
4794 \glossaryentrynumbers\string{\string\relax \string}
4795 \write\glswrite{delim_2 \string\string{\string
4796 \glossaryentrynumbers\string{\string\relax \string}
4797 \write\glswrite{delim_t \string\string}\string}\string}
4798 \write\glswrite{delim_n \string\string\delimN \string}
4799 \write\glswrite{delim_r \string\string\delimR \string}
4800 \write\glswrite{headings_flag 1}
4801 \write\glswrite{heading_prefix
4802 \string\string\glsgroupheading\string{\string}
4803 \write\glswrite{heading_suffix
4804 \string\string}\string\relax
4805 \string\glssresetentrylist \string}
4806 \write\glswrite{symhead_positive \string\glssymbols\string}
4807 \write\glswrite{numhead_positive \string\glssnumbers\string}
4808 \write\glswrite{page_compositor \string\glsscompositor\string}
4809 \@glscbsdq\glscbsdsuffixF
4810 \@glscbsdq\glscbsdsuffixFF
4811 \ifx\glscbsdsuffixF\@empty
4812 \else
4813 \write\glswrite{suffix_2p \string\glscbsdsuffixF\string}
4814 \fi
4815 \ifx\glscbsdsuffixFF\@empty
4816 \else
4817 \write\glswrite{suffix_3p \string\glscbsdsuffixFF\string}
4818 \fi

```

Hook for any additional information:

```
4819 \gls@writeisthook
```

Close the file and disable \writeist.

```
4820 \closeout\glswrite
4821 \let\writeist\relax
4822 }
4823 \fi
```

**SetWriteIstHook** Allow user to append information to the style file.

```
4824 \newcommand*{\GlsSetWriteIstHook}[1]{\renewcommand*{\gls@writeisthook}{#1}}
4825 \@onlypremakeg\GlsSetWriteIstHook
```

**ls@writeisthook**

```
4826 \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.

```
4827 \ifglxindy
4828 \newcommand*{\GlsSetQuote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4829 \newcommand*{\gls@nosetquote}[1]{\glsnomakeindexwarning\GlsSetQuote}
4830 \else
4831 \newcommand*{\GlsSetQuote}[1]{\edef\gls@quotechar{\string#1}%
```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```
4832 \ifpackageloaded{tracklang}%
4833 {%
4834 \IfTrackedLanguage{german}%
4835 {%
4836 \def\@gls@extramakeindexopts{-g}%
4837 }%
4838 }%
4839 }%
4840 {}%
```

Need to redefine \gls@checkquote

```
4841 \edef\gls@docheckquotedef{%
4842 \noexpand\def\noexpand\gls@checkquote####1#1####2#1####3\noexpand\null{%
4843 \noexpand\gls@tmpb=\noexpand\expandafter\noexpand\gls@checkedmkidx}%
4844 \noexpand\toks@={####1}%
4845 \noexpand\ifx\noexpand\null####2\noexpand\null
4846 \noexpand\ifx\noexpand\null####3\noexpand\null
4847 \noexpand\edef\noexpand\gls@checkedmkidx{%
4848 \noexpand\the\noexpand\gls@tmpb\noexpand\the\noexpand\toks@}%
4849 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
4850 \noexpand\else
4851 \noexpand\edef\noexpand\gls@checkedmkidx{%
4852 \noexpand\the\noexpand\gls@tmpb\noexpand\the\noexpand\toks@
4853 \noexpand\gls@quotechar\noexpand\gls@quotechar
4854 \noexpand\gls@quotechar\noexpand\gls@quotechar}%
```

```

4855     \noexpand\def\noexpand\@@gls@checkquote{%
4856         \noexpand\@gls@checkquote####3\noexpand\null}%
4857     \noexpand\fi
4858 \noexpand\else
4859     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4860         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4861         \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4862     \noexpand\ifx\noexpand\null####3\noexpand\null
4863     \noexpand\def\noexpand\@@gls@checkquote{%
4864         \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
4865     \noexpand\else
4866     \noexpand\def\noexpand\@@gls@checkquote{%
4867         \noexpand\@gls@checkquote####2#1####3\noexpand\null}%
4868     \noexpand\fi
4869 \noexpand\fi
4870 \noexpand\@@gls@checkquote
4871 }%
4872 }%
4873 \@gls@docheckquotedef
4874 \edef\@gls@docheckquotedef{%
4875     \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
4876         \noexpand\def\noexpand\@gls@checkedmkidx{%
4877             \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
4878             #1#1\noexpand\null
4879             \noexpand\expandafter\noexpand\@gls@updatechecked
4880             \noexpand\@gls@checkedmkidx{####1}%
4881             \noexpand\def\noexpand\@gls@checkedmkidx{%
4882                 \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
4883                 \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
4884                 \noexpand\null
4885                 \noexpand\expandafter\noexpand\@gls@updatechecked
4886                 \noexpand\@gls@checkedmkidx{####1}%
4887                 \noexpand\def\noexpand\@gls@checkedmkidx{%
4888                     \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
4889                     \noexpand\?\noexpand\?\noexpand\null
4890                     \noexpand\expandafter\noexpand\@gls@updatechecked
4891                     \noexpand\@gls@checkedmkidx{####1}%
4892                     \noexpand\def\noexpand\@gls@checkedmkidx{%
4893                         \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
4894                         \noexpand?\noexpand?\noexpand\null
4895                         \noexpand\expandafter\noexpand\@gls@updatechecked
4896                         \noexpand\@gls@checkedmkidx{####1}%
4897                         \noexpand\def\noexpand\@gls@checkedmkidx{%
4898                             \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
4899                             \noexpand|\noexpand|\noexpand\null
4900                             \noexpand\expandafter\noexpand\@gls@updatechecked
4901                             \noexpand\@gls@checkedmkidx{####1}%
4902                             \noexpand\def\noexpand\@gls@checkedmkidx{%
4903                                 \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil

```

```

4904         \noexpand\|\noexpand\|\noexpand\null
4905     \noexpand\expandafter\noexpand\@gls@updatechecked
4906         \noexpand\@gls@checkedmkidx{####1}%
4907     \noexpand\def\noexpand\@gls@checkedmkidx{%
4908     \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
4909         \noexpand!\noexpand!\noexpand\null
4910     \noexpand\expandafter\noexpand\@gls@updatechecked
4911         \noexpand\@gls@checkedmkidx{####1}%
4912 }%
4913 }%
4914 \@gls@docheckquotedef
4915 \edef\@gls@docheckquotedef{%
4916     \noexpand\def\noexpand\@gls@checkescquote####1%
4917         \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
4918     ####3\noexpand\null{%
4919         \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4920         \noexpand\toks@={####1}%
4921     \noexpand\ifx\noexpand\null####2\noexpand\null
4922     \noexpand\ifx\noexpand\null####3\noexpand\null
4923         \noexpand\edef\noexpand\@gls@checkedmkidx{%
4924             \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4925     \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
4926     \noexpand\else
4927     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4928         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4929         \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4930             \csname#1\endcsname}\noexpand\@gls@quotechar
4931         \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4932             \csname#1\endcsname}\noexpand\@gls@quotechar}%
4933     \noexpand\def\noexpand\@gls@checkescquote{%
4934         \noexpand\@gls@checkescquote####3\noexpand\null}%
4935     \noexpand\fi
4936     \noexpand\else
4937     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4938         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4939         \noexpand\@gls@quotechar\noexpand\string
4940         \expandonce{\csname#1\endcsname}\noexpand\@gls@quotechar}%
4941     \noexpand\ifx\noexpand\null####3\noexpand\null
4942     \noexpand\def\noexpand\@gls@checkescquote{%
4943         \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4944         \expandonce{\csname#1\endcsname}\noexpand\null}%
4945     \noexpand\else
4946     \noexpand\def\noexpand\@gls@checkescquote{%
4947         \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4948         ####3\noexpand\null}%
4949     \noexpand\fi
4950     \noexpand\fi
4951     \noexpand\@gls@checkescquote
4952 }%

```

```

4953 }%
4954 \@gls@docheckquotedef
4955 }
4956 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
4957 {\string\GlsSetQuote\space not permitted here}%
4958 {Move \string\GlsSetQuote\space earlier in the preamble, as
4959 soon as possible after glossaries.sty has been loaded}}
4960 \fi

```

ramakeindexopts

```

4961 \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`

```

4962 \newcommand{\noist}{%
  Update attributes list
4963 \@gls@addpredefinedattributes
4964 \let\writeist\relax
4965 }

```

`\@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`

```

4966 \newcommand*{\@makeglossary}[1]{%
4967 \ifglossaryexists{#1}%
4968 {%

```

Only create a new write if `savewrites=false` otherwise create a token to collect the information.

```

4969 \ifglssavewrites
4970 \expandafter\newtoks\csname glo@#1@filetok\endcsname
4971 \else
4972 \expandafter\newwrite\csname glo@#1@file\endcsname
4973 \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4974 \fi
4975 \@gls@renewglossary
4976 \writeist
4977 }%

```

```

4978 {%
4979   \PackageError{glossaries}%
4980   {Glossary type ‘#1’ not defined}%
4981   {New glossaries must be defined before using \string\makeglossary}%
4982 }%
4983 }

```

`\@glsopenfile` Open write file associated with the given glossary.

```

4984 \newcommand*{\@glsopenfile}[2]{%
4985   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4986   \PackageInfo{glossaries}{Writing glossary file
4987     \jobname.\csname @glotype@#2@out\endcsname}%
4988 }

```

`\@closegls`

```

4989 \newcommand*{\@closegls}[1]{%
4990   \closeout\csname glo@#1@file\endcsname
4991 }

```

`\@gls@automake`

```

4992 \ifglxindy
4993 \newcommand*{\@gls@automake}[1]{%
4994   \ifglossaryexists{#1}
4995   {%
4996     \@closegls{#1}%
4997     \ifdefstring{\glsorder}{letter}%
4998     {\def\@gls@order{-M ord/letorder }}%
4999     {\let\@gls@order\@empty}%
5000     \ifcsundef{\xdy@#1@language}%
5001     {\let\@gls@langmod\@xdy@main@language}%
5002     {\letcs\@gls@langmod{\xdy@#1@language}}%
5003     \edef\@gls@dothiswrite{\noexpand\write18{xindy
5004       -I xindy
5005       \@gls@order
5006       -L \@gls@langmod\space
5007       -M \@gls@istfilebase\space
5008       -C \@gls@codepage\space
5009       -t \jobname.\csuse{@glotype@#1@log}
5010       -o \jobname.\csuse{@glotype@#1@in}
5011       \jobname.\csuse{@glotype@#1@out}}}%
5012     }%
5013     \@gls@dothiswrite
5014   }%
5015   {%
5016     \GlossariesWarning{Can't make glossary ‘#1’, it doesn't exist}%
5017   }%
5018 }
5019 \else
5020 \newcommand*{\@gls@automake}[1]{%

```



```

5021 \ifglossaryexists{#1}
5022 {%
5023   \@closegls{#1}%
5024   \ifdefstring{\glsorder}{letter}%
5025     {\def\@gls@order{-l }}%
5026     {\let\@gls@order\@empty}%
5027   \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5028     -s \istfilename\space
5029     -t \jobname.\csuse{\glotype@#1@log}
5030     -o \jobname.\csuse{\glotype@#1@in}
5031     \jobname.\csuse{\glotype@#1@out}}}%
5032   }%
5033   \@gls@dothiswrite
5034   }%
5035   {%
5036     \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5037   }%
5038 }
5039 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

5040 \newcommand*{\@warn@nomakeglossaries}{}
    Only use this if warning if \printglossary has been used without \makeglossaries
5041 \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`

```

5042 \newcommand*{\makeglossaries}{%
    Define the write used for style file also used for all other output files if savewrites=true.
5043   \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.
5044   \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{} }
5045   \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{} }
    If \@gls@extramakeindexopts has been defined, write it:
5046   \ifundef\@gls@extramakeindexopts
5047   {}%
5048   {%
5049     \protected@write\@auxout{}{\string\providecommand
5050       \string\@gls@extramakeindexopts[1]{} }
5051     \protected@write\@auxout{}{\string\@gls@extramakeindexopts
5052       {\@gls@extramakeindexopts}}%
5053   }%

```

Write the name of the style file to the aux file (needed by makeglossaries)

```
5054 \protected@write\@auxout{}\string\istfilename{\istfilename}}%
5055 \protected@write\@auxout{}\string\glsorder{\glsorder}}
```

Iterate through each glossary type and activate it.

```
5056 \@for\@glo@type:=\@glo@types\do{%
5057   \ifthenelse{\equal{\@glo@type}{}}{}{}%
5058   \@makeglossary{\@glo@type}}%
5059 }%
```

New glossaries must be created before \makeglossaries so disable \newglossary.

```
5060 \renewcommand*\newglossary[4][]{%
5061   \PackageError{glossaries}{New glossaries
5062   must be created before \string\makeglossaries}{You need
5063   to move \string\makeglossaries\space after all your
5064   \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
5065 \let\@makeglossary\relax
5066 \let\makeglossary\relax
5067 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
5068 \@disable@onlypremakeg
```

Allow see key:

```
5069 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
5070 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
5071 \def\warn@noprntglossary{%
5072   \ifdefstring{\@glo@types}{,}%
5073   {%
5074     \GlossariesWarningNoLine{No glossaries have been defined}%
5075   }%
5076   {%
5077     \GlossariesWarningNoLine{No \string\printglossary\space
5078     or \string\printglossaries\space
5079     found. ^^J(Remove \string\makeglossaries\space if you
5080     don't want any glossaries.) ^^JThis document will not
5081     have a glossary}%
5082   }%
5083 }%
```

Declare list parser for \glsdisplaynumberlist

```
5084 \ifglssavenumberlist
5085   \edef\@gls@doddeflistparser{\noexpand\DeclareListParser
5086     {\noexpand\glsnumlistparser}{\delimN}}%
5087   \@gls@doddeflistparser
5088 \fi
```

Prevent user from also using \makenoidxglossaries

```
5089 \let\makenoidxglossaries\@no@makeglossaries
```

Prohibit sort key in printgloss family:

```
5090 \renewcommand*{\@printgloss@setsort}{%
5091 \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5092 }%
```

Check the automake setting:

```
5093 \ifglsautomake
5094 \renewcommand*{\@gls@doautomake}{%
5095 \@for\@gls@type:=\@glo@types\do{%
5096 \ifdefempty{\@gls@type}{}%
5097 {\@gls@automake{\@gls@type}}%
5098 }%
5099 }%
5100 \fi
```

Check the sort setting:

```
5101 \@glo@check@sortallowed\makeglossaries
5102 }
```

Must occur in the preamble:

```
5103 \@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

```
5104 \let\makeglossary\makeglossaries
```

If \makeglossaries hasn't been used, issue a warning. Also issue a warning if neither \printglossaries nor \printglossary have been used.

```
5105 \AtEndDocument{%
5106 \warn@nomakeglossaries
5107 \warn@noprintglossary
5108 }
```

noidxglossaries Analogous to \makeglossaries this activates the commands needed for \printnoidxglossary

```
5109 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5110 \renewcommand{\@gls@noref@warn}[1]{%
5111 \GlossariesWarning{Empty glossary for
5112 \string\printnoidxglossary[type={##1}].
5113 Rerun may be required (or you may have forgotten to use
5114 commands like \string\gls)}}%
5115 }%
```

Don't escape makeindex/xindy characters

```
5116 \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
5117 \let\@do@@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5118 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5119 \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5120 \renewcommand{\@do@seeglossary}[2]{%
5121   \edef\@gls@label{\glsdetoklabel{##1}}%
5122   \protected@write\@auxout{}{%
5123     \string\@gls@reference
5124     {\csname glo@\@gls@label @type\endcsname}%
5125     {\@gls@label}%
5126     {%
5127       \string\glsseeformat##2}%
5128     }%
5129   }%
5130 }%
```

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.

```
5131 \AtBeginDocument
5132 {%
5133   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5134 }%
```

Change warning about no glossaries

```
5135 \def\warn@noprintglossary{%
5136   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5137     or \string\printnoidxglossaries ^^J
5138     found. (Remove \string\makenoidxglossaries\space if you
5139     don't want any glossaries.)^^JThis document will not have a glossary}%
5140 }%
```

Suppress warning about no \makeglossaries

```
5141 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5142 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5143 \renewcommand*{\@printgloss@setsort}{%
5144   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
5145   \def\@glo@sorttype{\@glo@default@sorttype}%
5146 }%
```

All entries must be defined in the preamble:

```

5147 \renewcommand*\new@glossaryentry[2]{%
5148   \PackageError{glossaries}{Glossary entries must be
5149     defined in the preamble^^Jwhen you use
5150     \string\makenoidxglossaries}%
5151   {Either move your definitions to the preamble or use
5152     \string\makeglossaries}%
5153 }%

  Redefine \glsentrynumberlist
5154 \renewcommand*\glsentrynumberlist[1]{%
5155   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5156   \ifdef\@gls@loclist
5157   {%
5158     \glsnoidxloclist{\@gls@loclist}%
5159   }%
5160   {%
5161     ??\glsdoifexists{##1}%
5162     {%
5163       \GlossariesWarning{Missing location list for ‘##1’. Either
5164         a rerun is required or you haven’t referenced the entry}%
5165     }%
5166   }%
5167 }%

  Redefine \glsdisplaynumberlist
5168 \renewcommand*\glsdisplaynumberlist[1]{%
5169   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5170   \ifdef\@gls@loclist
5171   {%
5172     \def\@gls@noidxloclist@sep{%
5173       \def\@gls@noidxloclist@sep{%
5174         \def\@gls@noidxloclist@sep{%
5175           \glsnumlistsep
5176         }%
5177       \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5178     }%
5179   }%
5180   \def\@gls@noidxloclist@finalsep{}%
5181   \def\@gls@noidxloclist@prev{}%
5182   \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5183   \@gls@noidxloclist@finalsep
5184   \@gls@noidxloclist@prev
5185 }%
5186 {%
5187   ??\glsdoifexists{##1}%
5188   {%
5189     \GlossariesWarning{Missing location list for ‘##1’. Either
5190       a rerun is required or you haven’t referenced the entry}%
5191   }%

```

```

5192 }%
5193 }%

```

Provide a generic way of iterating through the number list:

```

5194 \renewcommand*{\glsnumberlistloop}[3]{%
5195   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5196   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5197   \let\@gls@org@glsseeformat\glsseeformat
5198   \let\glsnoidxdisplayloc##2\relax
5199   \let\glsseeformat##3\relax
5200   \ifdef\@gls@loclist
5201   {%
5202     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5203   }%
5204   {%
5205     ??\glsdoifexists{##1}%
5206     {%
5207       \GlossariesWarning{Missing location list for ‘##1’. Either
5208         a rerun is required or you haven’t referenced the entry}%
5209     }%
5210   }%
5211   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5212   \let\glsseeformat\@gls@org@glsseeformat
5213 }%

```

Modify sanitize sort function

```

5214 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5215 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5216 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5217 \@glo@check@sortallowed\makenoidxglossaries
5218 }

```

Preamble-only command:

```

5219 \@onlypreamble{\makenoidxglossaries}

```

```

\glsnumberlistloop

```

```

\glsnumberlistloop{<label>}{<handler>}

```

```

5220 \newcommand*{\glsnumberlistloop}[2]{%
5221   \PackageError{glossaries}{\string\glsnumberlistloop\space
5222     only works with \string\makenoidxglossaries}{}%
5223 }

```

`\listloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc {<prefix>}{<counter>}{<format>}{<n>}`)

```

5224 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5225   #1%
5226 }

```

```

@makeglossaries Can't use both \makeglossaries and \makenoidxglossaries
5227 \newcommand*{\@no@makeglossaries}{%
5228   \PackageError{glossaries}{You can't use both
5229   \string\makeglossaries\space and \string\makenoidxglossaries}%
5230   {Either use one or other (or none) of those commands but not both
5231   together.}%
5232 }

@gl@s@noref@warn Warning when no instances of \@gl@s@reference found.
5233 \newcommand{\@gl@s@noref@warn}[1]{%
5234   \GlossariesWarning{\string\makenoidxglossaries\space
5235   is required to make \string\printnoidxglossary[type={#1}] work}%
5236 }

s@noidxglossary Write the glossary information to the aux file:
5237 \newcommand*{\gl@s@noidxglossary}{%
5238   \protected@write\@auxout{}{%
5239     \string\@gl@s@reference
5240     {\csname glo@\@gl@s@label @type\endcsname}%
5241     {\@gl@s@label}%
5242     {\string\gl@s@noidxdisplayloc
5243     {\@glo@counterprefix}%
5244     {\@gl@s@counter}%
5245     {\@gl@s@numberformat}%
5246     {\@gl@s@locref}%
5247     }%
5248   }%
5249 }

```

## 1.14 Writing information to associated files

```

\istfile Deprecated.
5250 \def\istfile{\glswrite}

```

At the end of the document, the files should be created if savewrites=true.

```

5251 \AtEndDocument{%
5252   \glswritefiles
5253 }

```

```

\@glswritefiles Only write the files if savewrites=true
5254 \newcommand*{\@glswritefiles}{%
  Iterate through all the glossaries
5255   \forallglossaries{\@glo@type}{%
    Check for empty glossaries (patch provided by Patrick Häcker)
5256     \ifcsundef{glo@\@glo@type @filetok}%
5257     {%

```

```

5258     \def\gls@tmp{}%
5259 }%
5260 {%
5261     \edef\gls@tmp{\expandafter\the
5262         \csname glo@\@glo@type @filetok\endcsname}%
5263 }%
5264 \ifx\gls@tmp\@empty
5265     \ifx\@glo@type\glsdefaulttype
5266         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5267             entries.^^JRemember to use package option ‘nomain’ if
5268 you
5269             don’t want to^^Juse the main glossary}%
5270     \else
5271         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
5272             entries}%
5273     \fi
5274 \else
5275     \@glsopenfile{\glswrite}{\@glo@type}%
5276     \immediate\write\glswrite{%
5277         \expandafter\the
5278         \csname glo@\@glo@type @filetok\endcsname}%
5279     \immediate\closeout\glswrite
5280 \fi
5281 }%
5282 }

```

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`

```

5283 \if@gls@docloaded
5284 \else
5285     \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
5286 \fi

```

The associated number should be stored in `\theglentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```

5287 \newcommand*{\gls@glossary}[1]{%
5288     \@gls@glossary{#1}%
5289 }

```



`\@gls@glossary` (In v4.10, `\@glossary` was redefined to `\@gls@glossary` to avoid conflict with other packages.) Define internal `\@gls@glossary` to ignore its argument. This gets redefined in `\@makeglossary`. This is defined to just `\index` as memoir changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.) The argument #1 is the glossary type.

```
5290 \newcommand*{\@gls@glossary}[2]{%
5291   \if@gls@debug
5292     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5293   \fi
5294   \index{#2}%
5295 }
```

This is a convenience command to set `\@gls@glossary`. It's used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`s@renewglossary`

```
5296 \newcommand{\@gls@renewglossary}{%
5297   \gdef\@gls@glossary##1{\@bsphack\begin@group\gls@wrglossary{##1}}%
5298   \let\@gls@renewglossary\@empty
5299 }
```

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`

```
5300 \newcommand*{\gls@wrglossary}[2]{%
5301   \ifglssavewrites
5302     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5303     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5304       \expandafter{\@gls@tmp^^J}%
5305   \else
5306     \ifcsdef{glo@#1@file}%
5307     {%
5308       \expandafter\protected@write\csname glo@#1@file\endcsname{%
5309         \gls@disablepagerefexpansion}{#2}%
5310     }%
5311     {%
5312       \ifignoredglossary{#1}{}%
5313       {%
5314         \GlossariesWarning{No file defined for glossary '#1'}%
5315       }%
5316     }%
5317   \fi
5318   \endgroup\@esphack
5319 }
```

`\@do@wrglossary`

```

5320 \newcommand*{\@do@wrglossary}[1]{%
5321   \glswriteentry{#1}{\@do@wrglossary{#1}}%
5322 }

```

`\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.

```

5323 \newcommand*{\glswriteentry}[2]{%
5324   \ifglsindexonlyfirst
5325     \ifglsused{#1}{#2}%
5326   \else
5327     #2%
5328   \fi
5329 }

```

`protected@pagefmts` List of page formats to be protected against expansion.

```

5330 \newcommand{\gls@protected@pagefmts}{%
5331   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5332 }

```

`pagerefexpansion`

```

5333 \newcommand*{\gls@disablepagerefexpansion}{%
5334   \@for\@gls@this:=\gls@protected@pagefmts\do
5335   {%
5336     \expandafter\let\@gls@this\relax
5337   }%
5338 }

```

`\gls@alphpage`

```

5339 \newcommand*{\gls@alphpage}{\@alph\c@page}

```

`\gls@Alphpage`

```

5340 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

```

`\gls@numberpage`

```

5341 \newcommand*{\gls@numberpage}{\number\c@page}

```

`\gls@arabicpage`

```

5342 \newcommand*{\gls@arabicpage}{\@arabic\c@page}

```

`\gls@romanpage`

```

5343 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

```

`\gls@Romanpage`

```

5344 \newcommand*{\gls@Romanpage}{\@Roman\c@page}

```

protectedpagefmt

`\glsaddprotectedpagefmt{<cs name>}`

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a TeX register as the argument (`\<csname>\c@page` must be valid).

```
5345 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5346   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
5347   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5348   \eappto\wrglossarynumberhook{%
5349     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5350     \expandonce{\csname#1\endcsname}%
5351     \noexpand\def\expandonce{\csname#1\endcsname}{%
5352       \noexpand\@wrglossary@pageformat
5353       \expandonce{\csname gls#1page\endcsname}%
5354       \expandonce{\csname org@gls#1\endcsname}%
5355     }%
5356   }%
5357 }
```

ssarynumberhook Hook used by `\@do@wrglossary`

```
5358 \newcommand*\@wrglossarynumberhook{}
```

sary@pageformat

```
5359 \newcommand{\@wrglossary@pageformat}[3]{%
5360   \ifx#3\c@page #1\else #2#3\fi
5361 }
```

owprimitivemods Conditional to determine whether or not `\@do@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
5362 \newif\ifglswrallowprimitivemods
5363 \glswrallowprimitivemodstrue
```

@do@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.

```
5364 \newcommand*{\@do@wrglossary}[1]{%
5365   \begingroup
```

First a bit of hackery to prevent premature expansion of `\c@page`. Store original definitions:

```
5366   \let\orgthe\the
5367   \let\orgnumber\number

5368   \let\orgarabic\@arabic
5369   \let\orgromannumeral\romannumeral
5370   \let\orgalph\@alph
5371   \let\orgAlph\@Alph
5372   \let\orgRoman\@Roman
```

Redefine:

```
5373 \ifglswrallowprimitivemods
5374 \def\the##1{%
5375 \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5376 \def\number##1{%
5377 \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5378 \fi
5379 \def\@arabic##1{%
5380 \ifx##1\c@page \gls@arabicpage\else\orgarabic##1\fi}%
5381 \def\romannumeral##1{%
5382 \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5383 \def\@Roman##1{%
5384 \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5385 \def\@alph##1{%
5386 \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5387 \def\@Alph##1{%
5388 \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5389 \@wrglossarynumberhook

```

Prevent expansion:

```
5390 \gls@disablepagerefexpansion

```

Now store location in \@glslocref:

```
5391 \protected@xdef\@glslocref{\theHglentrycounter}%
5392 \endgroup

```

Escape any special characters

```
5393 \@gls@checkmkidxchars\@glslocref

```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
5394 \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5395 \def\@glo@counterprefix{%
5396 \else
5397 \protected@edef\@glsHlocref{\theHglentrycounter}%
5398 \@gls@checkmkidxchars\@glsHlocref
5399 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5400 {\@glslocref}{\@glsHlocref}%
5401 }%
5402 \@do@gls@getcounterprefix
5403 \fi

```

De-tok label if required

```
5404 \edef\@gls@label{\glsdetoklabel{#1}}%

```

Write the information to file:

```
5405 \@do@wrglossary
5406 }

```

@do@wrglossary

```
5407 \newcommand*{\@do@wrglossary}{%

```

Determine whether to use xindy or makeindex syntax

```
5408 \ifglsxindy
```

Need to determine if the formatting information starts with a ( or ) indicating a range.

```
5409 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5410 \def\@glo@range{}%
5411 \expandafter\if\@glo@prefix(\relax
5412 \def\@glo@range{:open-range}%
5413 \else
5414 \expandafter\if\@glo@prefix)\relax
5415 \def\@glo@range{:close-range}%
5416 \fi
5417 \fi
```

Write to the glossary file using xindy syntax.

```
5418 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5419 (indexentry :key (\csname glo@\@gls@label @index\endcsname)

5420 :locref \string"\@glo@counterprefix}{\@glslocref}\string" %
5421 :attr \string"\@gls@counter\@glo@suffix\string"
5422 \@glo@range
5423 )
5424 }%
5425 \else
```

Convert the format information into the format required for makeindex

```
5426 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5427 {\@glo@counterprefix}%
```

Write to the glossary file using makeindex syntax.

```
5428 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5429 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5430 \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
5431 \fi
5432 }
```

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.)

```
5433 \newcommand*\@gls@getcounterprefix[2]{%
5434 \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5435 \ifx\@gls@thisloc\@gls@thisHloc
5436 \def\@glo@counterprefix{}%
5437 \else
5438 \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5439 \def\@glo@tmp{##2}%
5440 \ifx\@glo@tmp\@empty
5441 \def\@glo@counterprefix{}%

```

```

5442     \else
5443         \def\@glo@counterprefix{##1}%
5444     \fi
5445 }%
5446 \@gls@get@counterprefix#2.#1\end@getprefix

Warn if no prefix can be formed.
5447     \ifx\@glo@counterprefix\@empty
5448         \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5449             prefixing^^Jlocation ‘#1’. You need to modify the
5450             definition of \string\theH\@gls@counter^^Jotherwise you
5451             will get the warning: “name{\@gls@counter.#1}’ has been^^J
5452             referenced but does not exist”}%
5453     \fi
5454 \fi
5455 }

```

## 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.

```

5456 \newcommand{\do@seeglossary}[2]{%
5457 \def\@gls@xref{#2}%
5458 \@onelevel@sanitize\@gls@xref
5459 \@gls@checkmkidxchars\@gls@xref
5460 \ifglxsindy
5461   \gls@glossary{\csname glo@#1@type\endcsname}{%
5462     (indexentry
5463       :key (\csname glo@#1@index\endcsname)
5464       :xref (\string"\@gls@xref\string")
5465       :attr \string"see\string"
5466     )
5467   }%
5468 \else
5469   \gls@glossary{\csname glo@#1@type\endcsname}{%
5470     \string\glossaryentry{\csname glo@#1@index\endcsname
5471     \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5472 \fi
5473 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5474 \def\@gls@fixbraces#1#2#3\@nil{%
5475   \ifx#2[\relax
5476     \@gls@fixbraces#1#2#3\@end@fixbraces
5477   \else
5478     \def#1{{#2#3}}%
5479   \fi

```

```
5480 }
```

@@gls@fixbraces

```
5481 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5482   \def#1{[#2]{#3}}%
5483 }
```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```
5484 \DeclareRobustCommand*\glssee[3][\seename]{%
5485   \@do@seeglossary{#2}{#1}{#3}}
5486 \newcommand*\@glssee[3][\seename]{%
5487   \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.

```
5488 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5489   \emph{#1} \glsseelist{#2}}
```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```
5490 \DeclareRobustCommand*\glsseelist[1]{%
```

If there is only one item in the list, set the last separator to do nothing.

```
5491   \let\@gls@dolast\relax
```

Don’t display separator on the first iteration of the loop

```
5492   \let\@gls@donext\relax
```

Iterate through the labels

```
5493   \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
5494     \ifx\@xfor@nextelement\@nnil
```

```
5495       \@gls@dolast
```

```
5496     \else
```

```
5497       \@gls@donext
```

```
5498     \fi
```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```
5499     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5500     \let\@gls@dolast\glsseelastsep
```

```
5501     \let\@gls@donext\glsseesep
```

```
5502   }%
```

```
5503 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5504 \newcommand*\glsseelastsep{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
5505 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5506 \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.)

```
5507 \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.

```
5508 \newcommand*{\gls@save@numberlist}[1]{%
5509   \ifglssavenumberlist
5510     \toks@{#1}%
5511     \edef\do@writeaux@info{%
5512       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5513     }%
5514     \@onelevel@sanitize\do@writeaux@info
5515     \protected@write\@auxout{}\do@writeaux@info%
5516   \fi
5517 }
```

`\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.)

```
5518 \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.

```
5519 \ifcsundef{printglossary}{}%
5520 {%
  If \printglossary is already defined, issue a warning and undefine it.
5521   \@gls@warnonglossdefined
5522   \undef\printglossary
5523 }
  \printglossary has an optional argument. The default value is to set the glossary type to
  the main glossary.
5524 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
```



```

5525 \printglossary{#1}{\@print@glossary}%
5526 }

```

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`

```

5527 \newcommand*{\printglossaries}{%
5528   \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5529 }

```

`\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.

```

5530 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5531   \@printglossary{#1}{\@print@noidx@glossary}%
5532 }

```

`\printnoidxglossaries` Analogous to `\printglossaries`

```

5533 \newcommand*{\printnoidxglossaries}{%
5534   \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5535 }

```

`\printgloss@setsort` Initialise to do nothing.

```

5536 \newcommand*{\@printgloss@setsort}{}

```

`\preglossaryhook`

```

5537 \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.

```

5538 \newcommand{\@printglossary}[2]{%

```

Set up defaults.

```

5539   \def\@glo@type{\glsdefaulttype}%
5540   \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%

5541   \def\glossarytoctitle{\glossarytitle}%
5542   \let\org@glossarytitle\glossarytitle

5543   \def\@glossarystyle{%
5544     \ifx\@glossary@default@style\relax
5545       \GlossariesWarning{No default glossary style provided \MessageBreak

```

```

5546         for the glossary '@glo@type'. \MessageBreak
5547         Using deprecated fallback. \MessageBreak
5548         To fix this set the style with \MessageBreak
5549         \string\setglossarystyle\space or use the \MessageBreak
5550         style key=value option}%
5551     \fi
5552 }%
5553 \def\gls@dotoc@title{\gls@set@toc@title{\@glo@type}}%

Store current value of \glossaryentrynumbers. (This may be changed via the optional argument)
5554 \let\org@glossaryentrynumbers\glossaryentrynumbers

Localise the effects of the optional argument
5555 \bgroup

Activate or deactivate sort key:
5556 \let\printgloss@set@sort

Determine settings specified in the optional argument.
5557 \setkeys{printgloss}{#1}%

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)
5558 \ifx\glossarytitle\org@glossarytitle
5559 \else
5560 \expandafter\let\csname @glo@type@\@glo@type @title\endcsname
5561 \glossarytitle
5562 \fi

Allow a high-level user command to indicate the current glossary
5563 \let\currentglossary\@glo@type

Enable individual number lists to be suppressed.
5564 \let\org@glossaryentrynumbers\glossaryentrynumbers
5565 \let\gls@nonextpages\@gls@nonextpages

Enable individual number list to be activated:
5566 \let\gls@nextpages\@gls@nextpages

Enable suppression of description terminators.
5567 \let\gls@nopostdesc\@nopostdesc

Set up the entry for the TOC
5568 \gls@dotoc@title

Set the glossary style
5569 \let\gls@style\@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):
5570 \let\gls@org@glossaryentryfield\glossentry
5571 \let\gls@org@glossarysubentryfield\subglossentry

```

```

5572 \renewcommand{\glossentry}[1]{%
5573 \xdef\glscurrententrylabel{\glsetoklabel{##1}}%
5574 \gls@org@glossaryentryfield{##1}%
5575 }%
5576 \renewcommand{\subglossentry}[2]{%
5577 \xdef\glscurrententrylabel{\glsetoklabel{##2}}%
5578 \gls@org@glossarysubentryfield{##1}{##2}%
5579 }%

5580 \@gls@preglossaryhook

```

Now do the handler macro that deals with the actual glossary:

```

5581 #2%

End the current scope
5582 \egroup

Reset \glossaryentrynumbers
5583 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers

Suppress warning about no \printglossary
5584 \global\let\warn@noprntglossary\relax
5585 }

```

@print@glossary Internal workings of \printglossary dealing with reading the external file.

```

5586 \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.)

5587 \makeatletter

Input the glossary file, if it exists.
5588 \@input@{\jobname.\csname @glo@type\@glo@type @in\endcsname}%

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all
write commands are done.) This might produce an empty page, but at this point the docu-
ment isn't complete, so it shouldn't matter.

5589 \IfFileExists{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
5590 {}%
5591 {\null}%

If xindy is being used, need to write the language dependent information to the .aux file for
makeglossaries.

5592 \ifglxindy
5593 \ifcsundef{@xdy@\@glo@type @language}%
5594 {%
5595 \edef\@do@auxoutstuff{%
5596 \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.

5597 \noexpand\immediate\noexpand\write\@auxout{%

```

```

5598         \string\providecommand\string\@xdylanguage[2]{}%
5599         \noexpand\immediate\noexpand\write\@auxout{%
5600         \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5601     }%
5602 }%
5603 }%
5604 {%
5605     \edef\@do@auxoutstuff{%
5606     \noexpand\AtEndDocument{%
5607     \noexpand\immediate\noexpand\write\@auxout{%
5608     \string\providecommand\string\@xdylanguage[2]{}%
5609     \noexpand\immediate\noexpand\write\@auxout{%
5610     \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5611     @language\endcsname}}%
5612     }%
5613     }%
5614     }%
5615     \@do@auxoutstuff
5616     \edef\@do@auxoutstuff{%
5617     \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.

```

5618         \noexpand\immediate\noexpand\write\@auxout{%
5619         \string\providecommand\string\@gls@codepage[2]{}%
5620         \noexpand\immediate\noexpand\write\@auxout{%
5621         \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5622     }%
5623 }%
5624 \@do@auxoutstuff
5625 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

5626 \renewcommand*{\@warn@nomakeglossaries}{%
5627 \GlossariesWarningNoLine{\string\makeglossaries\space
5628 hasn't been used,^^Jthe glossaries will not be updated}%
5629 }%
5630 }

```

The sort macros all have the syntax:

$$\backslash@glo@sortmacro@<order>\{<type>\}$$

where  $\langle order \rangle$  is the sort order as specified by the sort key and  $\langle type \rangle$  is the glossary type. (The referenced entry list is stored in  $\backslash@gls@sref@<type>$ . The actual sorting is done by  $\backslash@glo@sortentries\{<handler>\}\{<type>\}$ .

$\backslash@glo@sortentries$

```

5631 \newcommand*{\@glo@sortentries}[2]{%

```

```

5632 \glosortentrieswarning
5633 \def\@glo@sortinglist{}%
5634 \def\@glo@sortinghandler{#1}%
5635 \edef\@glo@type{#2}%
5636 \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
5637 \csdef{\@glsref@#2}{}%
5638 \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5639 \xifinlistcs{\@this@label}{\@glsref@#2}%
5640 {}%
5641 {%
5642 \listcsxadd{\@glsref@#2}{\@this@label}%
5643 }%
5644 \ifcsdef{\@glo@sortingchildren@\@this@label}%
5645 {%
5646 \@glo@addchildren{#2}{\@this@label}%
5647 }%
5648 {}%
5649 }%
5650 }

```

@glo@addchildren `\@glo@addchildren{<type>}{<parent>}`

```

5651 \newcommand*{\@glo@addchildren}[2]{%

```

Scope to allow nesting.

```

5652 \bgroup
5653 \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
5654 \@for\@this@childlabel:=\@glo@childlist\do
5655 {%

```

Check this label hasn't already been added.

```

5656 \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
5657 {}%
5658 {%
5659 \listcsxadd{\@glsref@#1}{\@this@childlabel}%
5660 }%

```

Does this child have children?

```

5661 \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
5662 {%
5663 \@glo@addchildren{#1}{\@this@childlabel}%
5664 }%
5665 {%
5666 }%
5667 }%
5668 \egroup
5669 }

```

@do@sortentries

```
5670 \newcommand*{\@glo@do@sortentries}[1]{%
5671   \ifglshasparent{#1}%
5672   {%
```

This entry has a parent, so add it to the child list

```
5673   \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
5674   \ifcsundef{glo@sortingchildren@\@glo@parent}%
5675   {%
5676     \csdef{glo@sortingchildren@\@glo@parent}{}%
5677   }%
5678   {}%
5679   \expandafter\@glo@sortedinsert
5680   \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
5681   \xifinlistcs{\@glo@parent}{@glstetoklabel{#1}@parent}%
5682   {%
```

Yes, it has so do nothing.

```
5683   }%
5684   {%
```

No, it hasn't so add it now.

```
5685   \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5686   }%
5687   }%
5688   {%
5689   \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5690   }%
5691 }
```

glo@sortedinsert

```
\@glo@sortedinsert{\<list>}{\<entry label>}
```

Insert into list.

```
5692 \newcommand*{\@glo@sortedinsert}[2]{%
5693   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5694 }
```

The sort handlers need to be in the form required by datatool's `\dtl@sortlist` macro. These must set the count register `\dtl@sortresult` to either  $-1$  ( $\#1$  less than  $\#2$ ),  $0$  ( $\#1 = \#2$ ) or  $+1$  ( $\#1$  greater than  $\#2$ ).

orthandler@word

```
5695 \newcommand*{\@glo@sorthandler@word}[2]{%
5696   \letcs@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5697   \letcs@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5698   \edef\@glo@do@compare{%
5699     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
```

```

5700     {\expandonce\@gls@sort@B}%
5701     {\expandonce\@gls@sort@A}%
5702 }%
5703 \glo@do@compare
5704 }

```

thandler@letter

```

5705 \newcommand*{\@glo@sorthandler@letter}[2]{%
5706   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5707   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5708   \edef\glo@do@compare{%
5709     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5710     {\expandonce\@gls@sort@B}%
5711     {\expandonce\@gls@sort@A}%
5712   }%
5713   \glo@do@compare
5714 }

```

orthandler@case Case-sensitive sort.

```

5715 \newcommand*{\@glo@sorthandler@case}[2]{%
5716   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5717   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5718   \edef\glo@do@compare{%
5719     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5720     {\expandonce\@gls@sort@B}%
5721     {\expandonce\@gls@sort@A}%
5722   }%
5723   \glo@do@compare
5724 }

```

thandler@nocase Case-insensitive sort.

```

5725 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5726   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5727   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5728   \edef\glo@do@compare{%
5729     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5730     {\expandonce\@gls@sort@B}%
5731     {\expandonce\@gls@sort@A}%
5732   }%
5733   \glo@do@compare
5734 }

```

@sortmacro@word Sort macro for ‘word’

```

5735 \newcommand*{\@glo@sortmacro@word}[1]{%
5736   \ifdefstring{\@glo@default@sorttype}{standard}%
5737   {%
5738     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5739   }%
5740   {%

```

```

5741 \PackageError{glossaries}{Conflicting sort options:^^J
5742 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5743 \string\printnoidxglossary[sort=word]}{}%
5744 }%
5745 }

```

ortmacro@letter Sort macro for ‘letter’

```

5746 \newcommand*{\@glo@sortmacro@letter}[1]{%
5747 \ifdefstring{\@glo@default@sorttype}{standard}%
5748 {%
5749 \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5750 }%
5751 {%
5752 \PackageError{glossaries}{Conflicting sort options:^^J
5753 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5754 \string\printnoidxglossary[sort=letter]}{}%
5755 }%
5756 }

```

tmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5757 \newcommand*{\@glo@sortmacro@standard}[1]{%
5758 \ifdefstring{\@glo@default@sorttype}{standard}%
5759 {%
5760 \ifcsdef{\@glo@sorthandler@glorder}%
5761 {%
5762 \@glo@sortentries{\csuse{\@glo@sorthandler@glorder}}{#1}%
5763 }%
5764 {%
5765 \PackageError{glossaries}{Unknown sort handler ‘\glorder’}{}%
5766 }%
5767 }%
5768 {%
5769 \PackageError{glossaries}{Conflicting sort options:^^J
5770 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5771 \string\printnoidxglossary[sort=standard]}{}%
5772 }%
5773 }

```

@sortmacro@case Sort macro for ‘case’

```

5774 \newcommand*{\@glo@sortmacro@case}[1]{%
5775 \ifdefstring{\@glo@default@sorttype}{standard}%
5776 {%
5777 \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5778 }%
5779 {%
5780 \PackageError{glossaries}{Conflicting sort options:^^J
5781 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5782 \string\printnoidxglossary[sort=case]}{}%
5783 }%

```



5784 }

ortmacro@nocase Sort macro for ‘nocase’

```
5785 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5786   \ifdefstring{\@glo@default@sorttype}{standard}%
5787   {%
5788     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5789   }%
5790   {%
5791     \PackageError{glossaries}{Conflicting sort options:^^J
5792       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5793       \string\printnoidxglossary[sort=nocase]}{}%
5794   }%
5795 }
```

o@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@*type*.

```
5796 \newcommand*{\@glo@sortmacro@def}[1]{%
5797   \def\@glo@sortinglist{%
5798     \for@gl@sentries[#1]{\@gls@thislabel}%
5799     {%
5800       \xifinlistcs{\@gls@thislabel}{\@gls@sref@#1}%
5801       {%
5802         \listadd{\@glo@sortinglist}{\@gls@thislabel}%
5803       }%
5804     }%
5805   }%
5806   }%
5807   \cslet{\@gls@sref@#1}{\@glo@sortinglist}%
5808 }
```

Hasn't been referenced.

```
5805   }%
5806   }%
5807   \cslet{\@gls@sref@#1}{\@glo@sortinglist}%
5808 }
```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5809 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5810   \ifinlistcs{#1}{\@gls@sref@\@glo@type}%
5811   {}%
5812   {%
5813     \listcsadd{\@gls@sref@\@glo@type}{#1}%
5814   }%
5815   \ifcsdef{\@glo@sortingchildren@#1}%
5816   {%
5817     \@glo@addchildren{\@glo@type}{#1}%
5818   }%
5819   {}%
5820 }
```

o@sortmacro@use Sort macro for ‘use’. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
5821 \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.

```
5822 \newcommand*{\@print@noidx@glossary}{%
5823   \ifcsdef{@glsref@{\@glo@type}}%
5824   {%
```

Sort the entries:

```
5825   \ifcsdef{@glo@sortmacro@{\@glo@sorttype}}%
5826   {%
5827     \csuse{@glo@sortmacro@{\@glo@sorttype}}{\@glo@type}%
5828   }%
5829   {%
5830     \PackageError{glossaries}{Unknown sort handler '@glo@sorttype'}{ }%
5831   }%
```

Do the glossary heading and preamble

```
5832   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5833   \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.

```
5834   \def\@gls@currentlettergroup{}%
5835   \begin{theglossary}%
5836   \glossaryheader
5837   \glsresetentrylist
```

Iterate through the entries.

```
5838   \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}}%
```

Finally end the glossary and do the postamble:

```
5839   \end{theglossary}%
5840   \glossarypostamble
5841 }%
5842 {%
5843   \@gls@noref@warn{\@glo@type}%
5844 }%
5845 }
```

\glo@grabfirst

```
5846 \def\glo@grabfirst#1#2\@nil{%
5847   \def\@gls@firsttok{#1}%
5848   \ifdefempty\@gls@firsttok
5849   {%
5850     \def\@glo@thislettergrp{0}%
5851   }%
5852   {%
```

Sanitize it:

```
5853 \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
5854 \expandafter\@glo@grabfirst\@gls@firsttok{}\{}\@nil
5855 }%
5856 }
```

\@glo@grabfirst

```
5857 \def\@glo@grabfirst#1#2\@nil{%
5858 \ifdefempty\@glo@thislettergrp
5859 {%
5860 \def\@glo@thislettergrp{glssymbols}%
5861 }%
5862 {%
5863 \count@=\uccode'#1\relax
5864 \ifnum\count@=0\relax
5865 \def\@glo@thislettergrp{glssymbols}%
5866 \else
5867 \ifdefstring\@glo@sorttype{case}%
5868 {%
5869 \count@='#1\relax
5870 }%
5871 {%
5872 }%
5873 \edef\@glo@thislettergrp{\the\count@}%
5874 \fi
5875 }%
5876 }
```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label.  
This only allows one sublevel.

```
5877 \newcommand{\@gls@noidx@do}[1]{%
```

Get this entry's location list

```
5878 \global\letcs{\@gls@loclist}{\glo@\glsdetoklabel{#1}@loclist}%
```

Does this entry have a parent?

```
5879 \ifglshasparent{#1}%
5880 {%
```

Has a parent.

```
5881 \gls@level=\csuse{\glo@\glsdetoklabel{#1}@level}\relax
5882 \ifdefvoid{\@gls@loclist}
5883 {%
5884 \subglossentry{\gls@level}{#1}{}%
5885 }%
5886 {%
5887 \subglossentry{\gls@level}{#1}%
5888 }
```

```

5889     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5890     }%
5891     }%
5892     }%
5893     {%

```

Doesn't have a parent Get this entry's sort key

```

5894     \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%

```

Fetch the first letter:

```

5895     \expandafter\glo@grabfirst\@gls@sort{}{} \@nil
5896     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5897     {}%
5898     {%

```

Do the group header:

```

5899     \ifdefempty{\@gls@currentlettergroup}{}%
5900     {%

```

The group skip may start a new scope, so make a global assignment.

```

5901     \global\let\@glo@thislettergrp\@glo@thislettergrp
5902     \glsgroupskip
5903     }%
5904     \glsgroupheading{\@glo@thislettergrp}%
5905     }%

5906     \global\let\@gls@currentlettergroup\@glo@thislettergrp

```

Do this entry:

```

5907     \ifdefvoid{\@gls@loclist}
5908     {%
5909     \glossentry{#1}{}%
5910     }%
5911     {%
5912     \glossentry{#1}%
5913     {%
5914     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5915     }%
5916     }%
5917     }%
5918 }

```

\glsnoidxloclist    \glsnoidxloclist{<list cs>}

Display location list.

```

5919 \newcommand*{\glsnoidxloclist}[1]{%
5920   \def\@gls@noidxloclist@sep{}%
5921   \def\@gls@noidxloclist@prev{}%
5922   \forlistloop{\glsnoidxloclisthandler}{#1}%
5923 }

```

`xloclisthandler` Handler for location list iterator.

```
5924 \newcommand*{\glsnoidxloclisthandler}[1]{%
5925   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5926   {%
      Same as previous location so skip.
5927   }%
5928   {%
5929     \@gls@noidxloclist@sep
5930     #1%
5931     \def\@gls@noidxloclist@sep{\delimN}%
5932     \def\@gls@noidxloclist@prev{#1}%
5933   }%
5934 }
```

`yloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```
5935 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
5936   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5937   {%
      Same as previous location so skip.
5938   }%
5939   {%
5940     \@gls@noidxloclist@sep
5941     \@gls@noidxloclist@prev
5942     \def\@gls@noidxloclist@prev{#1}%
5943   }%
5944 }
```

`snoidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
5945 \newcommand*\glsnoidxdisplayloc[4]{%
5946   \setentrycounter[#1]{#2}%
5947   \csuse{#3}{#4}%
5948 }
```

`\@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.

```
5949 \newcommand*\@gls@reference[3]{%
```

Add to label list

```
5950   \glsdoifexistsorwarn{#2}%
5951   {%
5952     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}}{}{}}%
```

```

5953 \ifinlistcs{#2}{@glsref@#1}%
5954 {}%
5955 {\listcsgadd{@glsref@#1}{#2}}%

```

Add to location list

```

5956 \ifcsundef{glo@glstdetoklabel{#2}@loclist}%
5957 {\csgdef{glo@glstdetoklabel{#2}@loclist}{}}%
5958 {}%
5959 \listcsgadd{glo@glstdetoklabel{#2}@loclist}{#3}%
5960 }%
5961 }

```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```

5962 \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`.

```

5963 \define@key{printgloss}{title}{%
5964 \def\glossarytitle{#1}%
5965 \let\gls@dotocitle\relax
5966 }

```

The toctitle sets the text used for the relevant entry in the table of contents.

```

5967 \define@key{printgloss}{toctitle}{%
5968 \def\glossarytoctitle{#1}%
5969 \let\gls@dotocitle\relax
5970 }

```

The style key sets the glossary style (but only for the given glossary).

```

5971 \define@key{printgloss}{style}{%
5972 \ifcsundef{@glsstyle@#1}%
5973 {%
5974 \PackageError{glossaries}%
5975 {Glossary style ‘#1’ undefined}{}%
5976 }%
5977 {%
5978 \def@glossarystyle{\setglossentrycompatibility
5979 \csname @glsstyle@#1\endcsname}%
5980 }%
5981 }

```

The numberedsection key determines if this glossary should be in a numbered section.

```

5982 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5983 false,nolabel,autolabel,nameref}[nolabel]{%
5984 \ifcase\nr\relax
5985 \renewcommand*{\@@glossarysecstar}{*}%
5986 \renewcommand*{\@@glossaryseclabel}{}%
5987 \or
5988 \renewcommand*{\@@glossarysecstar}{}%
5989 \renewcommand*{\@@glossaryseclabel}{}%

```

```

5990 \or
5991   \renewcommand*{\@@glossarysecstar}{}%
5992   \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5993 \or
5994   \renewcommand*{\@@glossarysecstar}{*}%
5995   \renewcommand*{\@@glossaryseclabel}{%
5996     \protected@edef\@currentlabelname{\glossarytoctitle}%
5997     \label{\glsautoprefix\@glo@type}}%
5998 \fi
5999 }

```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

6000 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
6001   \csuse{glsnogroupskip#1}%
6002 }

```

The `nopostdot` key has the same effect as the package option of the same name.

```

6003 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
6004   \csuse{glsnopostdot#1}%
6005 }

```

The `entrycounter` key is the same as the package option but localised to the current glossary.

```

6006 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
6007   \csuse{glsentrycounter#1}%
6008   \ifglsentrycounter
6009     \ifx\@gls@counterwithin\@empty
6010       \newcounter{glossaryentry}%
6011     \else
6012       \newcounter{glossaryentry}[\@gls@counterwithin]%
6013     \fi
6014     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
6015     \renewcommand*{\glsresetentrycounter}{%
6016       \setcounter{glossaryentry}{0}%
6017     }%
6018     \renewcommand*{\glsstepentry}[1]{%
6019       \refstepcounter{glossaryentry}%
6020       \label{glsentry-\glsdetoklabel{##1}}%
6021     }%
6022     \renewcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}%
6023     \renewcommand*{\glsentryitem}[1]{%
6024       \glsstepentry{##1}\glsentrycounterlabel
6025     }%
6026   \else
6027     \renewcommand*{\glsresetentrycounter}{}%
6028     \renewcommand*{\glsstepentry}[1]{}%
6029     \renewcommand*{\glsentrycounterlabel}{}%
6030     \renewcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
6031   \fi
6032 }

```

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.

```

6033 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6034   \csuse{glssubentrycounter#1}%
6035   \ifglssubentrycounter
6036     \ifundef\c@glossarysubentry
6037     {%
6038       \ifglsentrycounter
6039         \newcounter{glossarysubentry}[glossaryentry]%
6040       \else
6041         \newcounter{glossarysubentry}
6042       \fi
6043     }{}%
6044     \renewcommand*{\glstepsentry}[1]{%
6045       \edef\currentglssubentry{\glstetoklabel{##1}}%
6046       \refstepcounter{glossarysubentry}%
6047       \label{glstentry-\currentglssubentry}%
6048     }%
6049     \renewcommand*{\glresetsubentrycounter}{%
6050       \setcounter{glossarysubentry}{0}%
6051     }%
6052     \renewcommand*{\glssubentryitem}[1]{%
6053       \glstepsentry{##1}\glssubentrycounterlabel
6054     }%
6055     \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry\space}%
6056     \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6057   \else
6058     \renewcommand*{\glssubentryitem}[1]{}%
6059     \renewcommand*{\glstepsentry}[1]{}%
6060     \renewcommand*{\glresetsubentrycounter}{}%
6061     \renewcommand*{\glssubentrycounterlabel}{}%
6062   \fi
6063 }

```

The nonumberlist key determines if this glossary should have a number list.

```

6064 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
6065   \ifglsnonumberlist
6066     \def\glossaryentrynumbers##1{%
6067   \else
6068     \def\glossaryentrynumbers##1{##1}%
6069   \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

6070 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

@assign@sortkey Issue error if used with \printglossary

```

6071 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6072   \PackageError{glossaries}{‘sort’ key not permitted with
6073   \string\printglossary}%

```



```

6074 {The ‘sort’ key may only be used with \string\printnoidxglossary}%
6075 }

```

`@assign@sortkey` For use with `\printnoidxglossary`

```

6076 \newcommand*{\@glo@assign@sortkey}[1]{%
6077   \def\@glo@sorttype{#1}%
6078 }

```

`\glsnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonextpages` is 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.

```

6079 \newcommand*{\@glsnonextpages}{%
6080   \gdef\glossaryentrynumbers##1{%
6081     \glsresetentrylist
6082   }%
6083 }

```

`\@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.

```

6084 \newcommand*{\@glsnextpages}{%
6085   \gdef\glossaryentrynumbers##1{%
6086     ##1\glsresetentrylist}}

```

`sresetentrylist` Resets `\glossaryentrynumbers`

```

6087 \newcommand*{\glsresetentrylist}{%
6088   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glsnonextpages` Outside of `\printglossary` this does nothing.

```

6089 \newcommand*{\glsnonextpages}{}

```

`\glsnextpages` Outside of `\printglossary` this does nothing.

```

6090 \newcommand*{\glsnextpages}{}

```

`glossaryentry` If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

6091 \ifglentrycounter
6092   \ifx\@gls@counterwithin\@empty
6093     \newcounter{glossaryentry}
6094   \else
6095     \newcounter{glossaryentry}[\@gls@counterwithin]
6096   \fi
6097   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
6098 \fi

```

`\glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```

6099 \ifglssubentrycounter
6100   \ifglsentrycounter
6101     \newcounter{glossarysubentry}[glossaryentry]
6102   \else
6103     \newcounter{glossarysubentry}
6104   \fi
6105   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6106 \fi

```

`subentrycounter` Resets the `glossarysubentry` counter.

```

6107 \ifglssubentrycounter
6108   \newcommand*{\glsresetsubentrycounter}{%
6109     \setcounter{glossarysubentry}{0}%
6110   }
6111 \else
6112   \newcommand*{\glsresetsubentrycounter}{}
6113 \fi

```

`subentrycounter` Resets the `glossareentry` counter.

```

6114 \ifglsentrycounter
6115   \newcommand*{\glsresetentrycounter}{%
6116     \setcounter{glossaryentry}{0}%
6117   }
6118 \else
6119   \newcommand*{\glsresetentrycounter}{}
6120 \fi

```

`\glsstepentry` Advance the `glossaryentry` counter if in use. The argument is the label associated with the entry.

```

6121 \ifglsentrycounter
6122   \newcommand*{\glsstepentry}[1]{%
6123     \refstepcounter{glossaryentry}%
6124     \label{glsentry-\glsdetoklabel{#1}}%
6125   }
6126 \else
6127   \newcommand*{\glsstepentry}[1]{}
6128 \fi

```

`glsstepsubentry` Advance the `glossarysubentry` counter if in use. The argument is the label associated with the subentry.

```

6129 \ifglssubentrycounter
6130   \newcommand*{\glsstepsubentry}[1]{%
6131     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6132     \refstepcounter{glossarysubentry}%
6133     \label{glsentry-\currentglssubentry}%
6134   }

```

```

6135 \else
6136   \newcommand*{\glsstepsubentry}[1]{%
6137 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

6138 \ifglsentrycounter
6139   \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
6140 \else
6141   \ifglssubentrycounter
6142     \newcommand*{\glsrefentry}[1]{\ref{glsentry-\glsdetoklabel{#1}}}
6143   \else
6144     \newcommand*{\glsrefentry}[1]{\gls{#1}}
6145   \fi
6146 \fi

```

`trycounterlabel` Defines how to display the glossaryentry counter.

```

6147 \ifglsentrycounter
6148   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
6149 \else
6150   \newcommand*{\glsentrycounterlabel}{}
6151 \fi

```

`trycounterlabel` Defines how to display the glossarysubentry counter.

```

6152 \ifglssubentrycounter
6153   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry)\space}
6154 \else
6155   \newcommand*{\glssubentrycounterlabel}{}
6156 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```

6157 \ifglsentrycounter
6158   \newcommand*{\glsentryitem}[1]{%
6159     \glsstepentry{#1}\glsentrycounterlabel
6160   }
6161 \else
6162   \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
6163 \fi

```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

6164 \ifglssubentrycounter
6165   \newcommand*{\glssubentryitem}[1]{%
6166     \glsstepsubentry{#1}\glssubentrycounterlabel
6167   }
6168 \else
6169   \newcommand*{\glssubentryitem}[1]{}
6170 \fi

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

6171 \ifcsundef{theglossary}%
6172 {%
6173   \newenvironment{theglossary}{}{}%
6174 }%
6175 {%
6176   \@gls@warnontheGLOSSdefined
6177   \renewenvironment{theglossary}{}{}%
6178 }

```

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`

```

6179 \newcommand*{\glossaryheader}{}

```

`\glstarget`    `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

6180 \newcommand*{\glstarget}[2]{\@glstarget{\glo@linkprefix#1}{#2}}

```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

`\compatibleglossentry`

```
\glossentry{<label>}{<page-list>}
```

```

6181 \providecommand*{\compatibleglossentry}[2]{%
6182   \toks0{#2}%
6183   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6184     {\noexpand\glsnamefont
6185       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6186     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6187     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6188     {\the\toks0}}%
6189   }%
6190   \@do@glossentry
6191 }

```

`\glossentryname`

```

6192 \newcommand*{\glossentryname}[1]{%
6193   \glsdoifexistsorwarn{#1}%
6194   {%

```

```

6195 \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
6196 \expandafter\glsnamefont\expandafter{\glo@name}%
6197 }%
6198 }

```

\Glossentryname

```

6199 \newcommand*{\Glossentryname}[1]{%
6200 \glsdoifexistsorwarn{#1}%
6201 {%
6202 \glsnamefont{\Glsentryname{#1}}%
6203 }%
6204 }

```

\glossentrydesc

```

6205 \newcommand*{\glossentrydesc}[1]{%
6206 \glsdoifexistsorwarn{#1}%
6207 {%
6208 \glsentrydesc{#1}%
6209 }%
6210 }

```

\Glossentrydesc

```

6211 \newcommand*{\Glossentrydesc}[1]{%
6212 \glsdoifexistsorwarn{#1}%
6213 {%
6214 \Glsentrydesc{#1}%
6215 }%
6216 }

```

\glossentrysymbol

```

6217 \newcommand*{\glossentrysymbol}[1]{%
6218 \glsdoifexistsorwarn{#1}%
6219 {%
6220 \glsentrysymbol{#1}%
6221 }%
6222 }

```

\Glossentrysymbol

```

6223 \newcommand*{\Glossentrysymbol}[1]{%
6224 \glsdoifexistsorwarn{#1}%
6225 {%
6226 \Glsentrysymbol{#1}%
6227 }%
6228 }

```

\blesubglossentry

```
\subglossentry{<level>}{<label>}{<page-list>}
```

```

6229 \providecommand*\compatiblesubglossentry}[3]{%
6230   \toks@{#3}%
6231   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6232     {#2}%
6233     {\noexpand\glsnamefont
6234       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6235     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6236     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6237     {\the\toks@}}%
6238   }%
6239   \@do@subglossentry
6240 }

```

rycompatibility

```

6241 \newcommand*\setglossentrycompatibility{%
6242   \let\glossentry\compatibleglossentry
6243   \let\subglossentry\compatiblesubglossentry
6244 }
6245 \setglossentrycompatibility

```

glossaryentryfield

```

\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}
{<page-list>}

```

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```

6246 \newcommand{\glossaryentryfield}[5]{%
6247   \GlossariesWarning
6248   {Deprecated use of \string\glossaryentryfield.^^J
6249     I recommend you change to \string\glossentry.^^J
6250     If you've just upgraded, try removing your gls auxiliary
6251     files^^J and recompile}%
6252   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

glossarysubentryfield

```

\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.)

```

6253 \newcommand*\glossarysubentryfield}[6]{%
6254   \GlossariesWarning
6255   {Deprecated use of \string\glossarysubentryfield.^^J
6256     I recommend you change to \string\subglossentry.^^J
6257     If you've just upgraded, try removing your gls auxiliary

```

```

6258   files^^J and recompile}%
6259   \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`

```

6260 \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`

```

6261 \newcommand*{\glsgroupheading}[1]{}

```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an a, while entries belonging to another group could be defined so that the sort key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgetgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by *<label>*. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

`\glsgetgrouptitle`

```

6262 \newcommand*{\glsgetgrouptitle}[1]{%
6263   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6264   \@gls@grptitle
6265 }

```

`s@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
6266 \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.

```
6267 \dtl@ifsingle{#1}%
6268 {%
6269   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6270 }%
6271 {%
6272   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6273               or test{\ifstrequal{#1}{glsnumbers}}}%
6274   {%
6275     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6276   }%
6277   {%
6278     \def#2{#1}%
6279   }%
6280 }%
6281 }
```

`x@getgrouptitle` Version for the no-indexing app option:

```
6282 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6283   \DTLifint{#1}%
6284   {\edef#2{\char#1\relax}}%
6285   {%
6286     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6287   }%
6288 }
```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`lsgetgrouplabel`

```
6289 \newcommand*{\glsgetgrouplabel}[1]{%
6290 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6291 \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`

```
6292 \newcommand*{\setentrycounter}[2] [] {%
```



```

6293 \def\@glo@counterprefix{#1}%
6294 \ifx\@glo@counterprefix\empty
6295   \def\@glo@counterprefix{.}%
6296 \else
6297   \def\@glo@counterprefix{.#1.}%
6298 \fi
6299 \def\glsentrycounter{#2}%
6300 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

6301 \newcommand*{\setglossarystyle}[1]{%
6302   \ifcsundef{@glsstyle@#1}%
6303   {%
6304     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6305   }%
6306   {%
6307     \csname @glsstyle@#1\endcsname
6308   }%

```

Set the default style if it's not already set.

```

6309 \ifx\@glossary@default@style\relax
6310   \protected@edef\@glossary@default@style{#1}%
6311 \fi
6312 }

```

`\glossarystyle`

```

6313 \newcommand*{\glossarystyle}[1]{%
6314   \ifcsundef{@glsstyle@#1}%
6315   {%
6316     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6317   }%
6318   {%
6319     \GlossariesWarning
6320     {Deprecated command \string\glossarystyle.^~J
6321      I recommend you switch to \string\setglossarystyle\space unless
6322      you want to maintain backward compatibility}%
6323     \setglossentrycompatibility
6324     \csname @glsstyle@#1\endcsname

6325     \ifcsdef{@glscompstyle@#1}%
6326     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6327     {}%
6328   }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6329 \ifx\@glossary@default@style\relax
6330   \protected@edef\@glossary@default@style{#1}%

```

```
6331 \fi
6332 }
```

`\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.

```
6333 \newcommand{\newglossarystyle}[2]{%
6334   \ifcsundef{@glsstyle@#1}%
6335   {%
6336     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6337   }%
6338   {%
6339     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6340   }%
6341 }
```

`\newglossarystyle` Code for this macro supplied by Marco Daniel.

```
6342 \newcommand{\renewglossarystyle}[2]{%
6343   \ifcsundef{@glsstyle@#1}%
6344   {%
6345     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6346   }%
6347   {%
6348     \csdef{@glsstyle@#1}{#2}%
6349   }%
6350 }
```

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`

```
6351 \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`

```
6352 \ifcsundef{hyperlink}%
6353 {%
6354   \def\glshypernumber#1{#1}%
6355 }%
6356 {%
6357   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6358 }
```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6359 \def\@glshypernumber#1\@nohyperpage#2#3\@nil{%
6360   \ifx\#1\%
6361     \else
6362       \@delimR#1\delimR\delimR\%
6363     \fi
6364     \ifx\#2\%
6365       \else
6366         #2%
6367       \fi
6368       \ifx\#3\%
6369         \else
6370           \@glshypernumber#3\@nil
6371         \fi
6372 }
```

`\@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`

```
6373 \def\@delimR#1\delimR #2\delimR #3\{%
6374 \ifx\#2\%
6375   \@delimN{#1}%
6376 \else
6377   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6378 \fi}
```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```
6379 \def\@delimN#1{\@delimN#1\delimN \delimN\%
6380 \def\@delimN#1\delimN #2\delimN#3\{%
6381 \ifx\#3\%
```

```

6382 \@gls@numberlink{#1}%
6383 \else
6384 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6385 \fi
6386 }

```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

6387 \def\@gls@numberlink#1{%
6388 \begingroup
6389 \toks@={}%
6390 \@gls@removespaces#1 \@nil
6391 \endgroup}

6392 \def\@gls@removespaces#1 #2\@nil{%
6393 \toks@=\expandafter{\the\toks@#1}%
6394 \ifx\#2\%
6395 \edef\x{\the\toks@}%
6396 \ifx\x\empty
6397 \else

6398 \hyperlink{\glstrycounter\@glo@counterprefix\the\toks@}%
6399 {\the\toks@}%
6400 \fi
6401 \else
6402 \@gls@ReturnAfterFi{%
6403 \@gls@removespaces#2\@nil
6404 }%
6405 \fi
6406 }
6407 \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
6408 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}

\hypersf
6409 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}

\hypertt
6410 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
6411 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}

\hypermd
6412 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}

```

```

\hyperit
6413 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
6414 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
6415 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
6416 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
6417 \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.

```

6418 \newcommand{\oldacronym}[4][\gls@label]{%
6419   \def\gls@label{#2}%
6420   \newacronym[#4]{#1}{#2}{#3}%
6421   \ifcsundef{xspace}%
6422   {%
6423     \expandafter\edef\csname#1\endcsname{%
6424       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6425   }%
6426  }%
6427   {%
6428     \expandafter\edef\csname#1\endcsname{%
6429       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6430         \noexpand\gls{#1}\noexpand\xspace}%

```

```

6431     }%
6432   }%
6433 }

```

```
\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`

```
6434 \newcommand{\newacronym}[4][{}]{}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix` Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, `ABCS` looks as though the “s” is part of the acronym, but `ABCS` looks as though the “s” is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is needed to cancel it out.

```
6435 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6436 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6437 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
6438 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6439 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
6440 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6441 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}

6442 \newcommand*\ns@acrfull[2] [] {%
6443   \new@ifnextchar[{\@acrfull{#1}{#2}}}%
6444   {\@acrfull{#1}{#2} []}%
6445 }
```

`\@acrfull` Low-level macro:

```
6446 \def\@acrfull#1#2[#3] {%
    Make it easier for acronym styles to change this:
6447   \acrfullfmt{#1}{#2}{#3}%
6448 }
```

Using `\acrlinkfullformat` and `\acrfullformat` is now deprecated as it can cause complications with the first letter upper case variants, but the package needs to provide backward compatibility support.

`\acrfullfmt` No case change full format.

```
6449 \newcommand*\acrfullfmt[3] {%
6450   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6451 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```
6452 \newcommand{\acrlinkfullformat}[5] {%
6453   \acrfullformat{#1{#3}{#4} [#5]}{#2{#3}{#4} []}%
6454 }
```

`\acrfullformat` Default full form is `<long>` (`<short>`).

```
6455 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
6456 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
6457 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}

6458 \newcommand*\ns@Acrfull[2] [] {%
6459   \new@ifnextchar[{\@Acrfull{#1}{#2}}}%
6460   {\@Acrfull{#1}{#2} []}%
6461 }
```

Low-level macro:

```
6462 \def\@Acrfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6463 \Acrfullfmt{#1}{#2}{#3}%  
6464 }
```

\Acrfullfmt First letter upper case full format.

```
6465 \newcommand*{\Acrfullfmt}[3]{%  
6466 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%  
6467 }
```

\ACRfull

```
6468 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}  
  
6469 \newcommand*\ns@ACRfull[2][{}]{%  
6470 \new@ifnextchar[{\@ACRfull{#1}{#2}}%  
6471 {\@ACRfull{#1}{#2}[]}%  
6472 }
```

Low-level macro:

```
6473 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6474 \ACRfullfmt{#1}{#2}{#3}%  
6475 }
```

\ACRfullfmt All upper case full format.

```
6476 \newcommand*{\ACRfullfmt}[3]{%  
6477 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6478 }
```

Plural:

\acrfullpl

```
6479 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}  
  
6480 \newcommand*\ns@acrfullpl[2][{}]{%  
6481 \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6482 {\@acrfullpl{#1}{#2}[]}%  
6483 }
```

Low-level macro:

```
6484 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6485 \acrfullplfmt{#1}{#2}{#3}%  
6486 }
```

\acrfullplfmt No case change plural full format.

```
6487 \newcommand*{\acrfullplfmt}[3]{%  
6488 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6489 }
```



`\Acrfullpl`

```
6490 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}
```

```
6491 \newcommand*\ns@Acrfullpl[2][\%  
6492   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6493   {\@Acrfullpl{#1}{#2}[]}%  
6494 }
```

Low-level macro:

```
6495 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6496   \Acrfullplfmt{#1}{#2}{#3}%  
6497 }
```

`\Acrfullplfmt` First letter upper case plural full format.

```
6498 \newcommand*{\Acrfullplfmt}[3]{%  
6499   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6500 }
```

`\ACRfullpl`

```
6501 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}
```

```
6502 \newcommand*\ns@ACRfullpl[2][\%  
6503   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%  
6504   {\@ACRfullpl{#1}{#2}[]}%  
6505 }
```

Low-level macro:

```
6506 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6507   \ACRfullplfmt{#1}{#2}{#3}%  
6508 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6509 \newcommand*{\ACRfullplfmt}[3]{%  
6510   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6511 }
```

## 1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6512 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6513 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}
```

`\acronymformat` The styles that allow an additional description use `\acronymformat{<short>}{<long>}` to determine what information is displayed in the name.

```
6514 \newcommand*{\acronymformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6515 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6516 \newtoks\glslabeltok
```

`\glsshorttok`

```
6517 \newtoks\glsshorttok
```

`\glslongtok`

```
6518 \newtoks\glslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
6519 \newcommand*{\newacronymhook}{}%
```

`\genericNewAcronym` New improved version of setting the acronym style.

```
6520 \newcommand*{\SetGenericNewAcronym}{%
  Change the behaviour of \Glsentryname to workaround expansion issues that cause a problem for \makefirstuc
  6521 \let\@Gls@entryname\@Gls@acrentryname
  Change the way acronyms are defined:
  6522 \renewcommand{\newacronym}[4][\%
  6523 \ifdefempty{\@glsacronymlists}%
  6524 {\%
  6525 \def\@glo@type{\acronymtype}%
  6526 \setkeys{glossentry}{##1}%
  6527 \DeclareAcronymList{\@glo@type}%
  6528 }%
  6529 }%
  6530 \glskeylisttok{##1}%
  6531 \glslabeltok{##2}%
  6532 \glsshorttok{##3}%
  6533 \glslongtok{##4}%
  6534 \newacronymhook
  6535 \protected@edef\@do@newglossaryentry{%
  6536 \noexpand\newglossaryentry{\the\glslabeltok}%
  6537 {%
  6538 type=\acronymtype,%
  6539 name={\expandonce{\acronymentry{##2}}},%
  6540 sort={\acronymssort{\the\glsshorttok}{\the\glslongtok}},%
  6541 text={\the\glsshorttok},%
```

```

6542      short={\the\glsshorttok},%
6543      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6544      long={\the\glslongtok},%
6545      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6546      \GenericAcronymFields,%
6547      \the\glskeylisttok
6548    }%
6549  }%
6550  \@do@newglossaryentry
6551 }%

```

Make sure that \acrfull etc reflects the new style:

```

6552 \renewcommand*{\acrfullfmt}[3]{%
6553   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6554 \renewcommand*{\Acrfullfmt}[3]{%
6555   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6556 \renewcommand*{\ACRfullfmt}[3]{%
6557   \glslink[##1]{##2}{%
6558     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6559 \renewcommand*{\acrfullplfmt}[3]{%
6560   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6561 \renewcommand*{\Acrfullplfmt}[3]{%
6562   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6563 \renewcommand*{\ACRfullplfmt}[3]{%
6564   \glslink[##1]{##2}{%
6565     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6566 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6567 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6568 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6569 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6570 }

```

`\icAcronymFields` Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

6571 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6572 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6573 \newcommand*{\acronymsort}[2]{##1}

```

`\setacronymstyle`      `\setacronymstyle{<style name>}`

```

6574 \newcommand*{\setacronymstyle}[1]{%
6575   \ifcsundef{@glsacr@dispstyle@#1}%
6576   {%
6577     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6578   }%
6579   {%
6580     \ifdefempty{@glsacronymlists}%
6581     {%
6582       \DeclareAcronymList{\acronymtype}%
6583     }%
6584     {}%
6585     \SetGenericNewAcronym
6586     \GlsUseAcrStyleDefs{#1}%
6587     \@for\@gls@type:=\@glsacronymlists\do{%
6588       \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6589     }%
6590   }%
6591 }

```

`\newacronymstyle`      `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```

6592 \newcommand*{\newacronymstyle}[3]{%
6593   \ifcsdef{@glsacr@dispstyle@#1}%
6594   {%
6595     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6596   }%
6597   {%
6598     \csdef{@glsacr@dispstyle@#1}{#2}%
6599     \csdef{@glsacr@styledefs@#1}{#3}%
6600   }%
6601 }

```

`\renewacronymstyle`      Redefines the given acronym style.

```

6602 \newcommand*{\renewacronymstyle}[3]{%
6603   \ifcsdef{@glsacr@dispstyle@#1}%
6604   {%
6605     \csdef{@glsacr@dispstyle@#1}{#2}%
6606     \csdef{@glsacr@styledefs@#1}{#3}%
6607   }%
6608   {%
6609     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6610   }%
6611 }

```

rEntryDispStyle

```
6612 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

UseAcrStyleDefs

```
6613 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

long-short    *<long>* (*<short>*) acronym style.

```
6614 \newacronymstyle{long-short}%
```

```
6615 {%
```

Check for long form in case this is a mixed glossary.

```
6616 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
```

```
6617 }%
```

```
6618 {%
```

```
6619 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

```
6620 \renewcommand*{\genacrfullformat}[2]{%
```

```
6621 \glsentrylong{##1}##2\space
```

```
6622 (\protect\firstacronymfont{\glsentryshort{##1}})%
```

```
6623 }%
```

```
6624 \renewcommand*{\Genacrfullformat}[2]{%
```

```
6625 \Glsentrylong{##1}##2\space
```

```
6626 (\protect\firstacronymfont{\glsentryshort{##1}})%
```

```
6627 }%
```

```
6628 \renewcommand*{\genplacrfullformat}[2]{%
```

```
6629 \glsentrylongpl{##1}##2\space
```

```
6630 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
```

```
6631 }%
```

```
6632 \renewcommand*{\Genplacrfullformat}[2]{%
```

```
6633 \Glsentrylongpl{##1}##2\space
```

```
6634 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
```

```
6635 }%
```

```
6636 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
```

```
6637 \renewcommand*{\acronymsort}[2]{##1}%
```

```
6638 \renewcommand*{\acronymfont}[1]{##1}%
```

```
6639 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
```

```
6640 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
```

```
6641 }
```

long-sp-short    Similar to the previous style but allows the space between the long and short form to be customized.

```
6642 \newacronymstyle{long-sp-short}%
```

```
6643 {%
```

Check for long form in case this is a mixed glossary.

```
6644 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
```

```
6645 }%
```

```
6646 {%
```

```
6647 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

```

6648 \renewcommand*{\genacrfullformat}[2]{%
6649 \glsentrylong{##1}##2\glsacspace{##1}%
6650 (\protect\firstacronymfont{\glsentryshort{##1}})%
6651 }%
6652 \renewcommand*{\Genacrfullformat}[2]{%
6653 \Glsentrylong{##1}##2\glsacspace{##1}%
6654 (\protect\firstacronymfont{\glsentryshort{##1}})%
6655 }%
6656 \renewcommand*{\genplacrfullformat}[2]{%
6657 \glsentrylongpl{##1}##2\glsacspace{##1}%
6658 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6659 }%
6660 \renewcommand*{\Genplacrfullformat}[2]{%
6661 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6662 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6663 }%
6664 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6665 \renewcommand*{\acronymsort}[2]{##1}%
6666 \renewcommand*{\acronymfont}[1]{##1}%
6667 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6668 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6669 }

```

`\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.

```

6670 \newcommand*{\glsacspace}[1]{%
6671 \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{##1}})}%
6672 \ifdim\dimen@<3em~\else\space\fi
6673 }

```

`short-long` (*short*) (*long*) acronym style.

```

6674 \newacronymstyle{short-long}%
6675 {%
  Check for long form in case this is a mixed glossary.
6676 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6677 }%
6678 {%
6679 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6680 \renewcommand*{\genacrfullformat}[2]{%
6681 \protect\firstacronymfont{\glsentryshort{##1}}##2\space
6682 (\glsentrylong{##1})%
6683 }%
6684 \renewcommand*{\Genacrfullformat}[2]{%
6685 \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6686 (\glsentrylong{##1})%
6687 }%
6688 \renewcommand*{\genplacrfullformat}[2]{%
6689 \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space

```

```

6690 (\glsentrylongpl{##1})%
6691 }%
6692 \renewcommand*{\Genplacrfullformat}[2]{%
6693 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6694 (\glsentrylongpl{##1})%
6695 }%

6696 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6697 \renewcommand*{\acronymsort}[2]{##1}%
6698 \renewcommand*{\acronymfont}[1]{##1}%
6699 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6700 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6701 }

```

long-sc-short    *<long>* (\textsc{<short>}) acronym style.

```

6702 \newacronymstyle{long-sc-short}%
6703 {%
6704 \GlsUseAcrEntryDisplayStyle{long-short}%
6705 }%
6706 {%
6707 \GlsUseAcrStyleDefs{long-short}%
6708 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6709 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6710 }

```

long-sm-short    *<long>* (\textsmaller{<short>}) acronym style.

```

6711 \newacronymstyle{long-sm-short}%
6712 {%
6713 \GlsUseAcrEntryDisplayStyle{long-short}%
6714 }%
6715 {%
6716 \GlsUseAcrStyleDefs{long-short}%
6717 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6718 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6719 }

```

sc-short-long    *<short>* (\textsc{<long>}) acronym style.

```

6720 \newacronymstyle{sc-short-long}%
6721 {%
6722 \GlsUseAcrEntryDisplayStyle{short-long}%
6723 }%
6724 {%
6725 \GlsUseAcrStyleDefs{short-long}%
6726 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6727 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6728 }

```

sm-short-long    *<short>* (\textsmaller{<long>}) acronym style.

```

6729 \newacronymstyle{sm-short-long}%

```

```

6730 {%
6731   \GlsUseAcrEntryDisplayStyle{short-long}%
6732 }%
6733 {%
6734   \GlsUseAcrStyleDefs{short-long}%
6735   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6736   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6737 }

```

long-short-desc    *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6738 \newacronymstyle{long-short-desc}%
6739 {%
6740   \GlsUseAcrEntryDisplayStyle{long-short}%
6741 }%
6742 {%
6743   \GlsUseAcrStyleDefs{long-short}%
6744   \renewcommand*{\GenericAcronymFields}{}%
6745   \renewcommand*{\acronymsort}[2]{##2}%
6746   \renewcommand*{\acronymentry}[1]{%
6747     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6748 }

```

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.

```

6749 \newacronymstyle{long-sp-short-desc}%
6750 {%
6751   \GlsUseAcrEntryDisplayStyle{long-sp-short}%
6752 }%
6753 {%
6754   \GlsUseAcrStyleDefs{long-sp-short}%
6755   \renewcommand*{\GenericAcronymFields}{}%
6756   \renewcommand*{\acronymsort}[2]{##2}%
6757   \renewcommand*{\acronymentry}[1]{%
6758     \glsentrylong{##1}\glsacspace{##1}(\acronymfont{\glsentryshort{##1}})}%
6759 }

```

g-sc-short-desc    *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6760 \newacronymstyle{long-sc-short-desc}%
6761 {%
6762   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
6763 }%
6764 {%
6765   \GlsUseAcrStyleDefs{long-sc-short}%
6766   \renewcommand*{\GenericAcronymFields}{}%
6767   \renewcommand*{\acronymsort}[2]{##2}%
6768   \renewcommand*{\acronymentry}[1]{%
6769     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%

```



6770 }

g-sm-short-desc    *⟨long⟩* (\textsmaller{⟨*short⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```
6771 \newacronymstyle{long-sm-short-desc}%
6772 {%
6773   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
6774 }%
6775 {%
6776   \GlsUseAcrStyleDefs{long-sm-short}%
6777   \renewcommand*{\GenericAcronymFields}{}%
6778   \renewcommand*{\acronymsort}[2]{##2}%
6779   \renewcommand*{\acronymentry}[1]{%
6780     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6781 }
```

short-long-desc    *⟨short⟩* ({⟨*long⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```
6782 \newacronymstyle{short-long-desc}%
6783 {%
6784   \GlsUseAcrEntryDisplayStyle{short-long}%
6785 }%
6786 {%
6787   \GlsUseAcrStyleDefs{short-long}%
6788   \renewcommand*{\GenericAcronymFields}{}%
6789   \renewcommand*{\acronymsort}[2]{##2}%
6790   \renewcommand*{\acronymentry}[1]{%
6791     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6792 }
```

short-long-desc    *⟨long⟩* (\textsc{⟨*short⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```
6793 \newacronymstyle{sc-short-long-desc}%
6794 {%
6795   \GlsUseAcrEntryDisplayStyle{sc-short-long}%
6796 }%
6797 {%
6798   \GlsUseAcrStyleDefs{sc-short-long}%
6799   \renewcommand*{\GenericAcronymFields}{}%
6800   \renewcommand*{\acronymsort}[2]{##2}%
6801   \renewcommand*{\acronymentry}[1]{%
6802     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6803 }
```

short-long-desc    *⟨long⟩* (\textsmaller{⟨*short⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```
6804 \newacronymstyle{sm-short-long-desc}%
6805 {%
```

```

6806 \GlsUseAcrEntryDispStyle{sm-short-long}%
6807 }%
6808 {%
6809 \GlsUseAcrStyleDefs{sm-short-long}%
6810 \renewcommand*{\GenericAcronymFields}{}%
6811 \renewcommand*{\acronymsort}[2]{##2}%
6812 \renewcommand*{\acronymentry}[1]{%
6813 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6814 }

```

dua *<long>* only acronym style.

```

6815 \newacronymstyle{dua}%
6816 {%

```

Check for long form in case this is a mixed glossary.

```

6817 \ifdefempty\glscustomtext
6818 {%
6819 \ifglshaslong{\glslabel}%
6820 {%
6821 \glsifplural
6822 {%

```

Plural form:

```

6823 \glscapscase
6824 {%

```

Plural form, don't adjust case:

```

6825 \glentrylongpl{\glslabel}\glsinsert
6826 }%
6827 {%

```

Plural form, make first letter upper case:

```

6828 \Glsentrylongpl{\glslabel}\glsinsert
6829 }%
6830 {%

```

Plural form, all caps:

```

6831 \mfirstucMakeUppercase
6832 {\glentrylongpl{\glslabel}\glsinsert}%
6833 }%
6834 }%
6835 {%

```

Singular form

```

6836 \glscapscase
6837 {%

```

Singular form, don't adjust case:

```

6838 \glentrylong{\glslabel}\glsinsert
6839 }%
6840 {%

```

Subsequent singular form, make first letter upper case:

```
6841      \Glsentrylong{\glslabel}\glsinsert
6842      }%
6843      {%
```

Subsequent singular form, all caps:

```
6844      \mfirstucMakeUppercase
6845      {\glsentrylong{\glslabel}\glsinsert}%
6846      }%
6847      }%
6848      }%
6849      {%
```

Not an acronym:

```
6850      \glsentryfmt
6851      }%
6852      }%
6853      {\glscustomtext\glsinsert}%
6854      }%
6855      {%
6856      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6857      \renewcommand*{\acrfullfmt}[3]{%
6858      \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6859      (\acronymfont{\glsentryshort{##2}})}}%
6860      \renewcommand*{\Acrfullfmt}[3]{%
6861      \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6862      (\acronymfont{\glsentryshort{##2}})}}%
6863      \renewcommand*{\ACRfullfmt}[3]{%
6864      \glslink[##1]{##2}{%
6865      \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6866      (\acronymfont{\glsentryshort{##2}})}}}%

6867      \renewcommand*{\acrfullplfmt}[3]{%
6868      \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6869      (\acronymfont{\glsentryshortpl{##2}})}}%

6870      \renewcommand*{\Acrfullplfmt}[3]{%
6871      \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6872      (\acronymfont{\glsentryshortpl{##2}})}}%
6873      \renewcommand*{\ACRfullplfmt}[3]{%
6874      \glslink[##1]{##2}{%
6875      \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6876      (\acronymfont{\glsentryshortpl{##2}})}}}%
6877      \renewcommand*{\glsentryfull}[1]{%
6878      \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6879      }%
6880      \renewcommand*{\Glsentryfull}[1]{%
6881      \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6882      }%
```

```

6883 \renewcommand*{\glsentryfullpl}[1]{%
6884   \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6885 }%
6886 \renewcommand*{\Glsentryfullpl}[1]{%
6887   \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6888 }%
6889 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6890 \renewcommand*{\acronymsort}[2]{##1}%
6891 \renewcommand*{\acronymfont}[1]{##1}%
6892 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6893 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6894 \newacronymstyle{dua-desc}%
6895 {%
6896   \GlsUseAcrEntryDisplayStyle{dua}%
6897 }%
6898 {%
6899   \GlsUseAcrStyleDefs{dua}%
6900   \renewcommand*{\GenericAcronymFields}{}%
6901   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6902   \renewcommand*{\acronymsort}[2]{##2}%
6903 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6904 \newacronymstyle{footnote}%
6905 {%
6906   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6907 }%
6908 {%
6909   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6910   \glshyperfirstfalse
6911   \renewcommand*{\genacrfullformat}[2]{%
6912     \protect\firstacronymfont{\glsentryshort{##1}}##2%
6913     \protect\footnote{\glsentrylong{##1}}%
6914   }%
6915   \renewcommand*{\Genacrfullformat}[2]{%
6916     \firstacronymfont{\Glsentryshort{##1}}##2%
6917     \protect\footnote{\glsentrylong{##1}}%
6918   }%
6919   \renewcommand*{\genplacrfullformat}[2]{%
6920     \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
6921     \protect\footnote{\glsentrylongpl{##1}}%
6922   }%
6923   \renewcommand*{\Genplacrfullformat}[2]{%

```

```

6924 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6925 \protect\footnote{\glsentrylongpl{##1}}%
6926 }%
6927 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6928 \renewcommand*{\acronymsort}[2]{##1}%
6929 \renewcommand*{\acronymfont}[1]{##1}%
6930 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6931 \renewcommand*{\acrfullfmt}[3]{%
6932 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
6933 (\glsentrylong{##2})}%
6934 \renewcommand*{\Acrfullfmt}[3]{%
6935 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6936 (\glsentrylong{##2})}%
6937 \renewcommand*{\ACRfullfmt}[3]{%
6938 \glslink[##1]{##2}{%
6939 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
6940 (\glsentrylong{##2})}}}%
6941 \renewcommand*{\acrfullplfmt}[3]{%
6942 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
6943 (\glsentrylongpl{##2})}%
6944 \renewcommand*{\Acrfullplfmt}[3]{%
6945 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6946 (\glsentrylongpl{##2})}%
6947 \renewcommand*{\ACRfullplfmt}[3]{%
6948 \glslink[##1]{##2}{%
6949 \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
6950 (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

6951 \renewcommand*{\glsentryfull}[1]{%
6952 \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
6953 \renewcommand*{\Glsentryfull}[1]{%
6954 \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
6955 \renewcommand*{\glsentryfullpl}[1]{%
6956 \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6957 \renewcommand*{\Glsentryfullpl}[1]{%
6958 \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6959 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6960 \newacronymstyle{footnote-sc}%
6961 {%
6962 \GlsUseAcrEntryDispStyle{footnote}%
6963 }%
6964 {%
6965 \GlsUseAcrStyleDefs{footnote}%
6966 \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6967 \renewcommand{\acronymfont}[1]{\textsc{##1}}%

```

```

6968 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6969 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6970 \newacronymstyle{footnote-sm}%
6971 {%
6972 \GlsUseAcrEntryDisplayStyle{footnote}%
6973 }%
6974 {%
6975 \GlsUseAcrStyleDefs{footnote}%
6976 \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6977 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6978 \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6979 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6980 \newacronymstyle{footnote-desc}%
6981 {%
6982 \GlsUseAcrEntryDisplayStyle{footnote}%
6983 }%
6984 {%
6985 \GlsUseAcrStyleDefs{footnote}%
6986 \renewcommand*{\GenericAcronymFields}{}%
6987 \renewcommand*{\acronymsort}[2]{##2}%
6988 \renewcommand*{\acronymentry}[1]{%
6989 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6990 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6991 \newacronymstyle{footnote-sc-desc}%
6992 {%
6993 \GlsUseAcrEntryDisplayStyle{footnote-sc}%
6994 }%
6995 {%
6996 \GlsUseAcrStyleDefs{footnote-sc}%
6997 \renewcommand*{\GenericAcronymFields}{}%
6998 \renewcommand*{\acronymsort}[2]{##2}%
6999 \renewcommand*{\acronymentry}[1]{%
7000 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
7001 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

7002 \newacronymstyle{footnote-sm-desc}%
7003 {%
7004 \GlsUseAcrEntryDisplayStyle{footnote-sm}%

```

```

7005 }%
7006 {%
7007   \GlsUseAcrStyleDefs{footnote-sm}%
7008   \renewcommand*{\GenericAcronymFields}{}%
7009   \renewcommand*{\acronymsort}[2]{##2}%
7010   \renewcommand*{\acronymentry}[1]{%
7011     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
7012 }

```

## AcronymSynonyms

```

7013 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

```
\acs
```

```
7014 \let\acs\acrshort
```

First letter uppercase short form

```
\Acs
```

```
7015 \let\Acs\Acrshort
```

Plural short form

```
\acsp
```

```
7016 \let\acsp\acrshortpl
```

First letter uppercase plural short form

```
\Acsp
```

```
7017 \let\Acsp\Acrshortpl
```

Long form

```
\acl
```

```
7018 \let\acl\aclong
```

Plural long form

```
\aclp
```

```
7019 \let\aclp\aclongpl
```

First letter upper case long form

```
\Acl
```

```
7020 \let\Acl\Aclong
```

First letter upper case plural long form

```
\Aclp
```

```
7021 \let\Aclp\Aclongpl
```

Full form

`\acf`

7022 `\let\acf\acrfull`

Plural full form

`\acfp`

7023 `\let\acfp\acrfullpl`

First letter upper case full form

`\Acf`

7024 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

7025 `\let\Acfp\Acrfullpl`

Standard form

`\ac`

7026 `\let\ac\gls`

First upper case standard form

`\Ac`

7027 `\let\Ac\Gls`

Standard plural form

`\acp`

7028 `\let\acp\glspl`

Standard first letter upper case plural form

`\Acp`

7029 `\let\Acp\Glspl`

7030 }

Define synonyms if required

7031 `\ifglsacrshortcuts`

7032 `\DefineAcronymSynonyms`

7033 `\fi`

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\AcronymDisplayStyle` Sets the default acronym display style for given glossary.

7034 `\newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%`

7035 `\defglsentryfmt[#1]{\glsentryfmt}}%`

7036 }



`\letNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```

7037 \newcommand*{\DefaultNewAcronymDef}{%
7038   \edef\@do@newglossaryentry{%
7039     \noexpand\newglossaryentry{\the\glslabeltok}%
7040     {%
7041       type=\acronymtype,%
7042       name={\the\glsshorttok},%
7043       sort={\the\glsshorttok},%
7044       text={\the\glsshorttok},%
7045       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7046       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7047       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7048                   {\noexpand\expandonce\noexpand\@glo@shortpl}},%
7049       short={\the\glsshorttok},%
7050       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7051       long={\the\glslongtok},%
7052       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7053       description={\the\glslongtok},%
7054       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

Remaining options specified by the user:

```

7055     \the\glskeylisttok
7056   }%
7057 }%
7058 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7059 \let\@org@gls@assign@plural\gls@assign@plural
7060 \let\@org@gls@assign@descplural\gls@assign@descplural
7061 \def\gls@assign@firstpl##1##2{%
7062   \@gls@expand@field{##1}{firstpl}{##2}%
7063 }%
7064 \def\gls@assign@plural##1##2{%
7065   \@gls@expand@field{##1}{plural}{##2}%
7066 }%
7067 \def\gls@assign@descplural##1##2{%
7068   \@gls@expand@field{##1}{descplural}{##2}%
7069 }%
7070 \@do@newglossaryentry
7071 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7072 \let\gls@assign@plural\@org@gls@assign@plural
7073 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7074 }

```

`\letAcronymStyle` Set up the default acronym style:

```

7075 \newcommand*{\SetDefaultAcronymStyle}{%
  Set the display style:
7076   \@for\@gls@type:=\@glsacronymlists\do{%
7077     \SetDefaultAcronymDisplayStyle{\@gls@type}%
7078   }%

```

Set up the definition of `\newacronym`:

```
7079 \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).

```
7080 \ifx\@glsacronymlists\@empty
7081 \def\@glo@type{\acronymtype}%
7082 \setkeys{glossentry}{##1}%
7083 \DeclareAcronymList{\@glo@type}%
7084 \SetDefaultAcronymDisplayStyle{\@glo@type}%
7085 \fi
7086 \glskeylisttok{##1}%
7087 \glslabeltok{##2}%
7088 \glsshorttok{##3}%
7089 \gslongtok{##4}%
7090 \newacronymhook
7091 \DefaultNewAcronymDef
7092 }%
7093 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7094 }
```

`\acrfootnote` Used by the footnote acronym styles.

```
7095 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

`acrlinkfootnote`

```
7096 \newcommand*{\acrlinkfootnote}[3]{%
7097 \footnote{\glslink{#1}{#2}{#3}}%
7098 }
```

`crnolinkfootnote`

```
7099 \newcommand*{\acrlinkfootnote}[3]{%
7100 \footnote{#3}%
7101 }
```

`nymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```
7102 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7103 \defglsentryfmt{#1}{%
7104 \ifdefempty\glscustomtext
7105 {%
7106 \ifglsused{\glslabel}%
7107 {%
7108 \acronymfont{\glsentryfmt}%
7109 }%
7110 {%
7111 \firstacronymfont{\glsentryfmt}%
7112 \ifglsymbol{\glslabel}%
7113 {%
```

```

7114         \expandafter\protect\expandafter\acrfootnote\expandafter
7115         {\@gls@link@opts}{\@gls@link@label}%
7116         {%
7117         \glsifplural
7118         {\glsentrysymbolplural{\glslabel}}%
7119         {\glsentrysymbol{\glslabel}}%
7120         }%
7121     }%
7122 }%
7123 }%
7124 {\glscustomtext\glsinsert}%
7125 }%
7126 }

```

teNewAcronymDef

```

7127 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7128 \edef\@do@newglossaryentry{%
7129 \noexpand\newglossaryentry{\the\glslabeltok}%
7130 {%
7131     type=\acronymtype,%
7132     name={\noexpand\acronymfont{\the\glsshorttok}},%
7133     sort={\the\glsshorttok},%
7134     first={\the\glsshorttok},%
7135     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7136     text={\the\glsshorttok},%
7137     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7138     short={\the\glsshorttok},%
7139     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7140     long={\the\glslongtok},%
7141     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7142     symbol={\the\glslongtok},%
7143     symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7144     \the\glskeylisttok
7145 }%
7146 }%
7147 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7148 \let\@org@gls@assign@plural\gls@assign@plural
7149 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7150 \def\gls@assign@firstpl##1##2{%
7151     \@gls@expand@field{##1}{firstpl}{##2}%
7152 }%
7153 \def\gls@assign@plural##1##2{%
7154     \@gls@expand@field{##1}{plural}{##2}%
7155 }%
7156 \def\gls@assign@symbolplural##1##2{%
7157     \@gls@expand@field{##1}{symbolplural}{##2}%
7158 }%
7159 \@do@newglossaryentry
7160 \let\gls@assign@plural\@org@gls@assign@plural

```

```

7161 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7162 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7163 }

```

`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.

```

7164 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
7165   \renewcommand{\newacronym}[4][\]{%
7166     \ifx\@glsacronymlists\@empty
7167       \def\@glo@type{\acronymtype}%
7168       \setkeys{glossentry}{##1}%
7169       \DeclareAcronymList{\@glo@type}%
7170       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7171     \fi
7172     \glskeylisttok{##1}%
7173     \glslabeltok{##2}%
7174     \glsshorttok{##3}%
7175     \glslongtok{##4}%
7176     \newacronymhook
7177     \DescriptionFootnoteNewAcronymDef
7178   }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7179 \@for\@gls@type:=\@glsacronymlists\do{%
7180   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7181 }%

```

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.

```

7182 \ifglsacrsmallcaps
7183   \renewcommand*\acronymfont[1]{\textsc{##1}}%
7184   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7185 \else
7186   \ifglsacrsmaller
7187     \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
7188   \fi
7189 \fi

```

Check for package option clash

```

7190 \ifglsacrdua
7191   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7192     can’t both be set}{}%
7193 \fi
7194 }%

```

`nymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.

```

7195 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7196   \def\glsentryfmt[#1]{\glsentryfmt}%
7197 }

```

#### UANewAcronymDef

```

7198 \newcommand*{\DescriptionDUANewAcronymDef}{%
7199   \edef\@do@newglossaryentry{%
7200     \noexpand\newglossaryentry{\the\glslabeltok}%
7201     {%
7202       type=\acronymtype,%
7203       name={\the\glslongtok},%
7204       sort={\the\glslongtok},%
7205       text={\the\glslongtok},%
7206       first={\the\glslongtok},%
7207       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7208       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7209       short={\the\glsshorttok},%
7210       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7211       long={\the\glslongtok},%
7212       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7213       symbol={\the\glsshorttok},%
7214       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7215       \the\glskeylisttok
7216     }%
7217   }%
7218   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7219   \let\@org@gls@assign@plural\gls@assign@plural
7220   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7221   \def\gls@assign@firstpl##1##2{%
7222     \@@gls@expand@field{##1}{firstpl}{##2}%
7223   }%
7224   \def\gls@assign@plural##1##2{%
7225     \@@gls@expand@field{##1}{plural}{##2}%
7226   }%
7227   \def\gls@assign@symbolplural##1##2{%
7228     \@@gls@expand@field{##1}{symbolplural}{##2}%
7229   }%
7230   \@do@newglossaryentry
7231   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7232   \let\gls@assign@plural\@org@gls@assign@plural
7233   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7234 }

```

**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.

```

7235 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
7236   \ifglssacrsmallcaps
7237     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'

```

```

7238     can't both be set}{}%
7239 \else
7240     \ifglsmaller
7241         \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7242             can't both be set}{}%
7243     \fi
7244 \fi
7245 \renewcommand{\newacronym}[4][]{%
7246     \ifx\@glsacronymlists\@empty
7247         \def\@glo@type{\acronymtype}%
7248         \setkeys{glossentry}{##1}%
7249         \DeclareAcronymList{\@glo@type}%
7250         \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
7251     \fi
7252     \glskeylisttok{##1}%
7253     \glslabeltok{##2}%
7254     \glsshorttok{##3}%
7255     \glslongtok{##4}%
7256     \newacronymhook
7257     \DescriptionDUANewAcronymDef
7258 }%

Set display.
7259 \@for\@gls@type:=\@glsacronymlists\do{%
7260     \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
7261 }%
7262 }%

```

`\newacronymDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7263 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7264     \defglentryfmt[#1]{%

7265         \ifdefempty\glscustomtext
7266         {%
7267             \ifglused{\glslabel}%
7268             {%

Move the inserted text outside of \acronymfont
7269                 \let\gls@org@insert\glsinsert
7270                 \let\glsinsert\@empty
7271                 \acronymfont{\glsgenentryfmt}\gls@org@insert
7272             }%
7273         }%
7274         \glsgenentryfmt
7275         \ifglshassymbol{\glslabel}%
7276         {%
7277             \glsifplural
7278             {%
7279                 \def\@glo@symbol{\glstrysymbolplural{\glslabel}}%

```

```

7280         }%
7281         {%
7282         \def\@glo@symbol{\glsentrysymbol{\glslabel}}}%
7283         }%
7284         \space(\protect\firstacronymfont
7285         {\glscapscase
7286         {\@glo@symbol}
7287         {\@glo@symbol}
7288         {\mfirstucMakeUppercase{\@glo@symbol}}})}%
7289     }%
7290     {}%
7291 }%
7292 }%
7293 {\glscustomtext\glsinsert}%
7294 }%
7295 }

```

onNewAcronymDef

```

7296 \newcommand*{\DescriptionNewAcronymDef}{%
7297 \edef\@do@newglossaryentry{%
7298 \noexpand\newglossaryentry{\the\glslabeltok}%
7299 {%
7300 type=\acronymtype,%
7301 name={\noexpand
7302 \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
7303 sort={\the\glsshorttok},%
7304 first={\the\glslongtok},%
7305 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7306 text={\the\glsshorttok},%
7307 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7308 short={\the\glsshorttok},%
7309 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7310 long={\the\glslongtok},%
7311 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7312 symbol={\noexpand\@glo@text},%
7313 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7314 \the\glskeylisttok}%
7315 }%
7316 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7317 \let\@org@gls@assign@plural\gls@assign@plural
7318 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7319 \def\gls@assign@firstpl##1##2{%
7320 \@@gls@expand@field{##1}{firstpl}{##2}%
7321 }%
7322 \def\gls@assign@plural##1##2{%
7323 \@@gls@expand@field{##1}{plural}{##2}%
7324 }%
7325 \def\gls@assign@symbolplural##1##2{%
7326 \@@gls@expand@field{##1}{symbolplural}{##2}%

```

```

7327 }%
7328 \@do@newglossaryentry
7329 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7330 \let\gls@assign@plural\@org@gls@assign@plural
7331 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7332 }

```

**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.

```

7333 \newcommand*{\SetDescriptionAcronymStyle}{%
7334   \renewcommand{\newacronym}[4][]{%
7335     \ifx\@glsacronymlists\@empty
7336       \def\@glo@type{\acronymtype}%
7337       \setkeys{glossentry}{##1}%
7338       \DeclareAcronymList{\@glo@type}%
7339       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7340     \fi
7341     \glskeylisttok{##1}%
7342     \glslabeltok{##2}%
7343     \glsshorttok{##3}%
7344     \glslongtok{##4}%
7345     \newacronymhook
7346     \DescriptionNewAcronymDef
7347   }%

```

Set display.

```

7348 \@for\@gls@type:=\@glsacronymlists\do{%
7349   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7350 }%

```

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.

```

7351 \ifglsacrsmallcaps
7352   \renewcommand{\acronymfont}[1]{\textsc{##1}}
7353   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7354 \else
7355   \ifglsacrsmaller
7356     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7357   \fi
7358 \fi
7359 }%

```

**nymDisplayStyle** Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7360 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7361   \defglsentryfmt[##1]{%
7362     \ifdefempty\glscustomtext
7363     {%

```



Move the inserted text outside of \acronymfont

```

7364 \let\gls@org@insert\glsinsert
7365 \let\glsinsert\@empty
7366 \ifglsused{\glslabel}%
7367 {%
7368 \acronymfont{\glsentryfmt}\gls@org@insert
7369 }%
7370 {%
7371 \firstacronymfont{\glsentryfmt}\gls@org@insert
7372 \ifglschaslong{\glslabel}%
7373 {%
7374 \expandafter\protect\expandafter\acrfootnote\expandafter
7375 {\@gls@link@opts}{\@gls@link@label}%
7376 {%
7377 \glsifplural
7378 {\glsentrylongpl{\glslabel}}%
7379 {\glsentrylong{\glslabel}}%
7380 }%
7381 }%
7382 {}%
7383 }%
7384 }%
7385 {\glscustomtext\glsinsert}%
7386 }%
7387 }

```

teNewAcronymDef

```

7388 \newcommand*{\FootnoteNewAcronymDef}{%
7389 \edef\@do@newglossaryentry{%
7390 \noexpand\newglossaryentry{\the\glslabeltok}%
7391 {%
7392 type=\acronymtype,%
7393 name={\noexpand\acronymfont{\the\glsshorttok}},%
7394 sort={\the\glsshorttok},%
7395 text={\the\glsshorttok},%
7396 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7397 first={\the\glsshorttok},%
7398 firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7399 short={\the\glsshorttok},%
7400 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7401 long={\the\glslongtok},%
7402 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7403 description={\the\glslongtok},%
7404 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7405 \the\glskeylisttok
7406 }%
7407 }%
7408 \let\@org@gls@assign@plural\gls@assign@plural

```

```

7409 \let\@org@gl@s@assign@firstpl\gl@s@assign@firstpl
7410 \let\@org@gl@s@assign@descplural\gl@s@assign@descplural
7411 \def\gl@s@assign@firstpl##1##2{%
7412   \@@gl@s@expand@field{##1}{firstpl}{##2}%
7413 }%
7414 \def\gl@s@assign@plural##1##2{%
7415   \@@gl@s@expand@field{##1}{plural}{##2}%
7416 }%
7417 \def\gl@s@assign@descplural##1##2{%
7418   \@@gl@s@expand@field{##1}{descplural}{##2}%
7419 }%
7420 \do@newglossaryentry
7421 \let\gl@s@assign@plural\@org@gl@s@assign@plural
7422 \let\gl@s@assign@firstpl\@org@gl@s@assign@firstpl
7423 \let\gl@s@assign@descplural\@org@gl@s@assign@descplural
7424 }

```

**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.

```

7425 \newcommand*{\SetFootnoteAcronymStyle}{%
7426   \renewcommand{\newacronym}[4][]{%
7427     \ifx\@gl@s@acronymlists\@empty
7428       \def\@glo@type{\acronymtype}%
7429       \setkeys{glossentry}{##1}%
7430       \DeclareAcronymList{\@glo@type}%
7431       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7432     \fi
7433     \glskeylisttok{##1}%
7434     \glslabeltok{##2}%
7435     \glsshorttok{##3}%
7436     \glslongtok{##4}%
7437     \newacronymhook
7438     \FootnoteNewAcronymDef
7439   }%

```

Set display

```

7440   \@for\@gl@s@type:=\@gl@s@acronymlists\do{%
7441     \SetFootnoteAcronymDisplayStyle{\@gl@s@type}%
7442   }%

```

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.

```

7443   \ifgl@s@crsmallcaps
7444     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7445     \renewcommand*{\acrpluralsuffix}{\gl@supacrpluralsuffix}%
7446   \else
7447     \ifgl@s@crsmaller
7448       \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7449     \fi
7450   \fi

```

Check for option clash

```
7451 \ifglsacrdue
7452 \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
7453 can't both be set}{}%
7454 \fi
7455 }%
```

`\parenifnotempty` Do a space followed by the argument if the argument doesn't expand to empty or `\relax`. If argument isn't empty (or `\relax`), apply the macro to it given in the second argument.

```
7456 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
7457 \protected@edef\gls@tmp{#1}%
7458 \ifdefempty\gls@tmp
7459 {}%
7460 {%
7461 \ifx\gls@tmp\@gls@default@value
7462 \else
7463 \space (#2{#1})%
7464 \fi
7465 }%
7466 }
```

`\nymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
7467 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
7468 \defglsentryfmt[#1]{%
7469 \ifdefempty\glscustomtext
7470 {%
```

Move the inserted text outside of `\acronymfont`

```
7471 \let\gls@org@insert\glsinsert
7472 \let\glsinsert\@empty
7473 \ifglsused{\glslabel}%
7474 {%
7475 \acronymfont{\glsentryfmt}\gls@org@insert
7476 }%
7477 {%
7478 \glsentryfmt
7479 \ifglshassymbol{\glslabel}%
7480 {%
7481 \glsifplural
7482 {%
7483 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7484 }%
7485 {%
7486 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7487 }%
7488 \space
7489 (\glscapscase
```

```

7490         {\firstacronymfont{\@glo@symbol}}}%
7491         {\firstacronymfont{\@glo@symbol}}}%
7492         {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})}%
7493     }%
7494     {}%
7495 }%
7496 }%
7497 {\glscustomtext\glsinsert}%
7498 }%
7499 }

```

# 1.1NewAcronymDef

```

7500 \newcommand*{\SmallNewAcronymDef}{%
7501   \edef\@do@newglossaryentry{%
7502     \noexpand\newglossaryentry{\the\glslabeltok}%
7503     {%
7504       type=\acronymtype,%
7505       name={\noexpand\acronymfont{\the\glsshorttok}},%
7506       sort={\the\glsshorttok},%
7507       text={\the\glsshorttok},%
7508       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7509       first={\the\glslongtok},%
7510       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7511       short={\the\glsshorttok},%
7512       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7513       long={\the\glslongtok},%
7514       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7515       description={\noexpand\@glo@first},%
7516       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7517       symbol={\the\glsshorttok},%
7518       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7519       \the\glskeylisttok
7520     }%
7521   }%
7522   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7523   \let\@org@gls@assign@plural\gls@assign@plural
7524   \let\@org@gls@assign@descplural\gls@assign@descplural
7525   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7526   \def\gls@assign@firstpl##1##2{%
7527     \@gls@expand@field{##1}{firstpl}{##2}%
7528   }%
7529   \def\gls@assign@plural##1##2{%
7530     \@gls@expand@field{##1}{plural}{##2}%
7531   }%

```

```

7532 \def\gls@assign@descplural##1##2{%
7533   \@gls@expand@field{##1}{descplural}{##2}%
7534 }%
7535 \def\gls@assign@symbolplural##1##2{%
7536   \@gls@expand@field{##1}{symbolplural}{##2}%
7537 }%
7538 \do@newglossaryentry
7539 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7540 \let\gls@assign@plural\@org@gls@assign@plural
7541 \let\gls@assign@descplural\@org@gls@assign@descplural
7542 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7543 }

```

`\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.

```

7544 \newcommand*{\SetSmallAcronymStyle}{%
7545   \renewcommand{\newacronym}[4][]{%
7546     \ifx\glsacronymlists\@empty
7547       \def\@glo@type{\acronymtype}%
7548       \setkeys{glossentry}{##1}%
7549       \DeclareAcronymList{\@glo@type}%
7550       \SetSmallAcronymDisplayStyle{\@glo@type}%
7551     \fi
7552     \glskeylisttok{##1}%
7553     \glslabeltok{##2}%
7554     \glsshorttok{##3}%
7555     \glslongtok{##4}%
7556     \newacronymhook
7557     \SmallNewAcronymDef
7558   }%

```

Change the display since first only contains long form.

```

7559 \@for\@gls@type:=\@glsacronymlists\do{%
7560   \SetSmallAcronymDisplayStyle{\@gls@type}%
7561 }%

```

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.

```

7562 \ifglsacrsmallcaps
7563   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7564   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7565 \else
7566   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
7567 \fi

```

check for option clash

```

7568 \ifglsacrdua
7569   \ifglsacrsmallcaps
7570     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7571       can’t both be set}{}%

```

```

7572 \else
7573 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7574 can’t both be set}{}%
7575 \fi
7576 \fi
7577 }%

```

**DUADisplayStyle** Sets the acronym display style for given glossary with dua setting.

```

7578 \newcommand*{\SetDUADisplayStyle}[1]{%
7579 \def\glsentryfmt[#1]{\glsentryfmt}%
7580 }

```

**UANewAcronymDef**

```

7581 \newcommand*{\DUANewAcronymDef}{%
7582 \edef\@do@newglossaryentry{%
7583 \noexpand\newglossaryentry{\the\glslabeltok}%
7584 {%
7585 type=\acronymtype,%
7586 name={\the\glsshorttok},%
7587 text={\the\glslongtok},%
7588 first={\the\glslongtok},%
7589 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7590 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7591 short={\the\glsshorttok},%
7592 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7593 long={\the\glslongtok},%
7594 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7595 description={\the\glslongtok},%
7596 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7597 symbol={\the\glsshorttok},%
7598 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7599 \the\glskeylisttok
7600 }%
7601 }%
7602 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7603 \let\@org@gls@assign@plural\gls@assign@plural
7604 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7605 \let\@org@gls@assign@descplural\gls@assign@descplural
7606 \def\gls@assign@firstpl##1##2{%
7607 \@@gls@expand@field{##1}{firstpl}{##2}%
7608 }%
7609 \def\gls@assign@plural##1##2{%
7610 \@@gls@expand@field{##1}{plural}{##2}%
7611 }%
7612 \def\gls@assign@symbolplural##1##2{%
7613 \@@gls@expand@field{##1}{symbolplural}{##2}%
7614 }%
7615 \def\gls@assign@descplural##1##2{%
7616 \@@gls@expand@field{##1}{descplural}{##2}%

```

```

7617 }%
7618 \@do@newglossaryentry
7619 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7620 \let\gls@assign@plural\@org@gls@assign@plural
7621 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7622 \let\gls@assign@descplural\@org@gls@assign@descplural
7623 }

```

`\SetDUASStyle` Always expand acronyms.

```

7624 \newcommand*{\SetDUASStyle}{%
7625   \renewcommand{\newacronym}[4][]{%
7626     \ifx\@glsacronymlists\@empty
7627       \def\@glo@type{\acronymtype}%
7628       \setkeys{glossentry}{##1}%
7629       \DeclareAcronymList{\@glo@type}%
7630       \SetDUADisplayStyle{\@glo@type}%
7631     \fi
7632     \glskeylisttok{##1}%
7633     \glslabeltok{##2}%
7634     \glsshorttok{##3}%
7635     \glslongtok{##4}%
7636     \newacronymhook
7637     \DUANewAcronymDef
7638   }%
7639   \@for\@gls@type:=\@glsacronymlists\do{%
7640     \SetDUADisplayStyle{\@gls@type}%
7641   }%
7642 }

```

Set the display

`SetAcronymStyle`

```

7643 \newcommand*{\SetAcronymStyle}{%
7644   \SetDefaultAcronymStyle
7645   \ifglsacrdescription
7646     \ifglsacrfootnote
7647       \SetDescriptionFootnoteAcronymStyle
7648     \else
7649       \ifglsacrdua
7650         \SetDescriptionDUAAcronymStyle
7651       \else
7652         \SetDescriptionAcronymStyle
7653     \fi
7654   \fi
7655 \else
7656   \ifglsacrfootnote
7657     \SetFootnoteAcronymStyle
7658   \else
7659     \ifthenelse{\boolean{glsacrsmalldcaps}}{OR
7660       \boolean{glsacrsmaller}}}%

```

```

7661      {%
7662      \SetSmallAcronymStyle
7663      }%
7664      {%
7665      \ifglssacrdua
7666      \SetDUASStyle
7667      \fi
7668      }%
7669      \fi
7670      \fi
7671 }

```

Set the acronym style according to the package options

```

7672 \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.

```

7673 \newcommand*{\SetCustomDisplayStyle}[1]{%
7674   \defglssentryfmt[#1]{\glsgenentryfmt}%
7675 }

```

`\setacronymfields`

```

7676 \newcommand*{\CustomAcronymFields}{%
7677   name={\the\glsshorttok},%
7678   description={\the\glslongtok},%
7679   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7680   firstplural={\acrfullformat
7681     {\noexpand\glssentrylongpl{\the\glslabeltok}}}%
7682     {\noexpand\glssentryshortpl{\the\glslabeltok}}},%
7683   text={\the\glsshorttok},%
7684   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7685 }

```

`\setnewacronym`

```

7686 \newcommand*{\CustomNewAcronymDef}{%
7687   \protected@edef\do@newglossaryentry{%
7688     \noexpand\newglossaryentry{\the\glslabeltok}%
7689     {%
7690       type=\acronymtype,%
7691       short={\the\glsshorttok},%
7692       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7693       long={\the\glslongtok},%
7694       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7695       user1={\the\glsshorttok},%

```



```

7696     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7697     user3={\the\glslongtok},%
7698     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7699     \CustomAcronymFields,%
7700     \the\glskeylisttok
7701   }%
7702 }%
7703 \do@newglossaryentry
7704 }

```

\SetCustomStyle

```

7705 \newcommand*{\SetCustomStyle}{%
7706   \renewcommand{\newacronym}[4][]{%
7707     \ifx\@glsacronymlists\@empty
7708       \def\@glo@type{\acronymtype}%
7709       \setkeys{glossentry}{##1}%
7710       \DeclareAcronymList{\@glo@type}%
7711       \SetCustomDisplayStyle{\@glo@type}%
7712     \fi
7713     \glskeylisttok{##1}%
7714     \glslabeltok{##2}%
7715     \glsshorttok{##3}%
7716     \glslongtok{##4}%
7717     \newacronymhook
7718     \CustomNewAcronymDef
7719   }%
7720   \@for\@gls@type:=\@glsacronymlists\do{%
7721     \SetCustomDisplayStyle{\@gls@type}%
7722   }%
7723 }

```

Set the display

## 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.

```
7724 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
7725 \@gls@loadlist
```

The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
7726 \@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.

```
7727 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7728 \@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.

```
7729 \ifx\@glossary@default@style\relax
```

```
7730 \else
```

```
7731   \setglossarystyle{\@glossary@default@style}
```

```
7732 \fi
```

## 1.20 Debugging Commands

`\showgloparent`    `\showgloparent{<label>}`

```
7733 \newcommand*{\showgloparent}[1]{%
```

```
7734   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7735 }
```

`\showglolevel`    `\showglolevel{<label>}`

```
7736 \newcommand*{\showglolevel}[1]{%
```

```
7737   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7738 }
```

`\showglotext`    `\showglotext{<label>}`

```
7739 \newcommand*{\showglotext}[1]{%
```

```
7740   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7741 }
```

`\showgloplural`    `\showgloplural{<label>}`

```
7742 \newcommand*{\showgloplural}[1]{%
```

```
7743   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
```

```
7744 }
```

`\showglofirst`    `\showglofirst{<label>}`

```

7745 \newcommand*{\showglofirst}[1]{%
7746   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7747 }

```

\showglofirstpl \showglofirstpl{\label{}}

```

7748 \newcommand*{\showglofirstpl}[1]{%
7749   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7750 }

```

\showglotype \showglotype{\label{}}

```

7751 \newcommand*{\showglotype}[1]{%
7752   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7753 }

```

\showglocounter \showglocounter{\label{}}

```

7754 \newcommand*{\showglocounter}[1]{%
7755   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7756 }

```

\showglouserii \showglouserii{\label{}}

```

7757 \newcommand*{\showglouserii}[1]{%
7758   \expandafter\show\csname glo@glstetoklabel{#1}@userii\endcsname
7759 }

```

\showglouseriii \showglouseriii{\label{}}

```

7760 \newcommand*{\showglouseriii}[1]{%
7761   \expandafter\show\csname glo@glstetoklabel{#1}@useriii\endcsname
7762 }

```

\showglouseriiii \showglouseriiii{\label{}}

```

7763 \newcommand*{\showglouseriii}[1]{%
7764   \expandafter\show\csname glo\glsdetoklabel{#1}@useriii\endcsname
7765 }

```

\showglouseriv \showglouseriv{<label>}

```

7766 \newcommand*{\showglouseriv}[1]{%
7767   \expandafter\show\csname glo\glsdetoklabel{#1}@useriv\endcsname
7768 }

```

\showglouserv \showglouserv{<label>}

```

7769 \newcommand*{\showglouserv}[1]{%
7770   \expandafter\show\csname glo\glsdetoklabel{#1}@userv\endcsname
7771 }

```

\showglouservi \showglouservi{<label>}

```

7772 \newcommand*{\showglouservi}[1]{%
7773   \expandafter\show\csname glo\glsdetoklabel{#1}@uservi\endcsname
7774 }

```

\showgloname \showgloname{<label>}

```

7775 \newcommand*{\showgloname}[1]{%
7776   \expandafter\show\csname glo\glsdetoklabel{#1}@name\endcsname
7777 }

```

\showglodesc \showglodesc{<label>}

```

7778 \newcommand*{\showglodesc}[1]{%
7779   \expandafter\show\csname glo\glsdetoklabel{#1}@desc\endcsname
7780 }

```

howglodescplural \showglodescplural{<label>}

```

7781 \newcommand*{\showglodescplural}[1]{%
7782   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7783 }

```

\showglosort    \showglosort{<label>}

```

7784 \newcommand*{\showglosort}[1]{%
7785   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7786 }

```

\showglosymbol    \showglosymbol{<label>}

```

7787 \newcommand*{\showglosymbol}[1]{%
7788   \expandafter\show\csname glo@glstdetoklabel{#1}@symbol\endcsname
7789 }

```

wglosymbolplural    \showglosymbolplural{<label>}

```

7790 \newcommand*{\showglosymbolplural}[1]{%
7791   \expandafter\show\csname glo@glstdetoklabel{#1}@symbolplural\endcsname
7792 }

```

\showgloshort    \showgloshort{<label>}

```

7793 \newcommand*{\showgloshort}[1]{%
7794   \expandafter\show\csname glo@glstdetoklabel{#1}@short\endcsname
7795 }

```

\showglolong    \showglolong{<label>}

```

7796 \newcommand*{\showglolong}[1]{%
7797   \expandafter\show\csname glo@glstdetoklabel{#1}@long\endcsname
7798 }

```

\showgloindex    \showgloindex{<label>}

```

7799 \newcommand*{\showgloindex}[1]{%
7800   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7801 }

```

\showgloflag    \showgloflag{<label>}

```

7802 \newcommand*{\showgloflag}[1]{%
7803   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7804 }

```

\showgloloclist    \showgloloclist{<label>}

```

7805 \newcommand*{\showgloloclist}[1]{%
7806   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7807 }

```

\showglofield    \showglofield{<label>}{<field>}

```

7808 \newcommand*{\showglofield}[2]{%
7809   \cshow{glo@\glsdetoklabel{#1}@#2}%
7810 }

```

showacronymlists    \showacronymlists

Show list of glossaries that have been flagged as a list of acronyms.

```

7811 \newcommand*{\showacronymlists}{%
7812   \show\@glsacronymlists
7813 }

```

\showglossaries    \showglossaries

Show list of defined glossaries.

```

7814 \newcommand*{\showglossaries}{%
7815   \show\@glo@types
7816 }

```

\showglossaryin    \showglossaryin{<glossary-label>}

Show the ‘in’ extension for the given glossary.

```
7817 \newcommand*{\showglossaryin}[1]{%  
7818   \expandafter\show\csname @glotype@#1@in\endcsname  
7819 }
```

\showglossaryout    \showglossaryout{<glossary-label>}

Show the ‘out’ extension for the given glossary.

```
7820 \newcommand*{\showglossaryout}[1]{%  
7821   \expandafter\show\csname @glotype@#1@out\endcsname  
7822 }
```

showglossarytitle    \showglossarytitle{<glossary-label>}

Show the title for the given glossary.

```
7823 \newcommand*{\showglossarytitle}[1]{%  
7824   \expandafter\show\csname @glotype@#1@title\endcsname  
7825 }
```

wglossarycounter    \showglossarycounter{<glossary-label>}

Show the counter for the given glossary.

```
7826 \newcommand*{\showglossarycounter}[1]{%  
7827   \expandafter\show\csname @glotype@#1@counter\endcsname  
7828 }
```

wglossaryentries    \showglossaryentries{<glossary-label>}

Show the list of entry labels for the given glossary.

```
7829 \newcommand*{\showglossaryentries}[1]{%  
7830   \expandafter\show\csname glolist@#1\endcsname  
7831 }
```

## 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.

```

7832 \csname ifglscpatible-2.07\endcsname
7833   \RequirePackage{glossaries-compatible-207}
7834 \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.

```
7835 \NeedsTeXFormat{LaTeX2e}
```

```
7836 \ProvidesPackage{glossaries-prefix}[2017/08/10 v4.31 (NLCT)]
```

Pass all options to glossaries:

```
7837 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7838 \ProcessOptions
```

Load glossaries:

```
7839 \RequirePackage{glossaries}
```

Add the new keys:

```
7840 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7841 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7842 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7843 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7844 \appto\@gls@keymap{,%
```

```
7845   {prefixfirst}{prefixfirst},%
```

```
7846   {prefixfirstplural}{prefixfirstplural},%
```

```
7847   {prefix}{prefix},%
```

```
7848   {prefixplural}{prefixplural}}%
```

```
7849 }
```

Set the default values:

```
7850 \appto\@newglossaryentryprehook{%
```

```
7851   \def\@glo@entryprefix{}}%
```

```
7852   \def\@glo@entryprefixplural{}}%
```

```
7853   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7854   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7855 }
```

Set the assignment code:

```
7856 \appto\@newglossaryentryposthook{%
```

```
7857   \gls@assign@field{}{\@glo@label}{prefix}{\@glo@entryprefix}}%
```

```
7858   \gls@assign@field{}{\@glo@label}{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7859 \expandafter\gls@assign@field\expandafter
```

```
7860   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
7861   {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7862 \expandafter\gls@assign@field\expandafter
7863   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7864   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7865 }

```

Define commands to access these fields:

entryprefixfirst

```

7866 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7867 \newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7868 \newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7869 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7870 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
7871   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7872   \xmakefirstuc\@glo@text
7873 }

```

entryfirstplural

```

7874 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7875   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7876   \xmakefirstuc\@glo@text
7877 }

```

\Glsentryprefix

```

7878 \newrobustcmd*{\Glsentryprefix}[1]{%
7879   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7880   \xmakefirstuc\@glo@text
7881 }

```

entryprefixplural

```

7882 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7883   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7884   \xmakefirstuc\@glo@text
7885 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7886 \newcommand*{\ifglshasprefix}[3]{%
7887   \ifcempty{glo@#1@prefix}%
7888   {#3}%
7889   {#2}%
7890 }
```

hasprefixplural

```
7891 \newcommand*{\ifglshasprefixplural}[3]{%
7892   \ifcempty{glo@#1@prefixplural}%
7893   {#3}%
7894   {#2}%
7895 }
```

shasprefixfirst

```
7896 \newcommand*{\ifglshasprefixfirst}[3]{%
7897   \ifcempty{glo@#1@prefixfirst}%
7898   {#3}%
7899   {#2}%
7900 }
```

efixfirstplural

```
7901 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7902   \ifcempty{glo@#1@prefixfirstplural}%
7903   {#3}%
7904   {#2}%
7905 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7906 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7907 \newcommand*{\@pgls}[2][ ]{%
7908   \new@ifnextchar[%
7909   {\@pgls@{#1}{#2}}%
7910   {\@pgls@{#1}{#2}[ ]}%
7911 }
```

\@pgls@ Read in the final optional argument:

```
7912 \def\@pgls@#1#2[#3]{%
7913   \glsdoifexists{#2}%
7914   {%
7915     \ifglsused{#2}%
7916     {%
7917       \glstryprefix{#2}%
7918     }%

```

```

7919     {%
7920     \glstryentryprefixfirst{#2}%
7921     }%
7922     \@gls@{#1}{#2}[#3]%
7923     }%
7924 }

```

Similarly for the plural version:

```

\pglsp1
7925 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7926 \newcommand*{\@pglsp1}[2][ ]{%
7927   \new@ifnextchar[%
7928     {\@pglsp1@{#1}{#2}}%
7929     {\@pglsp1@{#1}{#2}[ ]}%
7930 }

```

\@pglsp1@ Read in the final optional argument:

```

7931 \def\@pglsp1@#1#2[#3]{%
7932   \glsdoifexists{#2}%
7933   {%
7934     \ifglsused{#2}%
7935     {%
7936       \glstryentryprefixplural{#2}%
7937     }%
7938     {%
7939       \glstryentryprefixfirstplural{#2}%
7940     }%
7941     \@glspl@{#1}{#2}[#3]%
7942   }%
7943 }

```

Now for the first letter upper case versions:

```

\Pgls
7944 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7945 \newcommand*{\@Pgls}[2][ ]{%
7946   \new@ifnextchar[%
7947     {\@Pgls@{#1}{#2}}%
7948     {\@Pgls@{#1}{#2}[ ]}%
7949 }

```

\@Pgls@ Read in the final optional argument:

```

7950 \def\@Pgls@#1#2[#3]{%

```

```

7951 \glsdoifexists{#2}%
7952 {%
7953   \ifglsused{#2}%
7954   {%
7955     \ifglshasprefix{#2}%
7956     {%
7957       \Glsentryprefix{#2}%
7958       \@gls@{#1}{#2}[#3]%
7959     }%
7960     {\@Gls@{#1}{#2}[#3]}%
7961   }%
7962   {%
7963     \ifglshasprefixfirst{#2}%
7964     {%
7965       \Glsentryprefixfirst{#2}%
7966       \@gls@{#1}{#2}[#3]%
7967     }%
7968     {\@Gls@{#1}{#2}[#3]}%
7969   }%
7970 }%
7971 }

```

Similarly for the plural version:

```

\Pglspl
7972 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

7973 \newcommand*{\@Pglspl}[2] [] {%
7974   \new@ifnextchar[%
7975   {\@Pglspl@{#1}{#2}}%
7976   {\@Pglspl@{#1}{#2} []}%
7977 }

```

\@Pglspl@ Read in the final optional argument:

```

7978 \def\@Pglspl@#1#2[#3] {%
7979   \glsdoifexists{#2}%
7980   {%
7981     \ifglsused{#2}%
7982     {%
7983       \ifglshasprefixplural{#2}%
7984       {%
7985         \Glsentryprefixplural{#2}%
7986         \@glspl@{#1}{#2}[#3]%
7987       }%
7988       {\@Glspl@{#1}{#2}[#3]}%
7989     }%
7990     {%
7991       \ifglshasprefixfirstplural{#2}%

```

```

7992      {%
7993      \Glsentryprefixfirstplural{#2}%
7994      \@glspl@{#1}{#2}[#3]%
7995      }%
7996      {\@Glspl@{#1}{#2}[#3]}%
7997      }%
7998  }%
7999 }

```

Finally the all upper case versions:

```

\PGLS
8000 \newrobustcmd{\PGLS}{\@gls@hyp@opt\@PGLS}

```

\@PGLS Unstarred version.

```

8001 \newcommand*{\@PGLS}[2][ ]{%
8002   \new@ifnextchar[%
8003   {\@PGLS@{#1}{#2}}%
8004   {\@PGLS@{#1}{#2}[ ]}%
8005 }

```

\@PGLS@ Read in the final optional argument:

```

8006 \def\@PGLS@#1#2[#3]{%
8007   \glsdoifexists{#2}%
8008   {%
8009     \ifglsused{#2}%
8010     {%
8011       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
8012     }%
8013     {%
8014       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
8015     }%
8016     \@GLS@{#1}{#2}[#3]%
8017   }%
8018 }

```

Plural version:

```

\PGLSpl
8019 \newrobustcmd{\PGLSpl}{\@gls@hyp@opt\@PGLSpl}

```

\@PGLSpl Unstarred version.

```

8020 \newcommand*{\@PGLSpl}[2][ ]{%
8021   \new@ifnextchar[%
8022   {\@PGLSpl@{#1}{#2}}%
8023   {\@PGLSpl@{#1}{#2}[ ]}%
8024 }

```

\@PGLSp1@ Read in the final optional argument:

```
8025 \def\@PGLSp1@#1#2[#3]{%
8026   \glsdoifexists{#2}%
8027   {%
8028     \ifglsused{#2}%
8029     {%
8030       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
8031     }%
8032     {%
8033       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
8034     }%
8035     \@GLSp1@{#1}{#2}[#3]%
8036   }%
8037 }
```

## 3 Glossary Styles

### 3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8038 \ProvidesPackage{glossary-hypernav}[2017/08/24 v4.32 (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`

```
8039 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8040   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8041   \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

`navhyperlinkname`

Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
8042 \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.

`snahypertarget`

```
8043 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
8044   \@glsnavhypertarget{#1}{#2}{#3}%
8045 }
```

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.

`snahypertarget`

```
8046 \newcommand*{\@glsnavhypertarget}[3]{%}
```



Add this group to the aux file for re-run check.

```
8047 \protected@write\auxout{}\string\@gls@hypergroup{#1}{#2}}%
```

Add the target.

```
8048 \@glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8049 \expandafter\let
```

```
8050 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8051 \@for\@gls@elem:=\@gls@list\do{%
```

```
8052 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
8053 \if@endfor
```

```
8054 \else
```

This group was not included in the list, so issue a warning.

```
8055 \GlossariesWarningNoLine{Navigation panel
```

```
8056 for glossary type ‘#1’~Jmissing group ‘#2’}%
```

```
8057 \gdef\gls@hypergroup@rerun{%
```

```
8058 \GlossariesWarningNoLine{Navigation panel
```

```
8059 has changed. Rerun LaTeX}}%
```

```
8060 \fi
```

```
8061 }
```

`\hypergroup@rerun` Give a warning at the end if re-run required

```
8062 \let\gls@hypergroup@rerun\relax
```

```
8063 \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.

```
8064 \newcommand*{\@gls@hypergroup}[2]{%
```

```
8065 \@ifundefined{\@gls@hypergroup@list@#1}{%
```

```
8066 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}}%
```

```
8067 }{%
```

```
8068 \expandafter\let\expandafter\@gls@tmp
```

```
8069 \csname @gls@hypergroup@list@#1\endcsname
```

```
8070 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
```

```
8071 \@gls@tmp,#2}}%
```

```
8072 }%
```

```
8073 }
```

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`

```
8074 \newcommand*{\glsnavigation}{%
8075   \def\@gls@between{}%
8076   \ifcsundef{\@gls@hypergroupplist@\@glo@type}%
8077   {%
8078     \def\@gls@list{}%
8079   }%
8080   {%
8081     \expandafter\let\expandafter\@gls@list
8082       \csname \@gls@hypergroupplist@\@glo@type\endcsname
8083   }%
8084   \@for\@gls@tmp:=\@gls@list\do{%
8085     \@gls@between

8086     \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8087     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8088     \let\@gls@between\glshypernavsep
8089   }%
8090 }
```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8091 \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`

```
8092 \newcommand*{\glssymbolnav}{%
8093   \glsnavhyperlink{glssymbols}{\@gls@getgrouptitle{glssymbols}}%
8094   \glshypernavsep
8095   \glsnavhyperlink{glsnumbers}{\@gls@getgrouptitle{glsnumbers}}%
8096   \glshypernavsep
8097 }
```

## 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.

```
8098 \ProvidesPackage{glossary-inline}[2017/08/24 v4.32 (NLCT)]
```

`inline` Define the inline style.

```
8099 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then changed by
    \glossentry)

8100   \renewenvironment{theglossary}%
8101   {%
```

```

8102     \def\gls@inlinesep{}%
8103     \def\gls@inlinesubsep{}%
8104     \def\gls@inlinepostchild{}%
8105     }%
8106     {\glspostinline}%

```

No header:

```

8107 \renewcommand*{\glossaryheader}{}%

```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```

8108 \renewcommand*{\glsgroupheading}[1]{}%

```

Just display separator followed by name and description:

```

8109 \renewcommand{\glossentry}[2]{%
8110     \glsinlinedopostchild
8111     \gls@inlinesep
8112     \glsentryitem{##1}%
8113     \glsinlinenameformat{##1}%
8114     \glossentryname{##1}%
8115     }%
8116     \ifglstdescsuppressed{##1}%
8117     {%
8118         \glsinlineemptydescformat
8119         {%
8120             \glossentrysymbol{##1}%
8121             }%
8122             {%
8123                 ##2%
8124             }%
8125         }%
8126         {%
8127             \ifglshasdesc{##1}%
8128             {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8129             {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8130         }%
8131         \ifglshaschildren{##1}%
8132         {%
8133             \glsresetsubentrycounter
8134             \glsinlineparentchildseparator
8135             \def\gls@inlinesubsep{}%
8136             \def\gls@inlinepostchild{\glsinlinepostchild}%
8137         }%
8138         {}%
8139     \def\gls@inlinesep{\glsinlineseparator}%
8140 }%

```

Sub-entries display description:

```

8141 \renewcommand{\subglossentry}[3]{%
8142     \gls@inlinesubsep%
8143     \glsinlinesubnameformat{##2}%

```

```

8144     \glossentryname{##2}}}%
8145     \glssubentryitem{##2}%
8146     \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8147     \def\gls@inlinesubsep{\glsinlinesubseparator}%
8148 }%

```

Nothing special between groups:

```

8149 \renewcommand*{\glsgroupskip}{}%
8150 }

```

linedopostchild

```

8151 \newcommand*{\glsinlinedopostchild}{%
8152     \gls@inlinepostchild
8153     \def\gls@inlinepostchild{}%
8154 }

```

inlineseparator Separator to use between entries.

```

8155 \newcommand*{\glsinlineseparator}{;\space}

```

inlinesubseparator Separator to use between sub-entries.

```

8156 \newcommand*{\glsinlinesubseparator}{,\space}

```

parentchildseparator Separator to use between parent and children.

```

8157 \newcommand*{\glsinlineparentchildseparator}{:\space}

```

inlinepostchild Hook to use between child and next entry

```

8158 \newcommand*{\glsinlinepostchild}{}

```

\glspostinline Terminator for inline glossary.

```

8159 \newcommand*{\glspostinline}{\glspostdescription\space}

```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```

8160 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}

```

inlinedescformat Formats the entry's description, symbol and location list:

```

8161 \newcommand*{\glsinlinedescformat}[3]{\space#1}

```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```

8162 \newcommand*{\glsinlineemptydescformat}[2]{}

```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```

8163 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}

```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```

8164 \newcommand*{\glsinlinesubdescformat}[3]{#1}

```

### 3.3 List Style (glossary-list.sty)

The style file defines glossary styles that use the description environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

8165 \ProvidesPackage{glossary-list}[2017/08/24 v4.32 (NLCT)]

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8166 \providecommand{\indexspace}{%
8167   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8168 }
```

`tgrouphaderfmt` Provide a way of adjusting the format of the group headings.

```
8169 \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.

```
8170 \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.

```
8171 \newglossarystyle{list}{%
```

Use description environment:

```
8172   \renewenvironment{theglossary}%
8173     {\begin{description}}{\end{description}}%
```

No header at the start of the environment:

```
8174   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8175   \renewcommand*{\glsgrpskip}[1]{}%
```

Main (level 0) entries start a new item in the list:

```
8176   \renewcommand*{\glossentry}[2]{%
8177     \item[\glsentryitem{##1}%
8178       \glstarget{##1}{\glossentryname{##1}}]
8179     \glossentrydesc{##1}\glspostdescription\space ##2}%
```

Sub-entries continue on the same line:

```
8180   \renewcommand*{\subglossentry}[3]{%
8181     \glssubentryitem{##2}%
```

```

8182     \glstarget{##2}{\strut}\space
8183     \glossentrydesc{##2}\glspostdescription\space ##3.}%

Add vertical space between groups:

8184     \renewcommand*{\glsgroupskip}{\ifglsgnoglobskip\else\indexspace\fi}%
8185 }

```

**listgroup** The listgroup style is like the list style, but the glossary groups have headings.

```

8186 \newglossarystyle{listgroup}{%

Base it on the list style:

8187     \setglossarystyle{list}%

Each group has a heading:

8188     \renewcommand*{\glsgroupheading}[1]{%
8189         \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.

```

8190 \newglossarystyle{listhypergroup}{%

Base it on the list style:

8191     \setglossarystyle{list}%

Add navigation links at the start of the environment.

8192     \renewcommand*{\glossaryheader}{%
8193         \glslistnavigationitem{\glsgroupnavigation}}%

Each group has a heading with a hypertarget:

8194     \renewcommand*{\glsgroupheading}[1]{%
8195         \item[\glslistgroupheaderfmt
8196             {\glsgrouphypertarget{##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.

```

8197 \newglossarystyle{altlist}{%

Base it on the list style:

8198     \setglossarystyle{list}%

Main (level 0) entries start a new item in the list with a line break after the entry name:

8199     \renewcommand*{\glossentry}[2]{%
8200         \item[\glsgroupitem{##1}%
8201             \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.

```

8202         \mbox{} \par \nobreak \@afterheading
8203         \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries start a new paragraph:

```
8204 \renewcommand{\subglossentry}[3]{%
8205   \par
8206   \glssubentryitem{##2}%
8207   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%
8208 }
```

**altlistgroup** The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8209 \newglossarystyle{altlistgroup}{%
      Base it on the altlist style:
8210   \setglossarystyle{altlist}%
      Each group has a heading:
8211   \renewcommand*{\glsgroupheading}[1]{%
8212     \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.

```
8213 \newglossarystyle{altlisthypergroup}{%
      Base it on the altlist style:
8214   \setglossarystyle{altlist}%
      Add navigation links at the start of the environment.
8215   \renewcommand*{\glossaryheader}{%
8216     \glslistnavigationitem{\glslnavigation}}%
      Each group has a heading with a hypertarget:
8217   \renewcommand*{\glsgroupheading}[1]{%
8218     \item[\glslistgroupheaderfmt
8219       {\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.

```
8220 \newglossarystyle{listdotted}{%
      Base it on the list style:
8221   \setglossarystyle{list}%
      Each main (level 0) entry starts a new item:
8222   \renewcommand*{\glossentry}[2]{%
8223     \item[]\makebox[\glslistdottedwidth][l]{%
8224       \glssentryitem{##1}%
8225       \glstarget{##1}{\glossentryname{##1}}%
8226       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
```

Sub entries have the same format as main entries:

```

8227 \renewcommand*{\subglossentry}[3]{%
8228   \item[\makebox[\glslistdottedwidth][l]{%
8229     \glssubentryitem{##2}}%
8230   \glstarget{##2}{\glossentryname{##2}}%
8231   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8232 }
```

`listdottedwidth`

```

8233 \newlength\glslistdottedwidth
8234 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8235 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
8236 \setglossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```

8237 \renewcommand*{\glossentry}[2]{%
8238   \item[\glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}}}%
8239 }
```

### 3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8240 \ProvidesPackage{glossary-long}[2017/08/24 v4.32 (NLCT)]
```

Requires the package:

```
8241 \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`.)

```

8242 \@ifundefined{glsdescwidth}{%
8243   \newlength\glsdescwidth
8244   \setlength{\glsdescwidth}{0.6\hsize}
8245 }{}
```

`lspagelistwidth` This is a length that governs the width of the page list column.

```

8246 \@ifundefined{glspagelistwidth}{%
8247   \newlength\glspagelistwidth
8248   \setlength{\glspagelistwidth}{0.1\hsize}
8249 }{}
```



**long** The long glossary style command which uses the longtable environment:

```
8250 \newglossarystyle{long}{%
```

Use longtable with two columns:

```
8251 \renewenvironment{theglossary}{%
8252     {\begin{longtable}\lp{\glstdescwidth}}}%
8253     {\end{longtable}}}%
```

Do nothing at the start of the environment:

```
8254 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8255 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8256 \renewcommand{\glossentry}[2]{%
8257     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8258     \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8259 }%
```

Sub entries displayed on the following row without the name:

```
8260 \renewcommand{\subglossentry}[3]{%
8261     &
8262     \glssubentryitem{##2}%
8263     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8264     ##3\tabularnewline
8265 }%
```

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>)

```
8266 \ifglsnogroupskip
8267     \renewcommand*{\glsgroupskip}{}%
8268 \else
8269     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8270 \fi
8271 }
```

**longborder** The longborder style is like the above, but with horizontal and vertical lines:

```
8272 \newglossarystyle{longborder}{%
```

Base it on the glostylelong style:

```
8273 \setglossarystyle{long}%
```

Use longtable with two columns with vertical lines between each column:

```
8274 \renewenvironment{theglossary}{%
8275     \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8276 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8277 }
```

**longheader** The longheader style is like the long style but with a header:

```
8278 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8279 \setglossarystyle{long}{%
```

Set the table's header:

```
8280 \renewcommand*{\glossaryheader}{%
8281 \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8282 }
```

`longheaderborder` The `longheaderborder` style is like the `long` style but with a header and border:

```
8283 \newglossarystyle{longheaderborder}{%
```

Base it on the `glostylelongborder` style:

```
8284 \setglossarystyle{longborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8285 \renewcommand*{\glossaryheader}{%
8286 \hline\bfseries \entryname & \bfseries
8287 \descriptionname\tabularnewline\hline
8288 \endhead
8289 \hline\endfoot}%
8290 }
```

`long3col` The `long3col` style is like `long` but with 3 columns

```
8291 \newglossarystyle{long3col}{%
```

Use a `longtable` with 3 columns:

```
8292 \renewenvironment{theglossary}{%
8293 {\begin{longtable}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8294 {\end{longtable}}}%

```

No table header:

```
8295 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```
8296 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8297 \renewcommand{\glossentry}[2]{%
8298 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8299 \glossentrydesc{##1} & ##2\tabularnewline
8300 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8301 \renewcommand{\subglossentry}[3]{%
8302 &
8303 \glssubentryitem{##2}%
8304 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8305 ##3\tabularnewline
8306 }%

```

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>)

```
8307 \ifglsgroupskip
8308 \renewcommand*{\glsgroupskip}{}%
8309 \else
8310 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8311 \fi
8312 }
```

**long3colborder** The long3colborder style is like the long3col style but with a border:

```
8313 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
8314 \setglossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
8315 \renewenvironment{theglossary}%
8316 {\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8317 {\end{longtable}}%
    Place horizontal lines at the head and foot of the table:
8318 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8319 }
```

**long3colheader** The long3colheader style is like long3col but with a header row:

```
8320 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
8321 \setglossarystyle{long3col}%
    Set the table's header:
8322 \renewcommand*{\glossaryheader}{%
8323 \bfseries\entryname&\bfseries\descriptionname&
8324 \bfseries\pagelistname\tabularnewline\endhead}%
8325 }
```

**colheaderborder** The long3colheaderborder style is like the above but with a border

```
8326 \newglossarystyle{long3colheaderborder}{%
    Base it on the glostylelong3colborder style:
8327 \setglossarystyle{long3colborder}%
    Set the table's header and add horizontal line at table's foot:
8328 \renewcommand*{\glossaryheader}{%
8329 \hline
8330 \bfseries\entryname&\bfseries\descriptionname&
8331 \bfseries\pagelistname\tabularnewline\hline\endhead
8332 \hline\endfoot}%
8333 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8334 \newglossarystyle{long4col}{%
```

Use a `longtable` with 4 columns:

```
8335 \renewenvironment{theglossary}{%
```

```
8336 {\begin{longtable}{llll}}%
```

```
8337 {\end{longtable}}%
```

No table header:

```
8338 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8339 \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):

```
8340 \renewcommand{\glossentry}[2]{%
```

```
8341 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
8342 \glossentrydesc{##1} &
```

```
8343 \glossentrysymbol{##1} &
```

```
8344 ##2\tabularnewline
```

```
8345 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8346 \renewcommand{\subglossentry}[3]{%
```

```
8347 &
```

```
8348 \glssubentryitem{##2}%
```

```
8349 \glstarget{##2}{\strut}\glossentrydesc{##2} &
```

```
8350 \glossentrysymbol{##2} & ##3\tabularnewline
```

```
8351 }%
```

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>)

```
8352 \ifglsgnogroupskip
```

```
8353 \renewcommand*{\glsgroupskip}{}%
```

```
8354 \else
```

```
8355 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
```

```
8356 \fi
```

```
8357 }
```

`long4colheader` The `long4colheader` style is like `long4col` but with a header row.

```
8358 \newglossarystyle{long4colheader}{%
```

Base it on the `glostylelong4col` style:

```
8359 \setglossarystyle{long4col}%
```

Table has a header:

```
8360 \renewcommand*{\glossaryheader}{%
```

```
8361 \bfseries\entryname&\bfseries\descriptionname&
```

```
8362 \bfseries \symbolname&
```

```

8363     \bfseries\pagelistname\tabularnewline\endhead}%
8364 }

```

**long4colborder** The long4colborder style is like long4col but with a border.

```

8365 \newglossarystyle{long4colborder}{%
    Base it on the glostylelong4col style:
8366     \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8367     \renewenvironment{theglossary}%
8368         {\begin{longtable}{|l|l|l|l|}}%
8369         {\end{longtable}}%
    Add horizontal lines to the head and foot of the table:
8370     \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8371 }

```

**colheaderborder** The long4colheaderborder style is like the above but with a border.

```

8372 \newglossarystyle{long4colheaderborder}{%
    Base it on the glostylelong4col style:
8373     \setglossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
8374     \renewenvironment{theglossary}%
8375         {\begin{longtable}{|l|l|l|l|}}%
8376         {\end{longtable}}%
    Add table header and horizontal line at the table's foot:
8377     \renewcommand*{\glossaryheader}{%
8378         \hline\bfseries\entryname&\bfseries\descriptionname&
8379         \bfseries \symbolname&
8380         \bfseries\pagelistname\tabularnewline\hline\endhead
8381         \hline\endfoot}%
8382 }

```

**altlong4col** The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8383 \newglossarystyle{altlong4col}{%
    Base it on the glostylelong4col style:
8384     \setglossarystyle{long4col}%
    Use a longtable with 4 columns where the second and last columns may have multiple lines
    in each row:
8385     \renewenvironment{theglossary}%
8386         {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8387         {\end{longtable}}%
8388 }

```

`altlong4colheader` The `altlong4colheader` style is like `altlong4col` but with a header row.

```
8389 \newglossarystyle{altlong4colheader}{%
```

Base it on the `glostylelong4colheader` style:

```
8390 \setglossarystyle{long4colheader}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8391 \renewenvironment{theglossary}{%
```

```
8392 {\begin{longtable}{\lp{\glsgdescwidth}\lp{\glspagelistwidth}}}%
```

```
8393 {\end{longtable}}}%
```

```
8394 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
8395 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
8396 \setglossarystyle{long4colborder}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8397 \renewenvironment{theglossary}{%
```

```
8398 {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
```

```
8399 {\end{longtable}}}%
```

```
8400 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
8401 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
8402 \setglossarystyle{long4colheaderborder}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8403 \renewenvironment{theglossary}{%
```

```
8404 {\begin{longtable}{\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
```

```
8405 {\end{longtable}}}%
```

```
8406 }
```

### 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>

```
8407 \ProvidesPackage{glossary-longbooktabs}[2017/08/24 v4.32 (NLCT)]
```

Requires `booktabs` package:

```
8408 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8409 \RequirePackage{glossary-long}
8410 \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.

```
8411 \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.

```
8412 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8413 \setglossarystyle{long}%
```

Add a header with rules.

```
8414 \renewcommand*{\glossaryheader}{%
8415 \toprule \bfseries \entryname & \bfseries
8416 \descriptionname\tabularnewline\midrule\endhead
8417 \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.

```
8418 \ifglsgroupskip
8419 \renewcommand*{\glsgroupskip}{}%
8420 \else
8421 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8422 \fi
8423 }
```

**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.

```
8424 \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.

```
8425 \glspatchLToutput
```

Use the long3col style as a base.

```
8426 \setglossarystyle{long3col}%
```

Add a header with rules.

```
8427 \renewcommand*{\glossaryheader}{%
8428 \toprule \bfseries \entryname &
8429 \bfseries \descriptionname &
8430 \bfseries \pagelistname
8431 \tabularnewline\midrule\endhead
8432 \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.

```
8433 \ifglsnogroupskip
8434 \renewcommand*{\glsgroupskip}{}%
8435 \else
8436 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8437 \fi
8438 }
```

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.

```
8439 \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.

```
8440 \glspatchLToutput
```

Use the long4col style as a base.

```
8441 \setglossarystyle{long4col}{%
```

Add a header with rules.

```
8442 \renewcommand*{\glossaryheader}{%
8443 \toprule \bfseries \entryname &
8444 \bfseries \descriptionname &
8445 \bfseries \symbolname &
8446 \bfseries \pagelistname
8447 \tabularnewline\midrule\endhead
8448 \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.

```
8449 \ifglsnogroupskip
8450 \renewcommand*{\glsgroupskip}{}%
8451 \else
8452 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8453 \fi
8454 }
```

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.

```
8455 \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.

```
8456 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8457 \setglossarystyle{long4col-booktabs}{%
```



Change the column specifications:

```
8458 \renewenvironment{theglossary}%  
8459   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%  
8460   {\end{longtable}}}%  
8461 }
```

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.

```
8462 \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.

```
8463 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8464 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8465 \renewenvironment{theglossary}%  
8466   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%  
8467   {\end{longtable}}}%  
8468 }
```

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.

```
8469 \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.

```
8470 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8471 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8472 \renewenvironment{theglossary}%  
8473   {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%  
8474     >{\raggedright}p{\glspagelistwidth}}}%  
8475   {\end{longtable}}}%  
8476 }
```

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.

```
8477 \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.

```
8478 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8479 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8480 \renewenvironment{theglossary}%  
8481 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8482 >{\raggedright}p{\glspagelistwidth}}}%  
8483 {\end{longtable}}%  
8484 }
```

sLTpenaltycheck

```
8485 \newcommand*{\glslTpenaltycheck}{%  
8486 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8487 }
```

enaltygroupskip

```
8488 \newcommand{\glspenaltygroupskip}{%  
8489 \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8490 \let\@gls@org@LT@output\LT@output  
8491 \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

```
8492 \newcommand*{\glspatchLToutput}{%  
8493 \renewcommand*{\LT@output}{%  
8494 \ifnum\outputpenalty < -\@Mi  
8495 \ifnum\outputpenalty > -\LT@end@pen  
8496 \LT@err{floats and marginpars not allowed in a longtable}\@ehc  
8497 \else  
8498 \setbox\z@\vbox{\unvbox\@cclv}%  
8499 \ifdim \ht\LT@lastfoot>\ht\LT@foot  
8500 \dimen@pagegoal  
8501 \advance\dimen@-\ht\LT@lastfoot  
8502 \ifdim\dimen@<\ht\z@  
8503 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%  
8504 \@makecol  
8505 \@outputpage  
8506 \setbox\z@\vbox{\box\LT@head\glslTpenaltycheck}%  
8507 \fi  
8508 \fi  
8509 \global\@colroom\@colht  
8510 \global\vsize\@colht  
8511 {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%  
8512 \fi  
8513 \else
```

```

8514 \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8515 \@makecol
8516 \@outputpage
8517 \global\ysize\@colroom
8518 \copy\LT@head
8519 \glsLTpenaltycheck
8520 \nobreak
8521 \fi
8522 }%
8523 }

```

### 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.

```
8524 \ProvidesPackage{glossary-longragged}[2017/08/24 v4.32 (NLCT)]
```

Requires the package:

```
8525 \RequirePackage{array}
```

Requires the package:

```
8526 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

8527 \@ifundefined{glsdescwidth}{%
8528 \newlength\glsdescwidth
8529 \setlength{\glsdescwidth}{0.6\hsize}
8530 }{}

```

`lspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8531 \@ifundefined{glspagelistwidth}{%
8532 \newlength\glspagelistwidth
8533 \setlength{\glspagelistwidth}{0.1\hsize}
8534 }{}

```

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

```
8535 \newglossarystyle{longragged}{%
```

Use longtable with two columns:

```

8536 \renewenvironment{theglossary}%
8537 {\begin{longtable}{l>\raggedright}p{\glsdescwidth}}%
8538 {\end{longtable}}%

```

Do nothing at the start of the environment:

```
8539 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8540 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8541 \renewcommand{\glossentry}[2]{%
8542   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8543   \glossentrydesc{##1}\glspostdescription\space ##2%
8544   \tabularnewline
8545 }%
```

Sub entries displayed on the following row without the name:

```
8546 \renewcommand{\subglossentry}[3]{%
8547   &
8548   \glssubentryitem{##2}%
8549   \glstarget{##2}{\strut}\glossentrydesc{##2}%
8550   \glspostdescription\space ##3%
8551   \tabularnewline
8552 }%
```

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>)

```
8553 \ifglsgroupskip
8554 \renewcommand*{\glsgroupskip}{}%
8555 \else
8556 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8557 \fi
8558 }
```

`ongraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8559 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8560 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8561 \renewenvironment{theglossary}{%
8562   \begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|}%
8563   {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8564 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8565 }
```

`ongraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8566 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8567 \setglossarystyle{longragged}%
```

Set the table's header:

```
8568 \renewcommand*{\glossaryheader}{%
8569   \bfseries \entryname & \bfseries \descriptionname
```

```

8570 \tabularnewline\endhead}%
8571 }

```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```

8572 \newglossarystyle{longraggedheaderborder}{%

```

Base it on the `glostylelongraggedborder` style:

```

8573 \setglossarystyle{longraggedborder}%

```

Set the table's header and add horizontal line to table's foot:

```

8574 \renewcommand*{\glossaryheader}{%
8575 \hline\bfseries \entryname & \bfseries \descriptionname
8576 \tabularnewline\hline
8577 \endhead
8578 \hline\endfoot}%
8579 }

```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```

8580 \newglossarystyle{longragged3col}{%

```

Use a `longtable` with 3 columns:

```

8581 \renewenvironment{theglossary}%
8582 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8583 >{\raggedright}p{\glspagelistwidth}}}%
8584 {\end{longtable}}%

```

No table header:

```

8585 \renewcommand*{\glossaryheader}{}%

```

No headings between groups:

```

8586 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8587 \renewcommand{\glossentry}[2]{%
8588 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8589 \glossentrydesc{##1} & ##2\tabularnewline
8590 }%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```

8591 \renewcommand{\subglossentry}[3]{%
8592 &
8593 \glssubentryitem{##2}%
8594 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8595 ##3\tabularnewline
8596 }%

```

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>)

```

8597 \ifglsnogroupskip
8598 \renewcommand*{\glsgroupskip}{}%

```

```

8599 \else
8600   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8601 \fi
8602 }

```

**ragged3colborder** The longragged3colborder style is like the longragged3col style but with a border:

```

8603 \newglossarystyle{longragged3colborder}{%
    Base it on the glostylelongragged3col style:
8604   \setglossarystyle{longragged3col}%
    Use a longtable with 3 columns with vertical lines around them:
8605   \renewenvironment{theglossary}%
8606     {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%
8607       >{\raggedright}p{\glspagelistwidth}|}%
8608     {\end{longtable}}}%
    Place horizontal lines at the head and foot of the table:
8609   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8610 }

```

**ragged3colheader** The longragged3colheader style is like longragged3col but with a header row:

```

8611 \newglossarystyle{longragged3colheader}{%
    Base it on the glostylelongragged3col style:
8612   \setglossarystyle{longragged3col}%
    Set the table's header:
8613   \renewcommand*{\glossaryheader}{%
8614     \bfseries\entryname&\bfseries\descriptionname&
8615     \bfseries\pagelistname\tabularnewline\endhead}%
8616 }

```

**colheaderborder** The longragged3colheaderborder style is like the above but with a border

```

8617 \newglossarystyle{longragged3colheaderborder}{%
    Base it on the glostylelongragged3colborder style:
8618   \setglossarystyle{longragged3colborder}%
    Set the table's header and add horizontal line at table's foot:
8619   \renewcommand*{\glossaryheader}{%
8620     \hline
8621     \bfseries\entryname&\bfseries\descriptionname&
8622     \bfseries\pagelistname\tabularnewline\hline\endhead
8623     \hline\endfoot}%
8624 }

```

**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.

```

8625 \newglossarystyle{altlongragged4col}{%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8626 \renewenvironment{theglossary}%
8627   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8628     >{\raggedright}p{\glspagelistwidth}}}%
8629   {\end{longtable}}%
```

No table header:

```
8630 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8631 \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):

```
8632 \renewcommand{\glossentry}[2]{%
8633   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8634   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8635   ##2\tabularnewline
8636 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8637 \renewcommand{\subglossentry}[3]{%
8638   &
8639   \glssubentryitem{##2}%
8640   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8641   \glossentrysymbol{##2} & ##3\tabularnewline
8642 }%
```

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>)

```
8643 \ifglsgroupskip
8644   \renewcommand*{\glsgroupskip}{}%
8645 \else
8646   \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8647 \fi
8648 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8649 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8650 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8651 \renewenvironment{theglossary}%
8652   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
8653     >{\raggedright}p{\glspagelistwidth}}}%
8654   {\end{longtable}}%
```

Table has a header:

```
8655 \renewcommand*{\glossaryheader}{%
8656 \bfseries\entryname&\bfseries\descriptionname&
8657 \bfseries \symbolname&
8658 \bfseries\pagelistname\tabularnewline\endhead}%
8659 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8660 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8661 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8662 \renewenvironment{theglossary}%
8663 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8664 >\raggedright}p{\glspagelistwidth}|}%
8665 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8666 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8667 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8668 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8669 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8670 \renewenvironment{theglossary}%
8671 {\begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|l|}%
8672 >\raggedright}p{\glspagelistwidth}|}%
8673 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8674 \renewcommand*{\glossaryheader}{%
8675 \hline\bfseries\entryname&\bfseries\descriptionname&
8676 \bfseries \symbolname&
8677 \bfseries\pagelistname\tabularnewline\hline\endhead
8678 \hline\endfoot}%
8679 }
```

### 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.

```
8680 \ProvidesPackage{glossary-mcols}[2017/08/24 v4.32 (NLCT)]
```



Required packages:

```
8681 \RequirePackage{multicol}
8682 \RequirePackage{glossary-tree}
```

`\indexspace` The are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8683 \providecommand{\indexspace}{%
8684   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8685 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8686 \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.)

```
8687 \newglossarystyle{mcolindex}{%
8688   \setglossarystyle{index}%
8689   \renewenvironment{theglossary}%
8690     {%
8691       \begin{multicols}{\glsmcols}
8692       \setlength{\parindent}{0pt}%
8693       \setlength{\parskip}{0pt plus 0.3pt}%
8694       \let\item\glstreeitem
8695       \let\subitem\glstreesubitem
8696       \let\subsubitem\glstreesubsubitem
8697     }%
8698     {\end{multicols}}}%
8699 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8700 \newglossarystyle{mcolindexgroup}{%
8701   \setglossarystyle{mcolindex}%
8702   \renewcommand*{\glsgroupheading}[1]{%
8703     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8704 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8705 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8706   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8707   \renewcommand*{\glossaryheader}{%
8708     \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.

```
8709 \renewcommand*{\glsgroupheading}[1]{%
8710 \item\glstreegroupheaderfmt
8711 {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8712 \indexspace}%
8713 }
```

**colindexspannav** Similar to **mcindexhypergroup**, but puts the navigation line in the optional argument of **multicols**.

```
8714 \newglossarystyle{colindexspannav}{%
8715 \setglossarystyle{index}%
8716 \renewenvironment{theglossary}%
8717 {%
8718 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8719 \setlength{\parindent}{0pt}%
8720 \setlength{\parskip}{0pt plus 0.3pt}%
8721 \let\item\glstreeitem}%
8722 {\end{multicols}}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8723 \renewcommand*{\glsgroupheading}[1]{%
8724 \item\glstreegroupheaderfmt
8725 {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}%
8726 \indexspace}%
8727 }
```

**mcoltree** Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```
8728 \newglossarystyle{mcoltree}{%
8729 \setglossarystyle{tree}%
8730 \renewenvironment{theglossary}%
8731 {%
8732 \begin{multicols}{\glsmcols}
8733 \setlength{\parindent}{0pt}%
8734 \setlength{\parskip}{0pt plus 0.3pt}%
8735 }%
8736 {\end{multicols}}%
8737 }
```

**mcoltreegroup** Like the **mcoltree** style but the glossary groups have headings.

```
8738 \newglossarystyle{mcoltreegroup}{%
Base it on the glostylemcoltree style:
8739 \setglossarystyle{mcoltree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
8740 \renewcommand{\glsgroupheading}[1]{\par
8741 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8742 }
```

**ltreehypergroup** The `mcoltreehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
8743 \newglossarystyle{mcoltreehypergroup}{%
```

Base it on the `glostylemcoltree` style:

```
8744 \setglossarystyle{mcoltree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8745 \renewcommand*{\glossaryheader}{%
8746 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8747 \renewcommand*{\glsgroupheading}[1]{%
8748 \par\noindent
8749 \glstreegroupheaderfmt{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
8750 \indexspace}%
8751 }
```

**mcoltreespannav** Similar to the `mcoltreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8752 \newglossarystyle{mcoltreespannav}{%
8753 \setglossarystyle{tree}%
8754 \renewenvironment{theglossary}%
8755 {%
8756 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8757 \setlength{\parindent}{0pt}%
8758 \setlength{\parskip}{0pt plus 0.3pt}%
8759 }%
8760 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8761 \renewcommand*{\glsgroupheading}[1]{%
8762 \par\noindent
8763 \glstreegroupheaderfmt{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
8764 \indexspace}%
8765 }
```

**mcoltreenoname** Multi-column index style. Same as the `treenoname`, but puts the glossary in multiple columns.

```
8766 \newglossarystyle{mcoltreenoname}{%
8767 \setglossarystyle{treenoname}%
8768 \renewenvironment{theglossary}%
8769 {%
```

```

8770     \begin{multicols}{\glsmcols}
8771     \setlength{\parindent}{0pt}%
8772     \setlength{\parskip}{0pt plus 0.3pt}%
8773 }%
8774 {\end{multicols}}%
8775 }

```

**treenonamegroup** Like the `mcoltreenoname` style but the glossary groups have headings.

```

8776 \newglossarystyle{mcoltreenonamegroup}{%
    Base it on the glostylemcoltreenoname style:
8777 \setglossarystyle{mcoltreenoname}%
    Give each group a heading:
8778 \renewcommand{\glsgroupheading}[1]{\par
8779 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8780 }

```

**onamehypergroup** The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

8781 \newglossarystyle{mcoltreenonamehypergroup}{%
    Base it on the glostylemcoltreenoname style:
8782 \setglossarystyle{mcoltreenoname}%
    Put navigation links to the groups at the start of the theglossary environment:
8783 \renewcommand*{\glossaryheader}{%
8784 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8785 \renewcommand*{\glsgroupheading}[1]{%
8786 \par\noindent
8787 \glstreegroupheaderfmt{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
8788 \indexspace}%
8789 }

```

**eenonamespannav** Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8790 \newglossarystyle{mcoltreenonamespannav}{%
8791 \setglossarystyle{treenoname}%
8792 \renewenvironment{theglossary}%
8793 {%
8794 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8795 \setlength{\parindent}{0pt}%
8796 \setlength{\parskip}{0pt plus 0.3pt}%
8797 }%
8798 {\end{multicols}}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
8799 \renewcommand*{\glsgroupheading}[1]{%
8800 \par\noindent

```

```

8801 \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8802 \indexspace}%
8803 }

```

**mcolalttree** Multi-column index style. Same as the alttree, but puts the glossary in multiple columns.

```

8804 \newglossarystyle{mcolalttree}{%
8805 \setglossarystyle{alttree}%
8806 \renewenvironment{theglossary}%
8807 {%
8808 \begin{multicols}{\glsmcols}
8809 \def\@gls@prevlevel{-1}%
8810 \mbox{}\par
8811 }%
8812 {\par\end{multicols}}}%
8813 }

```

**colalttreegroup** Like the mcolalttree style but the glossary groups have headings.

```

8814 \newglossarystyle{colalttreegroup}{%
      Base it on the glostylemcolalttree style:
8815 \setglossarystyle{mcolalttree}%
      Give each group a heading.
8816 \renewcommand{\glsgroupheading}[1]{\par
8817 \def\@gls@prevlevel{-1}%
8818 \hangindent0pt\relax
8819 \parindent0pt\relax
8820 \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}\par\indexspace}%
8821 }

```

**treehypergroup** The mcolalttreehypergroup style is like the mcolalttreegroup style, but has a set of links to the groups at the start of the glossary.

```

8822 \newglossarystyle{mcolalttreehypergroup}{%
      Base it on the glostylemcolalttree style:
8823 \setglossarystyle{mcolalttree}%
      Put the navigation links in the header
8824 \renewcommand*{\glossaryheader}{%
8825 \par
8826 \def\@gls@prevlevel{-1}%
8827 \hangindent0pt\relax
8828 \parindent0pt\relax
8829 \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8830 \renewcommand*{\glsgroupheading}[1]{%
8831 \par
8832 \def\@gls@prevlevel{-1}%
8833 \hangindent0pt\relax

```

```

8834 \parindent0pt\relax
8835 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8836 \indexspace}%
8837 }

```

`lalttreespannav` Similar to the `mcolalttreehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```

8838 \newglossarystyle{mcolalttreespannav}{%
8839 \setglossarystyle{alttree}%
8840 \renewenvironment{theglossary}%
8841 {%
8842 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8843 \def\@gls@prevlevel{-1}%
8844 \mbox{}\par
8845 }%
8846 {\par\end{multicols}}}%

```

Put a hypertarget at the start of each group

```

8847 \renewcommand*{\glsgroupheading}[1]{%
8848 \par
8849 \def\@gls@prevlevel{-1}%
8850 \hangindent0pt\relax
8851 \parindent0pt\relax
8852 \glstreegroupheaderfmt{\glshnavhypertarget{##1}{\glsggetgrouptitle{##1}}}\par
8853 \indexspace}
8854 }

```

### 3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the `supertabular` environment.

```

8855 \ProvidesPackage{glossary-super}[2017/08/24 v4.32 (NLCT)]

```

Requires the package:

```

8856 \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.

```

8857 \@ifundefined{glsdescwidth}{%
8858 \newlength\glsdescwidth
8859 \setlength{\glsdescwidth}{0.6\hsize}
8860 }{}

```

`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.

```

8861 \@ifundefined{glspagelistwidth}{%
8862 \newlength\glspagelistwidth
8863 \setlength{\glspagelistwidth}{0.1\hsize}

```

8864 }{}

**super** The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

8865 \newglossarystyle{super}{%

Put the glossary in a supertabular environment with two columns and no head or tail:

```
8866 \renewenvironment{theglossary}%  
8867 {\tablehead{}\tabletail{}}%  
8868 \begin{supertabular}{lp{\glsdescwidth}}%  
8869 {\end{supertabular}}%
```

Do nothing at the start of the table:

8870 \renewcommand\*{\glossaryheader}{}%

No group headings:

8871 \renewcommand\*{\glsgroupheading}[1]{}%

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8872 \renewcommand{\glossentry}[2]{%  
8873 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &  
8874 \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline  
8875 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8876 \renewcommand{\subglossentry}[3]{%  
8877 &  
8878 \glssubentryitem{##2}%  
8879 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space  
8880 ##3\tabularnewline  
8881 }%
```

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>)

```
8882 \ifglsnogroupskip  
8883 \renewcommand*{\glsgroupskip}{}%  
8884 \else  
8885 \renewcommand*{\glsgroupskip}{& \tabularnewline}%  
8886 \fi  
8887 }
```

**superborder** The superborder style is like the above, but with horizontal and vertical lines:

8888 \newglossarystyle{superborder}{%

Base it on the glostylesuper style:

8889 \setglossarystyle{super}%

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
8890 \renewenvironment{theglossary}%  
8891 {\tablehead{\hline}\tabletail{\hline}%
```

```

8892     \begin{supertabular}{|l|p{\glsdescwidth}|}%
8893     {\end{supertabular}}%
8894 }

```

**superheader** The superheader style is like the super style, but with a header:

```
8895 \newglossarystyle{superheader}{%
```

Base it on the `glostylesuper` style:

```
8896 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

8897 \renewenvironment{theglossary}%
8898 {\tablehead{\bfseries \entryname &
8899 \bfseries\descriptionname\tabularnewline}%
8900 \tabletail{}}%
8901 \begin{supertabular}{lp{\glsdescwidth}}%
8902 {\end{supertabular}}%
8903 }

```

**superheaderborder** The superheaderborder style is like the super style but with a header and border:

```
8904 \newglossarystyle{superheaderborder}{%
```

Base it on the `glostylesuper` style:

```
8905 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

8906 \renewenvironment{theglossary}%
8907 {\tablehead{\hline\bfseries \entryname &
8908 \bfseries \descriptionname\tabularnewline\hline}%
8909 \tabletail{\hline}
8910 \begin{supertabular}{|l|p{\glsdescwidth}|}%
8911 {\end{supertabular}}%
8912 }

```

**super3col** The super3col style is like the super style, but with 3 columns:

```
8913 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

8914 \renewenvironment{theglossary}%
8915 {\tablehead{}\tabletail{}}%
8916 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
8917 {\end{supertabular}}%

```

Do nothing at the start of the table:

```
8918 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8919 \renewcommand*{\glsgroupheading}[1]{}%
```



Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8920 \renewcommand{\glossentry}[2]{%
8921   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8922   \glossentrydesc{##1} & ##2\tabularnewline
8923 }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8924 \renewcommand{\subglossentry}[3]{%
8925   &
8926   \glssubentryitem{##2}%
8927   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8928   ##3\tabularnewline
8929 }%
```

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>)

```
8930 \ifglsgroupskip
8931   \renewcommand*{\glsgroupskip}{}%
8932 \else
8933   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8934 \fi
8935 }
```

**super3colborder** The super3colborder style is like the super3col style, but with a border:

```
8936 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
8937 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
8938 \renewenvironment{theglossary}%
8939   {\tablehead{\hline}\tabletail{\hline}%
8940   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth|}}%
8941   {\end{supertabular}}%
8942 }
```

**super3colheader** The super3colheader style is like the super3col style but with a header row:

```
8943 \newglossarystyle{super3colheader}{%
```

Base it on the glostylesuper3col style:

```
8944 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
8945 \renewenvironment{theglossary}%
8946   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8947   \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8948   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
8949   {\end{supertabular}}%
8950 }
```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
8951 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
8952 \setglossarystyle{super3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
8953 \renewenvironment{theglossary}{%
8954   {\tablehead{\hline
8955     \bfseries\entryname&\bfseries\descriptionname&
8956     \bfseries\pagelistname\tabularnewline\hline}%
8957   \tabletail{\hline}%
8958   \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
8959   {\end{supertabular}}}%
8960 }
```

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.

```
8961 \newglossarystyle{super4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8962 \renewenvironment{theglossary}{%
8963   {\tablehead{}\tabletail{}}%
8964   \begin{supertabular}{|l|l|l|l|}%
8965   \end{supertabular}}%
```

Do nothing at the start of the table:

```
8966 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8967 \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:

```
8968 \renewcommand{\glossentry}[2]{%
8969   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8970   \glossentrydesc{##1} &
8971   \glossentrysymbol{##1} & ##2\tabularnewline
8972 }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
8973 \renewcommand{\subglossentry}[3]{%
8974   &
8975   \glssubentryitem{##2}%
8976   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8977   \glossentrysymbol{##2} & ##3\tabularnewline
8978 }%
```

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>)

```
8979 \ifglsgroupskip
8980 \renewcommand*{\glsgroupskip}{}%
8981 \else
8982 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8983 \fi
8984 }
```

**super4colheader** The super4colheader style is like the super4col but with a header row.

```
8985 \newglossarystyle{super4colheader}{%
    Base it on the glostylesuper4col style:
8986 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns, a header and no tail:
8987 \renewenvironment{theglossary}%
8988 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8989 \bfseries\symbolname &
8990 \bfseries\pagelistname\tabularnewline}%
8991 \tabletail{}}%
8992 \begin{supertabular}{l111}}%
8993 {\end{supertabular}}%
8994 }
```

**super4colborder** The super4colborder style is like the super4col but with a border.

```
8995 \newglossarystyle{super4colborder}{%
    Base it on the glostylesuper4col style:
8996 \setglossarystyle{super4col}%
    Put the glossary in a supertabular environment with four columns and a horizontal line in the
    head and tail:
8997 \renewenvironment{theglossary}%
8998 {\tablehead{\hline}\tabletail{\hline}%
8999 \begin{supertabular}{|l11|11|}}%
9000 {\end{supertabular}}%
9001 }
```

**colheaderborder** The super4colheaderborder style is like the super4col but with a header and border.

```
9002 \newglossarystyle{super4colheaderborder}{%
    Base it on the glostylesuper4col style:
9003 \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:
9004 \renewenvironment{theglossary}%
9005 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
9006 \bfseries\symbolname &
```

```

9007      \bfseries\pagelistname\tabularnewline\hline}%
9008      \tabletail{\hline}%
9009      \begin{supertabular}{|l|l|l|l|}%
9010      {\end{supertabular}}%
9011 }

```

**altsuper4col** The altsuper4col glossary style is like super4col but has provision for multiline descriptions.

```

9012 \newglossarystyle{altsuper4col}{%
      Base it on the glostylesuper4col style:
9013   \setglossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and no head or tail:
9014   \renewenvironment{theglossary}%
9015     {\tablehead{}\tabletail{}}%
9016     \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
9017     {\end{supertabular}}%
9018 }

```

**super4colheader** The altsuper4colheader style is like the altsuper4col but with a header row.

```

9019 \newglossarystyle{altsuper4colheader}{%
      Base it on the glostylesuper4colheader style:
9020   \setglossarystyle{super4colheader}%
      Put the glossary in a supertabular environment with four columns, a header and no tail:
9021   \renewenvironment{theglossary}%
9022     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9023       \bfseries\symbolname &
9024       \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9025     \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}%
9026     {\end{supertabular}}%
9027 }

```

**super4colborder** The altsuper4colborder style is like the altsuper4col but with a border.

```

9028 \newglossarystyle{altsuper4colborder}{%
      Base it on the glostylesuper4colborder style:
9029   \setglossarystyle{super4colborder}%
      Put the glossary in a supertabular environment with four columns and a horizontal line in the
      head and tail:
9030   \renewenvironment{theglossary}%
9031     {\tablehead{\hline}\tabletail{\hline}%
9032     \begin{supertabular}%
9033       {|lp{\glsgdescwidth}|lp{\glspagelistwidth}|}%
9034     {\end{supertabular}}%
9035 }

```

**colheaderborder** The altsuper4colheaderborder style is like the altsuper4col but with a header and border.

```

9036 \newglossarystyle{altsuper4colheaderborder}{%

```

Base it on the `glostylesuper4colheaderborder` style:

```
9037 \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:

```
9038 \renewenvironment{theglossary}%
9039   {\tablehead{\hline
9040     \bfseries\entryname &
9041     \bfseries\descriptionname &
9042     \bfseries\symbolname &
9043     \bfseries\pagelistname\tabularnewline\hline}%
9044   \tabletail{\hline}%
9045   \begin{supertabular}%
9046     {lllp{\glsdescwidth}llp{\glspagelistwidth}}}%
9047   {\end{supertabular}}%
9048 }
```

### 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.

```
9049 \ProvidesPackage{glossary-superragged}[2017/08/24 v4.32 (NLCT)]
```

Requires the package:

```
9050 \RequirePackage{array}
```

Requires the package:

```
9051 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```
9052 \@ifundefined{glsdescwidth}{%
9053   \newlength\glsdescwidth
9054   \setlength{\glsdescwidth}{0.6\hsize}
9055 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```
9056 \@ifundefined{glspagelistwidth}{%
9057   \newlength\glspagelistwidth
9058   \setlength{\glspagelistwidth}{0.1\hsize}
9059 }{}
```

`superragged` The `superragged` glossary style uses the `supertabular` environment.

```
9060 \newglossarystyle{superragged}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```
9061 \renewenvironment{theglossary}%
9062   {\tablehead{}\tabletail{}}%
9063   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9064   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9065 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9066 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
9067 \renewcommand{\glossentry}[2]{%
9068   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9069   \glossentrydesc{##1}\glspostdescription\space ##2%
9070   \tabularnewline
9071 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
9072 \renewcommand{\subglossentry}[3]{%
9073   &
9074   \glssubentryitem{##2}%
9075   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9076   ##3%
9077   \tabularnewline
9078 }%
```

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>)

```
9079 \ifglsgroupskip
9080   \renewcommand*{\glsgroupskip}{}%
9081 \else
9082   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9083 \fi
9084 }
```

`\perraggedborder` The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
9085 \newglossarystyle{superraggedborder}{}%
```

Base it on the `glostylesuperragged` style:

```
9086 \setglossarystyle{superragged}%
```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```
9087 \renewenvironment{theglossary}%
9088   {\tablehead{\hline}\tabletail{\hline}%
9089   \begin{supertabular}{l|l>{\raggedright}p{\glsgdescwidth}|}%
9090   {\end{supertabular}}%
9091 }
```

`superraggedheader` The `superraggedheader` style is like the `super` style, but with a header:

```
9092 \newglossarystyle{superraggedheader}{%
```

Base it on the `glostylesuperragged` style:

```
9093 \setglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns, a header and no tail:

```
9094 \renewenvironment{theglossary}{%
```

```
9095 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
```

```
9096 \tabularnewline}%
```

```
9097 \tabletail{}}%
```

```
9098 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}%
```

```
9099 {\end{supertabular}}%
```

```
9100 }
```

`superraggedheaderborder` The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
9101 \newglossarystyle{superraggedheaderborder}{%
```

Base it on the `glostylesuper` style:

```
9102 \setglossarystyle{superragged}{%
```

Put the glossary in a `supertabular` environment with two columns, a header and horizontal lines above and below the table:

```
9103 \renewenvironment{theglossary}{%
```

```
9104 {\tablehead{\hline\bfseries \entryname &
```

```
9105 \bfseries \descriptionname\tablearnewline\hline}%
```

```
9106 \tabletail{\hline}
```

```
9107 \begin{supertabular}{ll|>{\raggedright}p{\glsgdescwidth}}%
```

```
9108 {\end{supertabular}}%
```

```
9109 }
```

`superragged3col` The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
9110 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
9111 \renewenvironment{theglossary}{%
```

```
9112 {\tablehead{}\tabletail{}}%
```

```
9113 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
```

```
9114 >{\raggedright}p{\glspagelistwidth}}%
```

```
9115 {\end{supertabular}}%
```

Do nothing at the start of the table:

```
9116 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9117 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
9118 \renewcommand{\glossentry}[2]{%
```

```
9119 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
```

```
9120 \glossentrydesc{##1} &
```

```

9121     ##2\tabularnewline
9122 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9123 \renewcommand{\subglossentry}[3]{%
9124     &
9125     \glssubentryitem{##2}%
9126     \glstarget{##2}{\strut}\glossentrydesc{##2} &
9127     ##3\tabularnewline
9128 }%

```

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>)

```

9129 \ifglsnogroupskip
9130 \renewcommand*{\glsgroupskip}{}%
9131 \else
9132 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9133 \fi
9134 }

```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```

9135 \newglossarystyle{superragged3colborder}{%

```

Base it on the glostypesuperragged3col style:

```

9136 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

9137 \renewenvironment{theglossary}%
9138 {\tablehead{\hline}\tabletail{\hline}%
9139 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}||%
9140 >{\raggedright}p{\glspagelistwidth}||}%
9141 {\end{supertabular}}%
9142 }

```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```

9143 \newglossarystyle{superragged3colheader}{%

```

Base it on the glostypesuperragged3col style:

```

9144 \setglossarystyle{superragged3col}%

```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

9145 \renewenvironment{theglossary}%
9146 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9147 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9148 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9149 >{\raggedright}p{\glspagelistwidth}}%
9150 {\end{supertabular}}%
9151 }

```



colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9152 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostylesuperragged3colborder style:

```
9153 \setglossarystyle{superragged3colborder}{%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9154 \renewenvironment{theglossary}{%
9155   {\tablehead{\hline
9156     \bfseries\entryname&\bfseries\descriptionname&
9157     \bfseries\pagelistname\tabularnewline\hline}%
9158   \tabletail{\hline}%
9159   \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|%
9160     >{\raggedright}p{\glspagelistwidth}|}%
9161   {\end{supertabular}}}%
9162 }
```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```
9163 \newglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
9164 \renewenvironment{theglossary}{%
9165   {\tablehead{}\tabletail{}%
9166   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9167     >{\raggedright}p{\glspagelistwidth}}}%
9168   {\end{supertabular}}}%
```

Do nothing at the start of the table:

```
9169 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9170 \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:

```
9171 \renewcommand{\glossentry}[2]{%
9172   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9173   \glossentrydesc{##1} &
9174   \glossentrysymbol{##1} & ##2\tabularnewline
9175   }%
```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
9176 \renewcommand{\subglossentry}[3]{%
9177   &
9178   \glssubentryitem{##2}%
9179   \glstarget{##2}{\strut}\glossentrydesc{##2} &
9180   \glossentrysymbol{##2} & ##3\tabularnewline
9181   }%
```

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>)

```

9182 \ifglsgroupskip
9183 \renewcommand*{\glsgroupskip}{}%
9184 \else
9185 \renewcommand*{\glsgroupskip}{& & & \tabularnewline}%
9186 \fi
9187 }

```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```

9188 \newglossarystyle{altsuperragged4colheader}{%

```

Base it on the glostylealtsuperragged4col style:

```

9189 \setglossarystyle{altsuperragged4col}%

```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```

9190 \renewenvironment{theglossary}%
9191 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9192 \bfseries\symbolname &
9193 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9194 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9195 >{\raggedright}p{\glspagelistwidth}}}%
9196 {\end{supertabular}}%
9197 }

```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```

9198 \newglossarystyle{altsuperragged4colborder}{%

```

Base it on the glostylealtsuperragged4col style:

```

9199 \setglossarystyle{altsuper4col}%

```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```

9200 \renewenvironment{theglossary}%
9201 {\tablehead{\hline}\tabletail{\hline}%
9202 \begin{supertabular}%
9203 {l|>{\raggedright}p{\glsdescwidth}l|}%
9204 >{\raggedright}p{\glspagelistwidth}l}}%
9205 {\end{supertabular}}%
9206 }

```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```

9207 \newglossarystyle{altsuperragged4colheaderborder}{%

```

Base it on the glostylealtsuperragged4col style:

```

9208 \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:

```

9209 \renewenvironment{theglossary}%
9210   {\tablehead{\hline
9211     \bfseries\entryname &
9212     \bfseries\descriptionname &
9213     \bfseries\symbolname &
9214     \bfseries\pagelistname\tabularnewline\hline}%
9215   \tabletail{\hline}%
9216   \begin{supertabular}%
9217     {|||>{\raggedright}p{\glsdescwidth}|||}%
9218     >{\raggedright}p{\glspagelistwidth}||}%
9219   {\end{supertabular}}%
9220 }

```

### 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.

```

9221 \ProvidesPackage{glossary-tree}[2017/08/24 v4.32 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9222 \providecommand{\indexspace}{%
9223   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9224 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glstnamefont`.) This command was previously also used to format the group headings.

```

9225 \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.

```

9226 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9227 \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.

```

9228 \ifdef\@idxitem
9229 {\newcommand{\glstreeitem}{\@idxitem}}
9230 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```
9231 \ifdef\subitem
9232 {\let\glstreesubitem\subitem}
9233 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}
```

`\glstreesubsubitem` Level 1 item used in index style.

```
9234 \ifdef\subsubitem
9235 {\let\glstreesubsubitem\subsubitem}
9236 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}
```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `alttree` style).

```
9237 \newcommand{\glstreepredesc}{\space}
```

`\glstreechildpredesc` Allow the user to adjust the space before the description for sub-entries (except for the `treenoname` and `alttree` style).

```
9238 \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.

```
9239 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```
9240 \renewenvironment{theglossary}%
9241 {\setlength{\parindent}{0pt}}%
9242 {\setlength{\parskip}{0pt plus 0.3pt}}%
9243 \let\item\glstreeitem
9244 \let\subitem\glstreesubitem
9245 \let\subsubitem\glstreesubsubitem
9246 }%
```

```
9247 {\par}}%
```

Do nothing at the start of the environment:

```
9248 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9249 \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.

```
9250 \renewcommand*{\glossentry}[2]{%
9251 \item\glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9252 \ifglshassymbol{##1}{\space\glossentrysymbol{##1}}{}}%
9253 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9254 }%
```

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.

```

9255 \renewcommand{\subglossentry}[3]{%
9256   \ifcase##1\relax
9257     % level 0
9258     \item
9259   \or
9260     % level 1
9261     \subitem
9262     \glssubentryitem{##2}%
9263   \else
9264     % all other levels
9265     \subsubitem
9266   \fi
9267   \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}%
9268   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9269   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9270 }%
```

Vertical gap between groups is the same as that used by indices:

```

9271 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

**indexgroup** The `indexgroup` style is like the `index` style but has headings.

```

9272 \newglossarystyle{indexgroup}{%
```

Base it on the `glostyleindex` style:

```

9273 \setglossarystyle{index}%
```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9274 \renewcommand*{\glsgroupheading}[1]{%
9275   \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9276   \indexspace
9277 }%
9278 }
```

**indexhypergroup** The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

9279 \newglossarystyle{indexhypergroup}{%
```

Base it on the `glostyleindex` style:

```

9280 \setglossarystyle{index}%
```

Put navigation links to the groups at the start of the glossary:

```

9281 \renewcommand*{\glossaryheader}{%
9282   \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.

```

9283 \renewcommand*{\glsgroupheading}[1]{%
9284   \item\glstreegroupheaderfmt

```

```

9285      {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
9286      \indexspace}%
9287 }

```

**tree** The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9288 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9289 \renewenvironment{theglossary}%
9290 {\setlength{\parindent}{0pt}%
9291 \setlength{\parskip}{0pt plus 0.3pt}}%
9292 {}%

```

Do nothing at the start of the theglossary environment:

```

9293 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9294 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

9295 \renewcommand{\glossentry}[2]{%
9296 \hangindent0pt\relax
9297 \parindent0pt\relax
9298 \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9299 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9300 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9301 }%

```

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.

```

9302 \renewcommand{\subglossentry}[3]{%
9303 \hangindent##1\glstreeindent\relax
9304 \parindent##1\glstreeindent\relax
9305 \ifnum##1=1\relax
9306 \glssubentryitem{##2}%
9307 \fi
9308 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9309 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9310 \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9311 }%

```

Vertical gap between groups is the same as that used by indices:

```

9312 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

```

**treegroup** Like the tree style but the glossary groups have headings.

```

9313 \newglossarystyle{treegroup}{%

```

Base it on the glostyletree style:

```

9314 \setglossarystyle{tree}%

```

Each group has a heading (in bold) followed by a vertical gap):

```
9315 \renewcommand{\glsgroupheading}[1]{\par
9316 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
9317 \indexspace}%
9318 }
```

**treehypergroup** The **treehypergroup** style is like the **treegroup** style, but has a set of links to the groups at the start of the glossary.

```
9319 \newglossarystyle{treehypergroup}{%
```

Base it on the **glostyletree** style:

```
9320 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the **theglossary** environment:

```
9321 \renewcommand*{\glossaryheader}{%
9322 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9323 \renewcommand*{\glsgroupheading}[1]{%
9324 \par\noindent
9325 \glstreegroupheaderfmt
9326 {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
9327 \indexspace}%
9328 }
```

**\glstreeindent** Length governing left indent for each level of the tree style.

```
9329 \newlength\glstreeindent
9330 \setlength{\glstreeindent}{10pt}
```

**treenoname** The **treenoname** glossary style is like the **tree** style, but doesn't print the name or symbol for sub-levels.

```
9331 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
9332 \renewenvironment{theglossary}%
9333 {\setlength{\parindent}{0pt}%
9334 \setlength{\parskip}{0pt plus 0.3pt}}%
9335 {}%
```

No header:

```
9336 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9337 \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.

```
9338 \renewcommand{\glossentry}[2]{%
9339 \hangindent0pt\relax
9340 \parindent0pt\relax
9341 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
```

```

9342 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9343 \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9344 }%

```

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.

```

9345 \renewcommand{\subglossentry}[3]{%
9346 \hangindent##1\glstreeindent\relax
9347 \parindent##1\glstreeindent\relax
9348 \ifnum##1=1\relax
9349 \glssubentryitem{##2}%
9350 \fi
9351 \glstarget{##2}{\strut}%
9352 \glossentrydesc{##2}\glspostdescription\space##3\par
9353 }%

```

Vertical gap between groups is the same as that used by indices:

```

9354 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9355 }

```

`treenonamegroup` Like the `treenoname` style but the glossary groups have headings.

```

9356 \newglossarystyle{treenonamegroup}{%
  Base it on the glostyletreenoname style:
9357 \setglossarystyle{treenoname}%
  Give each group a heading:
9358 \renewcommand{\glsgroupheading}[1]{\par
9359 \noindent\glstreegroupheaderfmt
9360 {\glsgrouptitle{##1}}\par\indexspace}%
9361 }

```

`onamehypergroup` The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

9362 \newglossarystyle{treenonamehypergroup}{%
  Base it on the glostyletreenoname style:
9363 \setglossarystyle{treenoname}%
  Put navigation links to the groups at the start of the theglossary environment:
9364 \renewcommand*{\glossaryheader}{%
9365 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
  Each group has a heading (in bold with a target) followed by a vertical gap):
9366 \renewcommand*{\glsgroupheading}[1]{%
9367 \par\noindent
9368 \glstreegroupheaderfmt
9369 {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
9370 \indexspace}%
9371 }

```



`esttoplevelname` Find the widest name over all parentless entries in the given glossary or glossaries.

```

9372 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9373   \dimen@=0pt\relax
9374   \gls@tmplen=0pt\relax
9375   \forallglossaries[#1]{\@gls@type}%
9376   {%
9377     \forallglsentries[\@gls@type]{\@glo@label}%
9378     {%
9379       \ifglsahasparent{\@glo@label}%
9380       }%
9381       {%
9382         \settowidth{\dimen@}%
9383           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
9384         \ifdim\dimen@>\gls@tmplen
9385           \gls@tmplen=\dimen@
9386           \letcs{\@glswidestname}{glo\glsdetoklabel{\@glo@label}@name}%
9387         \fi
9388       }%
9389     }%
9390   }%
9391 }
```

`\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.

```

9392 \newcommand*{\glssetwidest}[2][0]{%
9393   \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9394     #2}%
9395 }
```

`\@glswidestname` Initialise `\@glswidestname`.

```

9396 \newcommand*{\@glswidestname}{}
```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9397 \newcommand*{\glstreenamebox}[2]{%
9398   \makebox[#1][l]{#2}%
9399 }
```

`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`.

```

9400 \newglossarystyle{alttree}{%
  Redefine theglossary environment.
9401   \renewenvironment{theglossary}%
9402     {\def\@gls@prevlevel{-1}%
9403      \mbox{}\par}%
9404     {\par}%
  Set the header and group headers to nothing.
9405   \renewcommand*{\glossaryheader}{}%
9406   \renewcommand*{\glsgroupheading}[1]{}%
}
```

Redefine the way that the level 0 entries are displayed.

```
9407 \renewcommand{\glossentry}[2]{%
9408   \ifnum\@gls@prevlevel=0\relax
9409   \else
```

Find out how big the indentation should be by measuring the widest entry.

```
9410     \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9411   \fi
```

Set the hangindent and paragraph indent.

```
9412   \hangindent\glstreeindent
9413   \parindent\glstreeindent
```

Put the name to the left of the paragraph block.

```
9414   \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
9415     \glssentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9416   \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9417   \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9418   \def\@gls@prevlevel{0}%
9419 }%
```

Redefine the way sub-entries are displayed.

```
9420 \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.

```
9421   \ifnum##1=1\relax
9422     \glssubentryitem{##2}%
9423   \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9424   \ifnum\@gls@prevlevel=##1\relax
9425   \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9426     \@ifundefined{@glswidestname\romannumeral##1}{%
9427       \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%
9428       \settowidth{\gls@tmplen}{\glstreenamefmt{%
9429         \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9430   \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9431      \setlength\glstreeindent\gls@tmplen
9432      \addtolength\glstreeindent\parindent
9433      \parindent\glstreeindent
9434      \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`.

```
9435      \ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9436      \settowidth{\glstreeindent}{\glstreenamfmt{%
9437      \@glswidestname\space}}}{%
9438      \settowidth{\glstreeindent}{\glstreenamfmt{%
9439      \csname @glswidestname\romannumeral\@gls@prevlevel
9440      \endcsname\space}}}{%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9441      \addtolength\parindent{-\glstreeindent}%
9442      \setlength\glstreeindent\parindent
9443      \fi
9444      \fi
```

Set the hanging indentation.

```
9445      \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
9446      \makebox[0pt][r]{\glstreenambox{\gls@tmplen}{%
9447      \glstreenamfmt{\glstarget{##2}{\glossentryname{##2}}}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9448      \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9449      \glossentrydesc{##2}\glspostdescription\space ##3\par
```

Set the previous level macro to the current level.

```
9450      \def\@gls@prevlevel{##1}%
9451      }%
```

Vertical gap between groups is the same as that used by indices:

```
9452      \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9453 }
```

`almtreegroup` Like the `almtree` style but the glossary groups have headings.

```
9454 \newglossarystyle{almtreegroup}{%
```

Base it on the `glostylealmtree` style:

```
9455      \setglossarystyle{almtree}%
```

Give each group a heading.

```
9456      \renewcommand{\glsgroupheading}[1]{\par
9457      \def\@gls@prevlevel{-1}%
9458      \hangindent0pt\relax
```

```

9459     \parindent0pt\relax
9460     \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
9461     \par\indexspace}%
9462 }

```

alttreehypergroup The alttreehypergroup style is like the alttreegroup style, but has a set of links to the groups at the start of the glossary.

```

9463 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9464     \setglossarystyle{alttree}%
    Put the navigation links in the header
9465     \renewcommand*{\glossaryheader}{%
9466         \par
9467         \def\@gls@prevlevel{-1}%
9468         \hangindent0pt\relax
9469         \parindent0pt\relax
9470         \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
    Put a hypertarget at the start of each group
9471     \renewcommand*{\glsgroupheading}[1]{%
9472         \par
9473         \def\@gls@prevlevel{-1}%
9474         \hangindent0pt\relax
9475         \parindent0pt\relax
9476         \glstreegroupheaderfmt
9477         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
9478         \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.

```
9479 \NeedsTeXFormat{LaTeX2e}
9480 \ProvidesPackage{glossaries-compatible-207}[2017/08/24 v4.32 (NLCT)]
```

**AddXdyAttribute** Adds an attribute in old format.

```
9481 \ifglsxindy
9482   \renewcommand*\GlsAddXdyAttribute[1]{%
9483     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9484     \expandafter\toks@\expandafter{\@xdylocref}%
9485     \edef\@xdylocref{\the\toks@ ^^J}%
9486     (markup-locref
9487     :open \string"\string~n\string\setentrycounter
9488       {\noexpand\glscounter}%
9489       \expandafter\string\csname#1\endcsname
9490       \expandafter\@gobble\string\{\string" ^^J
9491       :close \string"\expandafter\@gobble\string\}\string" ^^J
9492       :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
9493 \fi
```

**sAddXdyCounters**

```
9494 \renewcommand*\GlsAddXdyCounters[1]{%
9495   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9496     in compatibility mode.}%
9497 }
```

Add predefined attributes

```
9498 \GlsAddXdyAttribute{glsnumberformat}
9499 \GlsAddXdyAttribute{textrm}
9500 \GlsAddXdyAttribute{textsf}
9501 \GlsAddXdyAttribute{texttt}
9502 \GlsAddXdyAttribute{textbf}
9503 \GlsAddXdyAttribute{textmd}
9504 \GlsAddXdyAttribute{textit}
9505 \GlsAddXdyAttribute{textup}
9506 \GlsAddXdyAttribute{textsl}
```

```

9507 \GlsAddXdyAttribute{textsc}
9508 \GlsAddXdyAttribute{emph}
9509 \GlsAddXdyAttribute{glshypernumber}
9510 \GlsAddXdyAttribute{hyperrm}
9511 \GlsAddXdyAttribute{hypersf}
9512 \GlsAddXdyAttribute{hypertt}
9513 \GlsAddXdyAttribute{hyperbf}
9514 \GlsAddXdyAttribute{hypermd}
9515 \GlsAddXdyAttribute{hyperit}
9516 \GlsAddXdyAttribute{hyperup}
9517 \GlsAddXdyAttribute{hypersl}
9518 \GlsAddXdyAttribute{hypersc}
9519 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9520 \ifglxindy
9521 \renewcommand*{\GlsAddXdyLocation}[2]{%
9522   \edef\@xdyuserlocationdefs{%
9523     \@xdyuserlocationdefs ^^J%
9524     (define-location-class \string"#1\string"^^J\space\space
9525     \space(#2))
9526   }%
9527   \edef\@xdyuserlocationnames{%
9528     \@xdyuserlocationnames^^J\space\space\space
9529     \string"#1\string"}%
9530 }
9531 \fi

```

\@do@wrglossary

```

9532 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9533 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9534 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9535 \def\@glo@range{}%
9536 \expandafter\if\@glo@prefix(\relax
9537   \def\@glo@range{:open-range}%
9538 \else
9539   \expandafter\if\@glo@prefix)\relax
9540   \def\@glo@range{:close-range}%
9541 \fi
9542 \fi

  Get the location and escape any special characters
9543 \protected@edef\@glslocref{\theglentrycounter}%
9544 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9545 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9546 (indexentry :tkey (\csname glo@#1@index\endcsname)
9547 :locoref \string"\@glslocoref\string" %
9548 :attr \string"\@glo@suffix\string" \@glo@range
9549 )
9550 }%
9551 \else

```

Convert the format information into the format required for makeindex

```

9552 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9553 \glossary[\csname glo@#1@type\endcsname]{%
9554 \string\glossaryentry{\csname glo@#1@index\endcsname
9555 \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9556 \fi
9557 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9558 \def\@set@glo@numformat#1#2#3{%
9559 \expandafter\@glo@check@mkidxrangechar#3\@nil
9560 \protected@edef#1{%
9561 \@glo@prefix setentrycounter[]{\#2}%
9562 \expandafter\string\csname\@glo@suffix\endcsname
9563 }%
9564 \@gls@checkmkidxchars#1%
9565 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9566 \ifglxindy
9567 \def\writeist{%
9568 \openout\glswrite=\istfilename
9569 \write\glswrite{;; xindy style file created by the glossaries
9570 package in compatible-2.07 mode}%
9571 \write\glswrite{;; for document '\jobname' on
9572 \the\year-\the\month-\the\day}%
9573 \write\glswrite{^^J; required styles^^J}
9574 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9575 \ifx\@xdystyle\@empty
9576 \else
9577 \protected@write\glswrite{{(require
9578 \string"\@xdystyle.xdy\string")}}%
9579 \fi
9580 }%
9581 \write\glswrite{^^J%
9582 ; list of allowed attributes (number formats)^^J}%
9583 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9584 \write\glswrite{^^J; user defined alphabets^^J}%
9585 \write\glswrite{\@xdyuseralphabets}%
9586 \write\glswrite{^^J; location class definitions^^J}%
9587 \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9588     \string"roman-numbers-lowercase\string" :sep \string"}}%
9589 \@onelevel@sanitize\@gls@roman
9590 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9591     :sep \string"}}%
9592 \@onelevel@sanitize\@tmp
9593 \ifx\@tmp\@gls@roman
9594     \write\glswrite{(define-location-class
9595         \string"roman-page-numbers\string"^^J\space\space\space
9596         (\string"roman-numbers-lowercase\string")
9597         :min-range-length \@glsminrange)}}%
9598 \else
9599     \write\glswrite{(define-location-class
9600         \string"roman-page-numbers\string"^^J\space\space\space
9601         (:sep "\@gls@roman")
9602         :min-range-length \@glsminrange)}}%
9603 \fi
9604 \write\glswrite{(define-location-class
9605     \string"Roman-page-numbers\string"^^J\space\space\space
9606     (\string"roman-numbers-uppercase\string")
9607     :min-range-length \@glsminrange)}}%
9608 \write\glswrite{(define-location-class
9609     \string"arabic-page-numbers\string"^^J\space\space\space
9610     (\string"arabic-numbers\string")
9611     :min-range-length \@glsminrange)}}%
9612 \write\glswrite{(define-location-class
9613     \string"alpha-page-numbers\string"^^J\space\space\space
9614     (\string"alpha\string")
9615     :min-range-length \@glsminrange)}}%
9616 \write\glswrite{(define-location-class
9617     \string"Alpha-page-numbers\string"^^J\space\space\space
9618     (\string"ALPHA\string")
9619     :min-range-length \@glsminrange)}}%
9620 \write\glswrite{(define-location-class
9621     \string"Appendix-page-numbers\string"^^J\space\space\space
9622     (\string"ALPHA\string"
9623     :sep \string"\@glsAlphacompositor\string"
9624     \string"arabic-numbers\string")
9625     :min-range-length \@glsminrange)}}%
9626 \write\glswrite{(define-location-class
9627     \string"arabic-section-numbers\string"^^J\space\space\space
9628     (\string"arabic-numbers\string"
9629     :sep \string"\glscompositor\string"
9630     \string"arabic-numbers\string")
9631     :min-range-length \@glsminrange)}}%
9632 \write\glswrite{^^J; user defined location classes}%
9633 \write\glswrite{\@xdyuserlocationdefs}%
9634 \write\glswrite{^^J; define cross-reference class^^J}%
9635 \write\glswrite{(define-crossref-class \string"see\string"
9636     :unverified )}%

```



```

9637 \write\glswrite{(markup-crossref-list
9638 :class \string"see\string"^^J\space\space\space
9639 :open \string"\string\glseeformat\string"
9640 :close \string"{}\string")}%
9641 \write\glswrite{^^J; define the order of the location classes}%
9642 \write\glswrite{(define-location-class-order
9643 (\@xdylocationclassorder))}%
9644 \write\glswrite{^^J; define the glossary markup^^J}%
9645 \write\glswrite{(markup-index^^J\space\space\space
9646 :open \string"\string
9647 \glossarysection[\string\glossarytoctitle]{\string
9648 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9649 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9650 \space\space:close \string"\expandafter\@gobble
9651 \string%\string~n\string
9652 \end{theglossary}\string\glossarypostamble
9653 \string~n\string" ^^J\space\space\space
9654 :tree)}}%
9655 \write\glswrite{(markup-letter-group-list
9656 :sep \string"\string\glsgroupskip\string~n\string")}%
9657 \write\glswrite{(markup-indexentry
9658 :open \string"\string\relax \string\glresetentrylist
9659 \string~n\string")}%
9660 \write\glswrite{(markup-locclass-list :open
9661 \string"\glsoopenbrace\string\glossaryentrynumbers
9662 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9663 :sep \string", \string"
9664 :close \string"\glsclosebrace\glsclosebrace\string")}%
9665 \write\glswrite{(markup-locref-list
9666 :sep \string"\string\delimN\space\string")}%
9667 \write\glswrite{(markup-range
9668 :sep \string"\string\delimR\space\string")}%
9669 \@onelevel@sanitize\gls@suffixF
9670 \@onelevel@sanitize\gls@suffixFF
9671 \ifx\gls@suffixF\@empty
9672 \else
9673 \write\glswrite{(markup-range
9674 :close "\gls@suffixF" :length 1 :ignore-end)}%
9675 \fi
9676 \ifx\gls@suffixFF\@empty
9677 \else
9678 \write\glswrite{(markup-range
9679 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9680 \fi
9681 \write\glswrite{^^J; define format to use for locations^^J}%
9682 \write\glswrite{\@xdylocref}%
9683 \write\glswrite{^^J; define letter group list format^^J}%
9684 \write\glswrite{(markup-letter-group-list
9685 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9686 \write\glswrite{^^J; letter group headings^^J}%
9687 \write\glswrite{(markup-letter-group
9688   :open-head \string"\string\glsgroupheading
9689   \glsoopenbrace\string"^^J\space\space\space
9690   :close-head \string"\glsclosebrace\string")}%
9691 \write\glswrite{^^J; additional letter groups^^J}%
9692 \write\glswrite{\@xdylettergroups}%
9693 \write\glswrite{^^J; additional sort rules^^J}
9694 \write\glswrite{\@xdysortrules}%
9695 \noist}
9696 \else
9697 \edef\@gls@actualchar{\string?}
9698 \edef\@gls@encapchar{\string|}
9699 \edef\@gls@levelchar{\string!}
9700 \edef\@gls@quotechar{\string"}
9701 \def\writeist{\relax
9702   \openout\glswrite=\istfilename
9703   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9704     created by the glossaries package}
9705   \write\glswrite{\expandafter\@gobble\string\% for document
9706     '\jobname' on \the\year-\the\month-\the\day}
9707   \write\glswrite{actual '\@gls@actualchar'}
9708   \write\glswrite{encap '\@gls@encapchar'}
9709   \write\glswrite{level '\@gls@levelchar'}
9710   \write\glswrite{quote '\@gls@quotechar'}
9711   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9712   \write\glswrite{preamble \string"\string\glossarysection[\string
9713     \glossarytoctitle]{\string\glossarytitle}\string
9714     \glossarypreamble\string\n\string\begin{theglossary}\string
9715     \glossaryheader\string\n\string"}
9716   \write\glswrite{postamble \string"\string%\string\n\string
9717     \end{theglossary}\string\glossarypostamble\string\n
9718     \string"}
9719   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9720     \string"}
9721   \write\glswrite{item_0 \string"\string%\string\n\string"}
9722   \write\glswrite{item_1 \string"\string%\string\n\string"}
9723   \write\glswrite{item_2 \string"\string%\string\n\string"}
9724   \write\glswrite{item_01 \string"\string%\string\n\string"}
9725   \write\glswrite{item_x1
9726     \string"\string\relax \string\glsresetentrylist\string\n
9727     \string"}
9728   \write\glswrite{item_12 \string"\string%\string\n\string"}
9729   \write\glswrite{item_x2
9730     \string"\string\relax \string\glsresetentrylist\string\n
9731     \string"}
9732   \write\glswrite{delim_0 \string"\string\{\string
9733     \glossaryentrynumbers\string\{\string\relax \string"}
9734   \write\glswrite{delim_1 \string"\string\{\string

```

```

9735     \glossaryentrynumbers\string\{\string\relax \string}
9736 \write\glswrite{delim_2 \string"\string\{\string
9737     \glossaryentrynumbers\string\{\string\relax \string}
9738 \write\glswrite{delim_t \string"\string\}\string\}\string}
9739 \write\glswrite{delim_n \string"\string\delimN \string}
9740 \write\glswrite{delim_r \string"\string\delimR \string}
9741 \write\glswrite{headings_flag 1}
9742 \write\glswrite{heading_prefix
9743     \string"\string\glsgroupheading\string\{\string}
9744 \write\glswrite{heading_suffix
9745     \string"\string\}\string\relax
9746     \string\glsgroupresetentrylist \string}
9747 \write\glswrite{symhead_positive \string"glssymbols\string}
9748 \write\glswrite{numhead_positive \string"glslnumbers\string}
9749 \write\glswrite{page_compositor \string"glscpositor\string}
9750 \@gls@escbsdq\gls@suffixF
9751 \@gls@escbsdq\gls@suffixFF
9752 \ifx\gls@suffixF\@empty
9753 \else
9754     \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9755 \fi
9756 \ifx\gls@suffixFF\@empty
9757 \else
9758     \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9759 \fi
9760 \noist
9761 }
9762 \fi

```

\noist

```

9763 \renewcommand*{\noist}{\let\writeist\relax}

```

## 4.2 glossaries-compatible-307

```

9764 \NeedsTeXFormat{LaTeX2e}
9765 \ProvidesPackage{glossaries-compatible-307}[2017/08/24 v4.32 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9766 \newcommand{\compatglossarystyle}[2]{%
9767   \ifcsundef{@glscompstyle@#1}%
9768   {%
9769     \csdef{@glscompstyle@#1}{#2}%
9770   }%
9771   {%
9772     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9773   }%
9774 }

```

Backward compatible inline style.

```

9775 \compatglossarystyle{inline}{%
9776   \renewcommand{\glossaryentryfield}[5]{%
9777     \glsinlinedopostchild
9778     \gls@inlinesep
9779     \def\glo@desc{##3}%
9780     \def\@no@post@desc{\nopostdesc}%
9781     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
9782     \ifx\glo@desc\@no@post@desc
9783       \glsinlineemptydescformat{##4}{##5}%
9784     \else
9785       \ifstrepty{##3}%
9786         {\glsinlineemptydescformat{##4}{##5}}%
9787         {\glsinlinedescformat{##3}{##4}{##5}}%
9788     \fi
9789     \ifglshaschildren{##1}%
9790     {%
9791       \glsresetsubentrycounter
9792       \glsinlineparentchildseparator
9793       \def\gls@inlinesubsep{}%
9794       \def\gls@inlinepostchild{\glsinlinepostchild}%
9795     }%
9796   }%
9797   \def\gls@inlinesep{\glsinlineseparator}%
9798 }%
```

Sub-entries display description:

```

9799 \renewcommand{\glossarysubentryfield}[6]{%
9800   \gls@inlinesubsep%
9801   \glsinlinesubnameformat{##2}{##3}%
9802   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9803   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9804 }%
9805 }
```

Backward compatible list style.

```

9806 \compatglossarystyle{list}{%
9807   \renewcommand*{\glossaryentryfield}[5]{%
9808     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
9809     ##3\glspostdescription\space ##5}%
9810 }
```

Sub-entries continue on the same line:

```

9810 \renewcommand*{\glossarysubentryfield}[6]{%
9811   \glssubentryitem{##2}%
9812   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9813 }
```

Backward compatible listgroup style.

```

9814 \compatglossarystyle{listgroup}{%
9815   \csuse{@glscompstyle@list}%
9816 }%
```

Backward compatible listhypergroup style.

```
9817 \compatglossarystyle{listhypergroup}{%
9818   \csuse{@glscompstyle@list}%
9819 }%
```

Backward compatible altlist style.

```
9820 \compatglossarystyle{altlist}{%
9821   \renewcommand*{\glossaryentryfield}[5]{%
9822     \item[\glstentryitem{##1}\glstarget{##1}{##2}]%
9823     \mbox{}\par\nobreak\@afterheading
9824     ##3\glspostdescription\space ##5}%
9825   \renewcommand{\glossarysubentryfield}[6]{%
9826     \par
9827     \glssubentryitem{##2}%
9828     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9829 }%
```

Backward compatible altlistgroup style.

```
9830 \compatglossarystyle{altlistgroup}{%
9831   \csuse{@glscompstyle@altlist}%
9832 }%
```

Backward compatible altlisthypergroup style.

```
9833 \compatglossarystyle{altlisthypergroup}{%
9834   \csuse{@glscompstyle@altlist}%
9835 }%
```

Backward compatible listdotted style.

```
9836 \compatglossarystyle{listdotted}{%
9837   \renewcommand*{\glossaryentryfield}[5]{%
9838     \item[\makebox[\glslistdottedwidth][l]{%
9839       \glstentryitem{##1}\glstarget{##1}{##2}%
9840       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9841   \renewcommand*{\glossarysubentryfield}[6]{%
9842     \item[\makebox[\glslistdottedwidth][l]{%
9843       \glssubentryitem{##2}%
9844       \glstarget{##2}{##3}%
9845       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9846 }%
```

Backward compatible sublistdotted style.

```
9847 \compatglossarystyle{sublistdotted}{%
9848   \csuse{@glscompstyle@listdotted}%
9849   \renewcommand*{\glossaryentryfield}[5]{%
9850     \item[\glstentryitem{##1}\glstarget{##1}{##2}]}%
9851 }%
```

Backward compatible long style.

```
9852 \compatglossarystyle{long}{%
9853   \renewcommand*{\glossaryentryfield}[5]{%
9854     \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9855   \renewcommand*{\glossarysubentryfield}[6]{%
9856     \glssubentryitem{##2}\glstarget{##2}{##3} & ##4\glspostdescription\space ##5\\}%
9857 }%
```

```

9856      &
9857      \glssubentryitem{##2}%
9858      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9859 }%

```

Backward compatible longborder style.

```

9860 \compatglossarystyle{longborder}{%
9861   \csuse{@glscmpstyle@long}%
9862 }%

```

Backward compatible longheader style.

```

9863 \compatglossarystyle{longheader}{%
9864   \csuse{@glscmpstyle@long}%
9865 }%

```

Backward compatible longheaderborder style.

```

9866 \compatglossarystyle{longheaderborder}{%
9867   \csuse{@glscmpstyle@long}%
9868 }%

```

Backward compatible long3col style.

```

9869 \compatglossarystyle{long3col}{%
9870   \renewcommand*{\glossaryentryfield}[5]{%
9871     \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##5\\}%
9872     \renewcommand*{\glossarysubentryfield}[6]{%
9873       &
9874       \glssubentryitem{##2}%
9875       \glstarget{##2}{\strut}##4 & ##6\\}%
9876 }%

```

Backward compatible long3colborder style.

```

9877 \compatglossarystyle{long3colborder}{%
9878   \csuse{@glscmpstyle@long3col}%
9879 }%

```

Backward compatible long3colheader style.

```

9880 \compatglossarystyle{long3colheader}{%
9881   \csuse{@glscmpstyle@long3col}%
9882 }%

```

Backward compatible long3colheaderborder style.

```

9883 \compatglossarystyle{long3colheaderborder}{%
9884   \csuse{@glscmpstyle@long3col}%
9885 }%

```

Backward compatible long4col style.

```

9886 \compatglossarystyle{long4col}{%
9887   \renewcommand*{\glossaryentryfield}[5]{%
9888     \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9889     \renewcommand*{\glossarysubentryfield}[6]{%
9890       &
9891       \glssubentryitem{##2}%

```

9892 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%  
 9893 }%

Backward compatible long4colheader style.

9894 \compatglossarystyle{long4colheader}{%  
 9895 \csuse{@glscompstyle@long4col}%  
 9896 }%

Backward compatible long4colborder style.

9897 \compatglossarystyle{long4colborder}{%  
 9898 \csuse{@glscompstyle@long4col}%  
 9899 }%

Backward compatible long4colheaderborder style.

9900 \compatglossarystyle{long4colheaderborder}{%  
 9901 \csuse{@glscompstyle@long4col}%  
 9902 }%

Backward compatible altlong4col style.

9903 \compatglossarystyle{altlong4col}{%  
 9904 \csuse{@glscompstyle@long4col}%  
 9905 }%

Backward compatible altlong4colheader style.

9906 \compatglossarystyle{altlong4colheader}{%  
 9907 \csuse{@glscompstyle@long4col}%  
 9908 }%

Backward compatible altlong4colborder style.

9909 \compatglossarystyle{altlong4colborder}{%  
 9910 \csuse{@glscompstyle@long4col}%  
 9911 }%

Backward compatible altlong4colheaderborder style.

9912 \compatglossarystyle{altlong4colheaderborder}{%  
 9913 \csuse{@glscompstyle@long4col}%  
 9914 }%

Backward compatible long style.

9915 \compatglossarystyle{longragged}{%  
 9916 \renewcommand\*{\glossaryentryfield}[5]{%  
 9917 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%  
 9918 \tabularnewline}%  
 9919 \renewcommand\*{\glossarysubentryfield}[6]{%  
 9920 &  
 9921 \glssubentryitem{##2}%  
 9922 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%  
 9923 \tabularnewline}%  
 9924 }%

Backward compatible longraggedborder style.

9925 \compatglossarystyle{longraggedborder}{%  
 9926 \csuse{@glscompstyle@longragged}%  
 9927 }%

Backward compatible longraggedheader style.

```
9928 \compatglossarystyle{longraggedheader}{%  
9929 \csuse{@glscompstyle@longragged}%  
9930 }%
```

Backward compatible longraggedheaderborder style.

```
9931 \compatglossarystyle{longraggedheaderborder}{%  
9932 \csuse{@glscompstyle@longragged}%  
9933 }%
```

Backward compatible longragged3col style.

```
9934 \compatglossarystyle{longragged3col}{%  
9935 \renewcommand*{\glossaryentryfield}[5]{%  
9936 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9937 \renewcommand*{\glossarysubentryfield}[6]{%  
9938 &  
9939 \glssubentryitem{##2}%  
9940 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9941 }%
```

Backward compatible longragged3colborder style.

```
9942 \compatglossarystyle{longragged3colborder}{%  
9943 \csuse{@glscompstyle@longragged3col}%  
9944 }%
```

Backward compatible longragged3colheader style.

```
9945 \compatglossarystyle{longragged3colheader}{%  
9946 \csuse{@glscompstyle@longragged3col}%  
9947 }%
```

Backward compatible longragged3colheaderborder style.

```
9948 \compatglossarystyle{longragged3colheaderborder}{%  
9949 \csuse{@glscompstyle@longragged3col}%  
9950 }%
```

Backward compatible altlongragged4col style.

```
9951 \compatglossarystyle{altlongragged4col}{%  
9952 \renewcommand*{\glossaryentryfield}[5]{%  
9953 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9954 \renewcommand*{\glossarysubentryfield}[6]{%  
9955 &  
9956 \glssubentryitem{##2}%  
9957 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9958 }%
```

Backward compatible altlongragged4colheader style.

```
9959 \compatglossarystyle{altlongragged4colheader}{%  
9960 \csuse{@glscompstyle@altlong4col}%  
9961 }%
```

Backward compatible altlongragged4colborder style.

```
9962 \compatglossarystyle{altlongragged4colborder}{%
```



```

9963 \csuse{@glscompstyle@altlong4col}%
9964 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9965 \compatglossarystyle{altlongragged4colheaderborder}{%
9966 \csuse{@glscompstyle@altlong4col}%
9967 }%

```

Backward compatible index style.

```

9968 \compatglossarystyle{index}{%
9969 \renewcommand*{\glossaryentryfield}[5]{%
9970 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9971 \ifx\relax##4\relax
9972 \else
9973 \space{##4}%
9974 \fi
9975 \space ##3\glspostdescription \space ##5}%
9976 \renewcommand*{\glossarysubentryfield}[6]{%
9977 \ifcase##1\relax
9978 % level 0
9979 \item
9980 \or
9981 % level 1
9982 \subitem
9983 \glssubentryitem{##2}%
9984 \else
9985 % all other levels
9986 \subsubitem
9987 \fi
9988 \textbf{\glstarget{##2}{##3}}%
9989 \ifx\relax##5\relax
9990 \else
9991 \space{##5}%
9992 \fi
9993 \space##4\glspostdescription\space ##6}%
9994 }%

```

Backward compatible indexgroup style.

```

9995 \compatglossarystyle{indexgroup}{%
9996 \csuse{@glscompstyle@index}%
9997 }%

```

Backward compatible indexhypergroup style.

```

9998 \compatglossarystyle{indexhypergroup}{%
9999 \csuse{@glscompstyle@index}%
10000 }%

```

Backward compatible tree style.

```

10001 \compatglossarystyle{tree}{%
10002 \renewcommand{\glossaryentryfield}[5]{%
10003 \hangindent0pt\relax

```

```

10004 \parindent0pt\relax
10005 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10006 \ifx\relax##4\relax
10007 \else
10008 \space(##4)%
10009 \fi
10010 \space ##3\glspostdescription \space ##5\par}%
10011 \renewcommand{\glossarysubentryfield}[6]{%
10012 \hangindent##1\glstreeindent\relax
10013 \parindent##1\glstreeindent\relax
10014 \ifnum##1=1\relax
10015 \glssubentryitem{##2}%
10016 \fi
10017 \textbf{\glstarget{##2}{##3}}%
10018 \ifx\relax##5\relax
10019 \else
10020 \space(##5)%
10021 \fi
10022 \space##4\glspostdescription\space ##6\par}%
10023 }%

```

Backward compatible treegroup style.

```

10024 \compatglossarystyle{treegroup}{%
10025 \csuse{@glscmpstyle@tree}%
10026 }%

```

Backward compatible treehypergroup style.

```

10027 \compatglossarystyle{treehypergroup}{%
10028 \csuse{@glscmpstyle@tree}%
10029 }%

```

Backward compatible treenoname style.

```

10030 \compatglossarystyle{treenoname}{%
10031 \renewcommand{\glossaryentryfield}[5]{%
10032 \hangindent0pt\relax
10033 \parindent0pt\relax
10034 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10035 \ifx\relax##4\relax
10036 \else
10037 \space(##4)%
10038 \fi
10039 \space ##3\glspostdescription \space ##5\par}%
10040 \renewcommand{\glossarysubentryfield}[6]{%
10041 \hangindent##1\glstreeindent\relax
10042 \parindent##1\glstreeindent\relax
10043 \ifnum##1=1\relax
10044 \glssubentryitem{##2}%
10045 \fi
10046 \glstarget{##2}{\strut}%
10047 ##4\glspostdescription\space ##6\par}%
10048 }%

```

Backward compatible treenonamegroup style.

```
10049 \compatglossarystyle{treenonamegroup}{%
10050   \csuse{@glscompstyle@treenoname}%
10051 }%
```

Backward compatible treenonamehypergroup style.

```
10052 \compatglossarystyle{treenonamehypergroup}{%
10053   \csuse{@glscompstyle@treenoname}%
10054 }%
```

Backward compatible alttree style.

```
10055 \compatglossarystyle{almtree}{%
10056   \renewcommand{\glossaryentryfield}[5]{%
10057     \ifnum\@gls@prevlevel=0\relax
10058     \else
10059       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10060       \hangindent\glstreeindent
10061       \parindent\glstreeindent
10062     \fi
10063     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10064       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10065     \ifx\relax##4\relax
10066     \else
10067       (##4)\space
10068     \fi
10069     ##3\glspostdescription \space ##5\par
10070     \def\@gls@prevlevel{0}%
10071   }%
10072   \renewcommand{\glossarysubentryfield}[6]{%
10073     \ifnum##1=1\relax
10074       \glssubentryitem{##2}%
10075     \fi
10076     \ifnum\@gls@prevlevel=##1\relax
10077     \else
10078       \@ifundefined{@glswidestname\romannumeral##1}{%
10079         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10080         \settowidth{\gls@tmplen}{\textbf{%
10081           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10082         \ifnum\@gls@prevlevel<##1\relax
10083           \setlength\glstreeindent\gls@tmplen
10084           \addtolength\glstreeindent\parindent
10085           \parindent\glstreeindent
10086         \else
10087           \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10088             \settowidth{\glstreeindent}{\textbf{%
10089               \@glswidestname\space}}{%
10090             \settowidth{\glstreeindent}{\textbf{%
10091               \csname @glswidestname\romannumeral\@gls@prevlevel
10092               \endcsname\space}}}%
10093             \addtolength\parindent{-\glstreeindent}%

```

```

10094      \setlength\glstreeindent\parindent
10095      \fi
10096      \fi
10097      \hangindent\glstreeindent
10098      \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10099        \textbf{\glstarget{##2}{##3}}}%
10100      \ifx##5\relax\relax
10101      \else
10102        (##5)\space
10103      \fi
10104      ##4\glspostdescription\space ##6\par
10105      \def\@gls@prevlevel{##1}%
10106    }%
10107 }%

```

Backward compatible alttreegroup style.

```

10108 \compatglossarystyle{alttreegroup}{%
10109   \csuse{@glscompstyle@alttree}%
10110 }%

```

Backward compatible alttreehypergroup style.

```

10111 \compatglossarystyle{alttreehypergroup}{%
10112   \csuse{@glscompstyle@alttree}%
10113 }%

```

Backward compatible mcolindex style.

```

10114 \compatglossarystyle{mcolindex}{%
10115   \csuse{@glscompstyle@index}%
10116 }%

```

Backward compatible mcolindexgroup style.

```

10117 \compatglossarystyle{mcolindexgroup}{%
10118   \csuse{@glscompstyle@index}%
10119 }%

```

Backward compatible mcolindexhypergroup style.

```

10120 \compatglossarystyle{mcolindexhypergroup}{%
10121   \csuse{@glscompstyle@index}%
10122 }%

```

Backward compatible mcoltree style.

```

10123 \compatglossarystyle{mcoltree}{%
10124   \csuse{@glscompstyle@tree}%
10125 }%

```

Backward compatible mcoltreegroup style.

```

10126 \compatglossarystyle{mcolindextreegroup}{%
10127   \csuse{@glscompstyle@tree}%
10128 }%

```

Backward compatible mcoltreehypergroup style.

```

10129 \compatglossarystyle{mcolindextreehypergroup}{%

```

```

10130 \csuse{@glscompstyle@tree}%
10131 }%

    Backward compatible mcoltreenoname style.
10132 \compatglossarystyle{mcoltreenoname}{%
10133 \csuse{@glscompstyle@tree}%
10134 }%

    Backward compatible mcoltreenonamegroup style.
10135 \compatglossarystyle{mcoltreenonamegroup}{%
10136 \csuse{@glscompstyle@tree}%
10137 }%

    Backward compatible mcoltreenonamehypergroup style.
10138 \compatglossarystyle{mcoltreenonamehypergroup}{%
10139 \csuse{@glscompstyle@tree}%
10140 }%

    Backward compatible mcolalmtree style.
10141 \compatglossarystyle{mcolalmtree}{%
10142 \csuse{@glscompstyle@almtree}%
10143 }%

    Backward compatible mcolalmtreegroup style.
10144 \compatglossarystyle{mcolalmtreegroup}{%
10145 \csuse{@glscompstyle@almtree}%
10146 }%

    Backward compatible mcolalmtreehypergroup style.
10147 \compatglossarystyle{mcolalmtreehypergroup}{%
10148 \csuse{@glscompstyle@almtree}%
10149 }%

    Backward compatible superragged style.
10150 \compatglossarystyle{superragged}{%
10151 \renewcommand*{\glossaryentryfield}[5]{%
10152 \glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10153 \tabularnewline}%
10154 \renewcommand*{\glossarysubentryfield}[6]{%
10155 &
10156 \glssubentryitem{##2}%
10157 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10158 \tabularnewline}%
10159 }%

    Backward compatible superraggedborder style.
10160 \compatglossarystyle{superraggedborder}{%
10161 \csuse{@glscompstyle@superragged}%
10162 }%

    Backward compatible superraggedheader style.
10163 \compatglossarystyle{superraggedheader}{%
10164 \csuse{@glscompstyle@superragged}%
10165 }%

```

Backward compatible superraggedheaderborder style.

```
10166 \compatglossarystyle{superraggedheaderborder}{%
10167   \csuse{@glscompstyle@superragged}%
10168 }%
```

Backward compatible superragged3col style.

```
10169 \compatglossarystyle{superragged3col}{%
10170   \renewcommand*{\glossaryentryfield}[5]{%
10171     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10172   \renewcommand*{\glossarysubentryfield}[6]{%
10173     &
10174     \glssubentryitem{##2}%
10175     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10176 }%
```

Backward compatible superragged3colborder style.

```
10177 \compatglossarystyle{superragged3colborder}{%
10178   \csuse{@glscompstyle@superragged3col}%
10179 }%
```

Backward compatible superragged3colheader style.

```
10180 \compatglossarystyle{superragged3colheader}{%
10181   \csuse{@glscompstyle@superragged3col}%
10182 }%
```

Backward compatible superragged3colheaderborder style.

```
10183 \compatglossarystyle{superragged3colheaderborder}{%
10184   \csuse{@glscompstyle@superragged3col}%
10185 }%
```

Backward compatible altsuperragged4col style.

```
10186 \compatglossarystyle{altsuperragged4col}{%
10187   \renewcommand*{\glossaryentryfield}[5]{%
10188     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10189   \renewcommand*{\glossarysubentryfield}[6]{%
10190     &
10191     \glssubentryitem{##2}%
10192     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10193 }%
```

Backward compatible altsuperragged4colheader style.

```
10194 \compatglossarystyle{altsuperragged4colheader}{%
10195   \csuse{@glscompstyle@altsuperragged4col}%
10196 }%
```

Backward compatible altsuperragged4colborder style.

```
10197 \compatglossarystyle{altsuperragged4colborder}{%
10198   \csuse{@glscompstyle@altsuperragged4col}%
10199 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10200 \compatglossarystyle{altsuperragged4colheaderborder}{%

```

```
10201 \csuse{@glscompstyle@altsuperragged4col}%
10202 }%
```

Backward compatible super style.

```
10203 \compatglossarystyle{super}{%
10204   \renewcommand*{\glossaryentryfield}[5]{%
10205     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10206   \renewcommand*{\glossarysubentryfield}[6]{%
10207     &
10208     \glssubentryitem{##2}%
10209     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10210 }%
```

Backward compatible superborder style.

```
10211 \compatglossarystyle{superborder}{%
10212   \csuse{@glscompstyle@super}%
10213 }%
```

Backward compatible superheader style.

```
10214 \compatglossarystyle{superheader}{%
10215   \csuse{@glscompstyle@super}%
10216 }%
```

Backward compatible superheaderborder style.

```
10217 \compatglossarystyle{superheaderborder}{%
10218   \csuse{@glscompstyle@super}%
10219 }%
```

Backward compatible super3col style.

```
10220 \compatglossarystyle{super3col}{%
10221   \renewcommand*{\glossaryentryfield}[5]{%
10222     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10223   \renewcommand*{\glossarysubentryfield}[6]{%
10224     &
10225     \glssubentryitem{##2}%
10226     \glstarget{##2}{\strut}##4 & ##6\\}%
10227 }%
```

Backward compatible super3colborder style.

```
10228 \compatglossarystyle{super3colborder}{%
10229   \csuse{@glscompstyle@super3col}%
10230 }%
```

Backward compatible super3colheader style.

```
10231 \compatglossarystyle{super3colheader}{%
10232   \csuse{@glscompstyle@super3col}%
10233 }%
```

Backward compatible super3colheaderborder style.

```
10234 \compatglossarystyle{super3colheaderborder}{%
10235   \csuse{@glscompstyle@super3col}%
10236 }%
```

Backward compatible super4col style.

```
10237 \compatglossarystyle{super4col}{%
10238   \renewcommand*{\glossaryentryfield}[5]{%
10239     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10240   \renewcommand*{\glossarysubentryfield}[6]{%
10241     &
10242     \glssubentryitem{##2}%
10243     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10244 }%
```

Backward compatible super4colheader style.

```
10245 \compatglossarystyle{super4colheader}{%
10246   \csuse{@glscompstyle@super4col}%
10247 }%
```

Backward compatible super4colborder style.

```
10248 \compatglossarystyle{super4colborder}{%
10249   \csuse{@glscompstyle@super4col}%
10250 }%
```

Backward compatible super4colheaderborder style.

```
10251 \compatglossarystyle{super4colheaderborder}{%
10252   \csuse{@glscompstyle@super4col}%
10253 }%
```

Backward compatible altsuper4col style.

```
10254 \compatglossarystyle{altsuper4col}{%
10255   \csuse{@glscompstyle@super4col}%
10256 }%
```

Backward compatible altsuper4colheader style.

```
10257 \compatglossarystyle{altsuper4colheader}{%
10258   \csuse{@glscompstyle@super4col}%
10259 }%
```

Backward compatible altsuper4colborder style.

```
10260 \compatglossarystyle{altsuper4colborder}{%
10261   \csuse{@glscompstyle@super4col}%
10262 }%
```

Backward compatible altsuper4colheaderborder style.

```
10263 \compatglossarystyle{altsuper4colheaderborder}{%
10264   \csuse{@glscompstyle@super4col}%
10265 }%
```



## 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.

```
10266 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10267 \ProvidesPackage{glossaries-accsupp}[2017/08/24 v4.32 (NLCT)]
```

```
10268 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10269 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10270 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10271 \@ifpackageloaded{glossaries-extra}
```

```
10272 {%
```

If the accsupp option was used, \glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10273 \ifx\glsxtr@doaccsupp\empty
```

```
10274 \GlossariesWarning{The 'glossaries-accsupp'
```

```
10275 package has been loaded\MessageBreak
```

```
10276 after the 'glossaries-extra' package. This\MessageBreak
```

```
10277 can cause a failure to integrate both packages. \MessageBreak
```

```
10278 Either use the 'accsupp' option when you load\MessageBreak
```

```
10279 'glossaries-extra' or load 'glossaries-accsupp'\MessageBreak
```

```
10280 before loading 'glossaries-extra'}%
```

```
10281 \fi
```

```
10282 }
```

```
10283 {}
```

tibleglossentry Override style compatibility macros:

```
10284 \def\compatibleglossentry#1#2{%
```

```
10285 \toks0{#2}%
```

```
10286 \protected@edef\do@glossentry{%
```

```
10287 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10288 {\noexpand\glsnamefont
```

```
10289 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```

10290    {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10291    {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10292    {\the\toks@}%
10293    }%
10294    \@do@glossentry
10295 }

```

lesubglossentry

```

10296 \def\compatiblesubglossentry#1#2#3{%
10297   \toks@{#3}%
10298   \protected@edef\@do@subglossentry{%
10299     \noexpand\accsuppglossarysubentryfield{\number#1}%
10300     {#2}%
10301     {\noexpand\glsnamefont
10302      {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}}%
10303     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10304     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10305     {\the\toks@}%
10306   }%
10307   \@do@subglossentry
10308 }

```

Required packages:

```

10309 \RequirePackage{glossaries}
10310 \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:

```

10311 \define@key{glossentry}{access}{%
10312   \def\@glo@access{#1}%
10313 }

```

**textaccess** The replacement text corresponding to the text key:

```

10314 \define@key{glossentry}{textaccess}{%
10315   \def\@glo@textaccess{#1}%
10316 }

```

**firstaccess** The replacement text corresponding to the first key:

```

10317 \define@key{glossentry}{firstaccess}{%
10318   \def\@glo@firstaccess{#1}%
10319 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10320 \define@key{glossentry}{pluralaccess}{%
10321   \def\@glo@pluralaccess{#1}%
10322 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

10323 \define@key{glossentry}{firstpluralaccess}{%
10324   \def\@glo@firstpluralaccess{#1}%
10325 }
```

symbolaccess The replacement text corresponding to the symbol key:

```

10326 \define@key{glossentry}{symbolaccess}{%
10327   \def\@glo@symbolaccess{#1}%
10328 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```

10329 \define@key{glossentry}{symbolpluralaccess}{%
10330   \def\@glo@symbolpluralaccess{#1}%
10331 }
```

descriptionaccess The replacement text corresponding to the description key:

```

10332 \define@key{glossentry}{descriptionaccess}{%
10333   \def\@glo@descaccess{#1}%
10334 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```

10335 \define@key{glossentry}{descriptionpluralaccess}{%
10336   \def\@glo@descpluralaccess{#1}%
10337 }
```

shortaccess The replacement text corresponding to the short key:

```

10338 \define@key{glossentry}{shortaccess}{%
10339   \def\@glo@shortaccess{#1}%
10340 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```

10341 \define@key{glossentry}{shortpluralaccess}{%
10342   \def\@glo@shortpluralaccess{#1}%
10343 }
```

longaccess The replacement text corresponding to the long key:

```

10344 \define@key{glossentry}{longaccess}{%
10345   \def\@glo@longaccess{#1}%
10346 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```

10347 \define@key{glossentry}{longpluralaccess}{%
10348   \def\@glo@longpluralaccess{#1}%
10349 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsacccsup{inches}{in}}.

Append these new keys to \@gls@keymap:

```
10350 \appto\@gls@keymap{,%
10351   {access}{access},%
10352   {textaccess}{textaccess},%
10353   {firstaccess}{firstaccess},%
10354   {pluralaccess}{pluralaccess},%
10355   {firstpluralaccess}{firstpluralaccess},%
10356   {symbolaccess}{symbolaccess},%
10357   {symbolpluralaccess}{symbolpluralaccess},%
10358   {descaccess}{descaccess},%
10359   {descpluralaccess}{descpluralaccess},%
10360   {shortaccess}{shortaccess},%
10361   {shortpluralaccess}{shortpluralaccess},%
10362   {longaccess}{longaccess},%
10363   {longpluralaccess}{longpluralaccess}%
10364 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
10365 \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):

```
10366 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10367 \renewcommand*{\@newglossaryentryprehook}{%
10368   \@gls@oldnewglossaryentryprehook
10369   \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```
10370 \def\@glo@textaccess{\@glo@access}%
10371 \def\@glo@firstaccess{\@glo@access}%
10372 \def\@glo@pluralaccess{\@glo@textaccess}%
10373 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10374 \def\@glo@symbolaccess{\relax}%
10375 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10376 \def\@glo@descaccess{\relax}%
10377 \def\@glo@descpluralaccess{\@glo@descaccess}%
10378 \def\@glo@shortaccess{\relax}%
10379 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10380 \def\@glo@longaccess{\relax}%
10381 \def\@glo@longpluralaccess{\@glo@longaccess}%
10382 }
```

Add to the end hook:

```
10383 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10384 \renewcommand*{\@newglossaryentryposthook}{%
10385   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```

10386 \expandafter
10387 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10388 \@glo@access}%
10389 \expandafter
10390 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10391 \@glo@textaccess}%
10392 \expandafter
10393 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10394 \@glo@firstaccess}%
10395 \expandafter
10396 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10397 \@glo@pluralaccess}%
10398 \expandafter
10399 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10400 \@glo@firstpluralaccess}%
10401 \expandafter
10402 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10403 \@glo@symbolaccess}%
10404 \expandafter
10405 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10406 \@glo@symbolpluralaccess}%
10407 \expandafter
10408 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10409 \@glo@descaccess}%
10410 \expandafter
10411 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10412 \@glo@descpluralaccess}%
10413 \expandafter
10414 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10415 \@glo@shortaccess}%
10416 \expandafter
10417 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10418 \@glo@shortpluralaccess}%
10419 \expandafter
10420 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10421 \@glo@longaccess}%
10422 \expandafter
10423 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10424 \@glo@longpluralaccess}%
10425 }

```

## 5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10426 \newcommand*{\glsentryaccess}[1]{%
10427 \@gls@entry@field{#1}{access}%
10428 }

```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
10429 \newcommand*{\glentrytextaccess}[1]{%
10430   \@gls@entry@field{#1}{textaccess}%
10431 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10432 \newcommand*{\glentryfirstaccess}[1]{%
10433   \@gls@entry@field{#1}{firstaccess}%
10434 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10435 \newcommand*{\glentrypluralaccess}[1]{%
10436   \@gls@entry@field{#1}{pluralaccess}%
10437 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10438 \newcommand*{\glentryfirstpluralaccess}[1]{%
10439   \csname glo@#1@firstpluralaccess\endcsname
10440 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10441 \newcommand*{\glentrysymbolaccess}[1]{%
10442   \@gls@entry@field{#1}{symbolaccess}%
10443 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10444 \newcommand*{\glentrysymbolpluralaccess}[1]{%
10445   \@gls@entry@field{#1}{symbolpluralaccess}%
10446 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10447 \newcommand*{\glentrydescaccess}[1]{%
10448   \@gls@entry@field{#1}{descaccess}%
10449 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10450 \newcommand*{\glentrydescpluralaccess}[1]{%
10451   \@gls@entry@field{#1}{descaccess}%
10452 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10453 \newcommand*{\glentryshortaccess}[1]{%
10454   \@gls@entry@field{#1}{shortaccess}%
10455 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10456 \newcommand*{\glentryshortpluralaccess}[1]{%
10457   \@gls@entry@field{#1}{shortpluralaccess}%
10458 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10459 \newcommand*{\glsentrylongaccess}[1]{%
10460   \@gls@entry@field{#1}{longaccess}%
10461 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10462 \newcommand*{\glsentrylongpluralaccess}[1]{%
10463   \@gls@entry@field{#1}{longpluralaccess}%
10464 }
```

\glsacccsupp \glsacccsupp{<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.)

```
10465 \newcommand*{\glsacccsupp}[2]{%
10466   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10467 }
```

\xglsacccsupp Fully expands replacement text before calling \glsacccsupp

```
10468 \newcommand*{\xglsacccsupp}[2]{%
10469   \protected@edef\@gls@replacementtext{#1}%
10470   \expandafter\glsacccsupp\expandafter{\@gls@replacementtext}{#2}%
10471 }
```

@access@display

```
10472 \newcommand*{\@gls@access@display}[2]{%
10473   \protected@edef\@glo@access{#2}%
10474   \ifx\@glo@access\@gls@noaccess
10475     #1%
10476   \else
10477     \xglsacccsupp{\@glo@access}{#1}%
10478   \fi
10479 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10480 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10481   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10482 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10483 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10484   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10485 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10486 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10487   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10488 }
```

staccessdisplay As above but for the firstaccess replacement text.

```

10489 \DeclareRobustCommand*\glfirstaccessdisplay}[2]{%
10490   \@gls@access@display{#1}{\glentryfirstaccess{#2}}%
10491 }

```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```

10492 \DeclareRobustCommand*\glfirstpluralaccessdisplay}[2]{%
10493   \@gls@access@display{#1}{\glentryfirstpluralaccess{#2}}%
10494 }

```

olaccessdisplay As above but for the symbolaccess replacement text.

```

10495 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
10496   \@gls@access@display{#1}{\glentrysymbolaccess{#2}}%
10497 }

```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```

10498 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
10499   \@gls@access@display{#1}{\glentrysymbolpluralaccess{#2}}%
10500 }

```

onaccessdisplay As above but for the descriptionaccess replacement text.

```

10501 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10502   \@gls@access@display{#1}{\glentrydescaccess{#2}}%
10503 }

```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```

10504 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10505   \@gls@access@display{#1}{\glentrydescpluralaccess{#2}}%
10506 }

```

rtaccessdisplay As above but for the shortaccess replacement text.

```

10507 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10508   \@gls@access@display{#1}{\glentryshortaccess{#2}}%
10509 }

```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```

10510 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10511   \@gls@access@display{#1}{\glentryshortpluralaccess{#2}}%
10512 }

```

ngaccessdisplay As above but for the longaccess replacement text.

```

10513 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10514   \@gls@access@display{#1}{\glentrylongaccess{#2}}%
10515 }

```

alaccessdisplay As above but for the longpluralaccess replacement text.

```

10516 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10517   \@gls@access@display{#1}{\glentrylongpluralaccess{#2}}%
10518 }

```



`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

10519 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10520   \@ifundefined{gls#1accessdisplay}%
10521   {%
10522     \PackageError{glossaries-accsupp}{No accessibility support
10523       for key ‘#1’}{}%
10524   }%
10525   {%
10526     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10527   }%
10528 }

```

`\default@entryfmt` Redefine the default entry format to use accessibility information

```

10529 \renewcommand*\@@gls@default@entryfmt}[2]{%
10530   \ifdefempty\glscustomtext
10531   {%
10532     \glsifplural
10533     {%

```

Plural form

```

10534     \glscapscase
10535     {%

```

Don't adjust case

```

10536     \ifglsused\glslabel
10537     {%

```

Subsequent use

```

10538     #2{\glspluralaccessdisplay
10539       {\glsentryplural{\glslabel}}{\glslabel}}%
10540     {\glsdescriptionpluralaccessdisplay
10541       {\glsentrydescplural{\glslabel}}{\glslabel}}%
10542     {\glsymbolpluralaccessdisplay
10543       {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10544     {\glsinsert}%
10545   }%
10546   {%

```

First use

```

10547     #1{\glsfirstpluralaccessdisplay
10548       {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10549     {\glsdescriptionpluralaccessdisplay
10550       {\glsentrydescplural{\glslabel}}{\glslabel}}%
10551     {\glsymbolpluralaccessdisplay
10552       {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10553     {\glsinsert}%
10554   }%
10555   }%
10556   {%

```

Make first letter upper case

```
10557      \ifglused\glslabel
10558      {%
```

Subsequent use.

```
10559      #2{\glsppluralaccessdisplay
10560          {\Glentryplural{\glslabel}}{\glslabel}}%
10561          {\glsddescriptionpluralaccessdisplay
10562              {\glentrydescplural{\glslabel}}{\glslabel}}%
10563          {\glssymbolpluralaccessdisplay
10564              {\glentrysymbolplural{\glslabel}}{\glslabel}}%
10565          {\glinsert}%
10566      }%
10567      {%
```

First use

```
10568      #1{\glfirstpluralaccessdisplay
10569          {\Glentryfirstplural{\glslabel}}{\glslabel}}%
10570          {\glsddescriptionpluralaccessdisplay
10571              {\glentrydescplural{\glslabel}}{\glslabel}}%
10572          {\glssymbolpluralaccessdisplay
10573              {\glentrysymbolplural{\glslabel}}{\glslabel}}%
10574          {\glinsert}%
10575      }%
10576      }%
10577      {%
```

Make all upper case

```
10578      \ifglused\glslabel
10579      {%
```

Subsequent use

```
10580      \MakeUppercase{%
10581      #2{\glsppluralaccessdisplay
10582          {\glentryplural{\glslabel}}{\glslabel}}%
10583          {\glsddescriptionpluralaccessdisplay
10584              {\glentrydescplural{\glslabel}}{\glslabel}}%
10585          {\glssymbolpluralaccessdisplay
10586              {\glentrysymbolplural{\glslabel}}{\glslabel}}%
10587          {\glinsert}}%
10588      }%
10589      {%
```

First use

```
10590      \MakeUppercase{%
10591      #1{\glfirstpluralaccessdisplay
10592          {\glentryfirstplural{\glslabel}}{\glslabel}}%
10593          {\glsddescriptionpluralaccessdisplay
10594              {\glentrydescplural{\glslabel}}{\glslabel}}%
10595          {\glssymbolpluralaccessdisplay
10596              {\glentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10597         {\glsinsert}}}%
10598     }%
10599 }%
10600 }%
10601 {%

```

#### Singular form

```

10602     \glscapscase
10603     {%

```

#### Don't adjust case

```

10604     \ifglsused\glslabel
10605     {%

```

#### Subsequent use

```

10606     #2{\glstextaccessdisplay
10607         {\glsentrytext{\glslabel}}{\glslabel}}%
10608     {\glsdescriptionaccessdisplay
10609         {\glsentrydesc{\glslabel}}{\glslabel}}%
10610     {\glssymbolaccessdisplay
10611         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10612     {\glsinsert}}%
10613 }%
10614 {%

```

#### First use

```

10615     #1{\glsfirstaccessdisplay
10616         {\glsentryfirst{\glslabel}}{\glslabel}}%
10617     {\glsdescriptionaccessdisplay
10618         {\glsentrydesc{\glslabel}}{\glslabel}}%
10619     {\glssymbolaccessdisplay
10620         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10621     {\glsinsert}}%
10622 }%
10623 }%
10624 {%

```

#### Make first letter upper case

```

10625     \ifglsused\glslabel
10626     {%

```

#### Subsequent use

```

10627     #2{\glstextaccessdisplay
10628         {\Glsentrytext{\glslabel}}{\glslabel}}%
10629     {\glsdescriptionaccessdisplay
10630         {\glsentrydesc{\glslabel}}{\glslabel}}%
10631     {\glssymbolaccessdisplay
10632         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10633     {\glsinsert}}%
10634 }%
10635 {%

```

#### First use

```

10636      #1{\glsfirstaccessdisplay
10637          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10638          {\glsdescriptionaccessdisplay
10639              {\glsentrydesc{\glslabel}}{\glslabel}}%
10640          {\glsymbolaccessdisplay
10641              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10642          {\glsinsert}}%
10643      }%
10644  }%
10645  {%

```

#### Make all upper case

```

10646      \ifglsused\glslabel
10647      {%

```

#### Subsequent use

```

10648      \MakeUppercase{%
10649          #2{\glstextaccessdisplay
10650              {\glsentrytext{\glslabel}}{\glslabel}}%
10651              {\glsdescriptionaccessdisplay
10652                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10653                  {\glsymbolaccessdisplay
10654                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10655                      {\glsinsert}}}%
10656      }%
10657  {%

```

#### First use

```

10658      \MakeUppercase{%
10659          #1{\glsfirstaccessdisplay
10660              {\glsentryfirst{\glslabel}}{\glslabel}}%
10661              {\glsdescriptionaccessdisplay
10662                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10663                  {\glsymbolaccessdisplay
10664                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10665                      {\glsinsert}}}%
10666      }%
10667  }%
10668  }%
10669  }%
10670  {%

```

#### Custom text provided in \glsdisp

```

10671      \ifglsused{\glslabel}%
10672      {%

```

#### Subsequent use

```

10673      #2{\glscustomtext}%
10674      {\glsdescriptionaccessdisplay
10675          {\glsentrydesc{\glslabel}}{\glslabel}}%

```

```

10676      {\glssymbolaccessdisplay
10677      {\glentrysymbol{\glslabel}}{\glslabel}}%
10678      {\glsinsert}}%
10679  }%
10680  {%

```

First use

```

10681      #1{\glscustomtext}%
10682      {\glsdescriptionaccessdisplay
10683      {\glentrydesc{\glslabel}}{\glslabel}}%
10684      {\glssymbolaccessdisplay
10685      {\glentrysymbol{\glslabel}}{\glslabel}}%
10686      {\glsinsert}}%
10687  }%
10688  }%
10689 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10690 \renewcommand*{\glsgenentryfmt}{%
10691   \ifdefempty\glscustomtext
10692   {%
10693     \glsifplural
10694     {%

```

Plural form

```

10695     \glscapscase
10696     {%

```

Don't adjust case

```

10697     \ifglused\glslabel
10698     {%

```

Subsequent use

```

10699     \glspluralaccessdisplay
10700     {\glentryplural{\glslabel}}{\glslabel}%
10701     \glsinsert
10702   }%
10703   {%

```

First use

```

10704     \glsfirstpluralaccessdisplay
10705     {\glentryfirstplural{\glslabel}}{\glslabel}%
10706     \glsinsert
10707   }%
10708   }%
10709   {%

```

Make first letter upper case

```

10710     \ifglused\glslabel
10711     {%

```

Subsequent use.

```
10712      \glspluralaccessdisplay
10713      {\Glsentryplural{\glslabel}}{\glslabel}%
10714      \glsinsert
10715      }%
10716      {%
```

First use

```
10717      \glsfirstpluralaccessdisplay
10718      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10719      \glsinsert
10720      }%
10721      }%
10722      {%
```

Make all upper case

```
10723      \ifglused\glslabel
10724      {%
```

Subsequent use

```
10725      \glspluralaccessdisplay
10726      {\mfirstucMakeUppercase{\Glsentryplural{\glslabel}}}%
10727      {\glslabel}%
10728      \mfirstucMakeUppercase{\glsinsert}%
10729      }%
10730      {%
```

First use

```
10731      \glsfirstpluralaccessdisplay
10732      {\mfirstucMakeUppercase{\Glsentryfirstplural{\glslabel}}}%
10733      {\glslabel}%
10734      \mfirstucMakeUppercase{\glsinsert}%
10735      }%
10736      }%
10737      }%
10738      {%
```

Singular form

```
10739      \glscapscale
10740      {%
```

Don't adjust case

```
10741      \ifglused\glslabel
10742      {%
```

Subsequent use

```
10743      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10744      \glsinsert
10745      }%
10746      {%
```

#### First use

```

10747      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10748      \glsinsert
10749      }%
10750      }%
10751      {%

```

#### Make first letter upper case

```

10752      \ifglsused\glslabel
10753      {%

```

#### Subsequent use

```

10754      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10755      \glsinsert
10756      }%
10757      {%

```

#### First use

```

10758      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10759      \glsinsert
10760      }%
10761      }%
10762      {%

```

#### Make all upper case

```

10763      \ifglsused\glslabel
10764      {%

```

#### Subsequent use

```

10765      \glstextaccessdisplay
10766      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10767      \mfirstucMakeUppercase{\glsinsert}%
10768      }%
10769      {%

```

#### First use

```

10770      \glsfirstaccessdisplay
10771      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10772      \mfirstucMakeUppercase{\glsinsert}%
10773      }%
10774      }%
10775      }%
10776      }%
10777      {%

```

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.

```

10778      \glscustomtext\glsinsert
10779      }%
10780      }

```

`\glsgenacfmt`   Redefine to include accessibility information.

```
10781 \renewcommand*{\glsgenacfmt}{%
10782   \ifdefempty\glscustomtext
10783   {%
10784     \ifglused\glslabel
10785     {%
```

Subsequent use:

```
10786     \glsifplural
10787     {%
```

Subsequent plural form:

```
10788     \glscapscase
10789     {%
```

Subsequent plural form, don't adjust case:

```
10790     \acronymfont
10791     {\glsshortpluralaccessdisplay
10792      {\glentryshortpl{\glslabel}}{\glslabel}}%
10793     \glsinsert
10794   }%
10795   {%
```

Subsequent plural form, make first letter upper case:

```
10796     \acronymfont
10797     {\glsshortpluralaccessdisplay
10798      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10799     \glsinsert
10800   }%
10801   {%
```

Subsequent plural form, all caps:

```
10802     \mfirstucMakeUppercase
10803     {\acronymfont
10804      {\glsshortpluralaccessdisplay
10805       {\glentryshortpl{\glslabel}}{\glslabel}}%
10806      \glsinsert}%
10807   }%
10808   }%
10809   {%
```

Subsequent singular form

```
10810     \glscapscase
10811     {%
```

Subsequent singular form, don't adjust case:

```
10812     \acronymfont
10813     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10814     \glsinsert
10815   }%
10816   {%
```



Subsequent singular form, make first letter upper case:

```
10817      \acronymfont
10818      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10819      \glsinsert
10820      }%
10821      {%
```

Subsequent singular form, all caps:

```
10822      \mfirstucMakeUppercase
10823      {\acronymfont{%
10824      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10825      \glsinsert}%
10826      }%
10827      }%
10828      }%
10829      {%
```

First use:

```
10830      \glsifplural
10831      {%
```

First use plural form:

```
10832      \glscapscase
10833      {%
```

First use plural form, don't adjust case:

```
10834      \genplacrfullformat{\glslabel}{\glsinsert}%
10835      }%
10836      {%
```

First use plural form, make first letter upper case:

```
10837      \Genplacrfullformat{\glslabel}{\glsinsert}%
10838      }%
10839      {%
```

First use plural form, all caps:

```
10840      \mfirstucMakeUppercase
10841      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10842      }%
10843      }%
10844      {%
```

First use singular form

```
10845      \glscapscase
10846      {%
```

First use singular form, don't adjust case:

```
10847      \genacrfullformat{\glslabel}{\glsinsert}%
10848      }%
10849      {%
```

First use singular form, make first letter upper case:

```
10850      \Genacrfullformat{\glslabel}{\glsinsert}%
10851      }%
10852      {%
```

First use singular form, all caps:

```
10853      \mfirstucMakeUppercase
10854      {\genacrfullformat{\glslabel}{\glsinsert}}%
10855      }%
10856      }%
10857      }%
10858      }%
10859      {%
```

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.

```
10860      \glscustomtext
10861      }%
10862 }
```

`enacrfullformat` Redefine to include accessibility information.

```
10863 \renewcommand*{\genacrfullformat}[2]{%
10864   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10865   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
10866 }
```

`enacrfullformat` Redefine to include accessibility information.

```
10867 \renewcommand*{\Genacrfullformat}[2]{%
10868   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10869   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
10870 }
```

`placrfullformat` Redefine to include accessibility information.

```
10871 \renewcommand*{\genplacrfullformat}[2]{%
10872   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10873   (\glsshortpluralaccessdisplay
10874     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10875 }
```

`placrfullformat` Redefine to include accessibility information.

```
10876 \renewcommand*{\Genplacrfullformat}[2]{%
10877   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10878   (\glsshortpluralaccessdisplay
10879     {\protect\firstacronymfont{\Glsentryshortpl{#1}}}{#1}}%
10880 }
```

`\@acrshort`

```
10881 \def\@acrshort#1#2[#3]{%
10882   \glsdoifexists{#2}%
```

```

10883 {%
10884   \let\do@gl@s@link@checkfirsthyper\relax

10885   \let\gl@sifplural\@secondoftwo
10886   \let\gl@scapscase\@firstofthree
10887   \let\gl@sinsert\@empty
10888   \def\glscustomtext{%
10889     \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10890   }%

  Call \@gl@s@link
10891   \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
10892 }%

10893 \glspostlinkhook
10894 }

```

\@Acrshort

```

10895 \def\@Acrshort#1#2[#3]{%
10896   \gl@sdoifexists{#2}%
10897   {%
10898     \let\do@gl@s@link@checkfirsthyper\relax

10899     \let\gl@sifplural\@secondoftwo
10900     \let\gl@scapscase\@secondofthree
10901     \let\gl@sinsert\@empty
10902     \def\glscustomtext{%
10903       \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10904     }%

      Call \@gl@s@link
10905       \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
10906     }%

10907     \glspostlinkhook
10908 }

```

\@ACRshort

```

10909 \def\@ACRshort#1#2[#3]{%
10910   \gl@sdoifexists{#2}%
10911   {%
10912     \let\do@gl@s@link@checkfirsthyper\relax

10913     \let\gl@sifplural\@secondoftwo
10914     \let\gl@scapscase\@thirdofthree
10915     \let\gl@sinsert\@empty
10916     \def\glscustomtext{%
10917       \acronymfont{\glsshortaccessdisplay
10918         {\MakeUppercase{\gl@sentryshort{#2}}}{#2}}#3%
10919     }%

```

```

    Call \@gls@link
10920   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10921   }%

10922   \glspostlinkhook
10923 }

```

\@acrlong

```

10924 \def\@acrlong#1#2[#3]{%
10925   \glsdoifexists{#2}%
10926   {%
10927     \let\do@gls@link@checkfirsthyper\relax

10928     \let\glsifplural\@secondoftwo
10929     \let\glscapscase\@firstofthree
10930     \let\glsinsert\@empty
10931     \def\glscustomtext{%
10932       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10933     }%

```

```

    Call \@gls@link
10934   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10935   }%

10936   \glspostlinkhook
10937 }

```

\@Acrlong

```

10938 \def\@Acrlong#1#2[#3]{%
10939   \glsdoifexists{#2}%
10940   {%
10941     \let\do@gls@link@checkfirsthyper\relax

10942     \let\glsifplural\@secondoftwo
10943     \let\glscapscase\@firstofthree
10944     \let\glsinsert\@empty
10945     \def\glscustomtext{%
10946       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10947     }%

```

```

    Call \@gls@link
10948   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10949   }%

10950   \glspostlinkhook
10951 }

```

\@ACRlong

```

10952 \def\@ACRlong#1#2[#3]{%
10953   \glsdoifexists{#2}%
10954   {%
10955     \let\do@gls@link@checkfirsthyper\relax

```

```

10956 \let\glsifplural\@secondoftwo
10957 \let\glsifcaps\@firstofthree
10958 \let\glsinsert\@empty
10959 \def\glscustomtext{%
10960 \acronymfont{\glslongaccessdisplay{%
10961 \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10962 }%

Call \@gls@link
10963 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10964 }%

10965 \glspostlinkhook
10966 }

```

## 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.

```

10967 \renewcommand*{\glossentryname}[1]{%
10968 \glsdoifexists{#1}%
10969 {%
10970 \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10971 }%
10972 }

10973 \renewcommand*{\glossentrydesc}[1]{%
10974 \glsdoifexists{#1}%
10975 {%
10976 \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10977 }%
10978 }

10979 \renewcommand*{\glossentrydesc}[1]{%
10980 \glsdoifexists{#1}%
10981 {%
10982 \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10983 }%
10984 }

10985 \renewcommand*{\Glossentrydesc}[1]{%
10986 \glsdoifexists{#1}%
10987 {%
10988 \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10989 }%
10990 }

```

```

10991 \renewcommand*{\glossentrysymbol}[1]{%
10992   \glsdoifexists{#1}%
10993   {%
10994     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10995   }%
10996 }

10997 \renewcommand*{\Glossentrysymbol}[1]{%
10998   \glsdoifexists{#1}%
10999   {%
11000     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
11001   }%
11002 }

```

ssaryentryfield

```

11003 \newcommand*{\accsuppglossaryentryfield}[5]{%
11004   \glossaryentryfield{#1}%
11005   {\glsnameaccessdisplay{#2}{#1}}%
11006   {\glsdescriptionaccessdisplay{#3}{#1}}%
11007   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
11008 }

```

rysubentryfield

```

11009 \newcommand*{\accsuppglossarysubentryfield}[6]{%
11010   \glossarysubentryfield{#1}{#2}%
11011   {\glsnameaccessdisplay{#3}{#2}}%
11012   {\glsdescriptionaccessdisplay{#4}{#2}}%
11013   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
11014 }

```

## 5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short    *<long>* (*<short>*) acronym style.

```

11015 \renewacronymstyle{long-short}%
11016 {%

```

Check for long form in case this is a mixed glossary.

```

11017   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11018 }%
11019 {%
11020   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11021   \renewcommand*{\genacrfullformat}[2]{%
11022     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
11023     (\glsshortaccessdisplay
11024       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11025   }%
11026   \renewcommand*{\Genacrfullformat}[2]{%

```

```

11027 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11028 (\glsshortaccessdisplay
11029   {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
11030 }%
11031 \renewcommand*{\genplacrfullformat}[2]{%
11032   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11033   (\glsshortpluralaccessdisplay
11034     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11035   }%
11036 \renewcommand*{\Genplacrfullformat}[2]{%
11037   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11038   (\glsshortpluralaccessdisplay
11039     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
11040   }%
11041 \renewcommand*{\acronymentry}[1]{%
11042   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
11043 \renewcommand*{\acronymsort}[2]{##1}%
11044 \renewcommand*{\acronymfont}[1]{##1}%
11045 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11046 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11047 }

```

short-long (*short*) (*long*) acronym style.

```

11048 \renewacronymstyle{short-long}%
11049 {%

```

Check for long form in case this is a mixed glossary.

```

11050 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11051 }%
11052 {%
11053 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11054 \renewcommand*{\genacrfullformat}[2]{%
11055   \glsshortaccessdisplay
11056     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
11057     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11058   }%
11059 \renewcommand*{\Genacrfullformat}[2]{%
11060   \glsshortaccessdisplay
11061     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
11062     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
11063   }%
11064 \renewcommand*{\genplacrfullformat}[2]{%
11065   \glsshortpluralaccessdisplay
11066     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
11067     (\glslongpluralaccessdisplay
11068       {\glsentrylongpl{##1}}{##1}})%
11069   }%
11070 \renewcommand*{\Genplacrfullformat}[2]{%
11071   \glsshortpluralaccessdisplay
11072     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

11073 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11074 }%
11075 \renewcommand*{\acronymentry}[1]{%
11076   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11077 \renewcommand*{\acronymsort}[2]{##1}%
11078 \renewcommand*{\acronymfont}[1]{##1}%
11079 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11080 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11081 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

11082 \renewacronymstyle{long-short-desc}%
11083 {%
11084   \GlsUseAcrEntryDisplayStyle{long-short}%
11085 }%
11086 {%
11087   \GlsUseAcrStyleDefs{long-short}%
11088   \renewcommand*{\GenericAcronymFields}{}%
11089   \renewcommand*{\acronymsort}[2]{##2}%
11090   \renewcommand*{\acronymentry}[1]{%
11091     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11092     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11093 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11094 \renewacronymstyle{long-sc-short-desc}%
11095 {%
11096   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
11097 }%
11098 {%
11099   \GlsUseAcrStyleDefs{long-sc-short}%
11100   \renewcommand*{\GenericAcronymFields}{}%
11101   \renewcommand*{\acronymsort}[2]{##2}%
11102   \renewcommand*{\acronymentry}[1]{%
11103     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11104     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11105 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11106 \renewacronymstyle{long-sm-short-desc}%
11107 {%
11108   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
11109 }%
11110 {%
11111   \GlsUseAcrStyleDefs{long-sm-short}%
11112   \renewcommand*{\GenericAcronymFields}{}%

```



```

11113 \renewcommand*{\acronymsort}[2]{##2}%
11114 \renewcommand*{\acronymentry}[1]{%
11115     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11116     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11117 }

```

short-long-desc *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

11118 \renewacronymstyle{short-long-desc}%
11119 {%
11120     \GlsUseAcrEntryDisplayStyle{short-long}%
11121 }%
11122 {%
11123     \GlsUseAcrStyleDefs{short-long}%
11124     \renewcommand*{\GenericAcronymFields}{}%
11125     \renewcommand*{\acronymsort}[2]{##2}%
11126     \renewcommand*{\acronymentry}[1]{%
11127         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11128         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11129 }

```

short-long-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11130 \renewacronymstyle{sc-short-long-desc}%
11131 {%
11132     \GlsUseAcrEntryDisplayStyle{sc-short-long}%
11133 }%
11134 {%
11135     \GlsUseAcrStyleDefs{sc-short-long}%
11136     \renewcommand*{\GenericAcronymFields}{}%
11137     \renewcommand*{\acronymsort}[2]{##2}%
11138     \renewcommand*{\acronymentry}[1]{%
11139         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11140         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11141 }

```

short-long-desc *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11142 \renewacronymstyle{sm-short-long-desc}%
11143 {%
11144     \GlsUseAcrEntryDisplayStyle{sm-short-long}%
11145 }%
11146 {%
11147     \GlsUseAcrStyleDefs{sm-short-long}%
11148     \renewcommand*{\GenericAcronymFields}{}%
11149     \renewcommand*{\acronymsort}[2]{##2}%
11150     \renewcommand*{\acronymentry}[1]{%
11151         \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11152         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

11153 }

dua *<long>* only acronym style.

11154 \renewacronymstyle{dua}%  
11155 {%

Check for long form in case this is a mixed glossary.

11156 \ifdefempty\glscustomtext  
11157 {%  
11158 \ifglshaslong{\glslabel}%  
11159 {%  
11160 \glsifplural  
11161 {%

Plural form:

11162 \glscapscase  
11163 {%

Plural form, don't adjust case:

11164 \glslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%  
11165 \glsinsert  
11166 }%  
11167 {%

Plural form, make first letter upper case:

11168 \glslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%  
11169 \glsinsert  
11170 }%  
11171 {%

Plural form, all caps:

11172 \glslongpluralaccessdisplay  
11173 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%  
11174 \mfirstucMakeUppercase{\glsinsert}%  
11175 }%  
11176 }%  
11177 {%

Singular form

11178 \glscapscase  
11179 {%

Singular form, don't adjust case:

11180 \glslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert  
11181 }%  
11182 {%

Subsequent singular form, make first letter upper case:

11183 \glslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert  
11184 }%  
11185 {%

Subsequent singular form, all caps:

```

11186         \glslongaccessdisplay
11187         {\mfirstucMakeUppercase
11188          {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11189         \mfirstucMakeUppercase{\glsinsert}%
11190     }%
11191 }%
11192 }%
11193 {%

```

Not an acronym:

```

11194     \glsgenentryfmt
11195 }%
11196 }%
11197 {\glscustomtext\glsinsert}%
11198}%
11199{%
1200 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
1201 \renewcommand*{\acrfullfmt}[3]{%
1202     \glslink[##1]{##2}{%
1203         \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
1204         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
1205 \renewcommand*{\Acrfullfmt}[3]{%
1206     \glslink[##1]{##2}{%
1207         \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
1208         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
1209 \renewcommand*{\ACRfullfmt}[3]{%
1210     \glslink[##1]{##2}{%
1211         \glslongaccessdisplay
1212         {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
1213         (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
1214 \renewcommand*{\acrfullplfmt}[3]{%
1215     \glslink[##1]{##2}{%
1216         \glslongpluralaccessdisplay
1217         {\glsentrylongpl{##2}}{##2}##3\space
1218         (\glsshortpluralaccessdisplay
1219         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
1220 \renewcommand*{\ACRfullplfmt}[3]{%
1221     \glslink[##1]{##2}{%
1222         \glslongpluralaccessdisplay
1223         {\Glsentrylongpl{##2}}{##2}##3\space
1224         (\glsshortpluralaccessdisplay
1225         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
1226 \renewcommand*{\ACRfullplplfmt}[3]{%
1227     \glslink[##1]{##2}{%
1228         \glslongpluralaccessdisplay
1229         {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
1230         (\glsshortpluralaccessdisplay
1231         {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
1232 \renewcommand*{\glsentryfull}[1]{%

```

```

11233 \glslongaccessdisplay{\glsentrylong{##1}}\space
11234 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11235 }%
11236 \renewcommand*{\Glsentryfull}[1]{%
11237 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11238 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11239 }%
11240 \renewcommand*{\Glsentryfullpl}[1]{%
11241 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11242 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11243 }%
11244 \renewcommand*{\Glsentryfullpl}[1]{%
11245 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11246 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11247 }%
11248 \renewcommand*{\acronymentry}[1]{%
11249 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11250 \renewcommand*{\acronymsort}[2]{##1}%
11251 \renewcommand*{\acronymfont}[1]{##1}%
11252 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11253 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11254 \renewacronymstyle{dua-desc}%
11255 {%
11256 \GlsUseAcrEntryDispStyle{dua}%
11257 }%
11258 {%
11259 \GlsUseAcrStyleDefs{dua}%
11260 \renewcommand*{\GenericAcronymFields}{}%
11261 \renewcommand*{\acronymentry}[1]{%
11262 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11263 \renewcommand*{\acronymsort}[2]{##2}%
11264 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11265 \renewacronymstyle{footnote}%
11266 {%
11267 \ifglshaslong{\glslabel}{\glsacronymfont}{\glsacronymentryfmt}%
11268 }%
11269 {%
11270 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11271 \glshyperfirstfalse
11272 \renewcommand*{\genacrfullformat}[2]{%
11273 \glsshortaccessdisplay
11274 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

11275 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11276 }%
11277 \renewcommand*{\Genacrfullformat}[2]{%
11278 \glsshortaccessdisplay
11279 {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11280 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11281 }%
11282 \renewcommand*{\genplacrfullformat}[2]{%
11283 \glsshortpluralaccessdisplay
11284 {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11285 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11286 }%
11287 \renewcommand*{\Genplacrfullformat}[2]{%
11288 \glsshortpluralaccessdisplay
11289 {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11290 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11291 }%
11292 \renewcommand*{\acronymentry}[1]{%
11293 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11294 \renewcommand*{\acronymsort}[2]{##1}%
11295 \renewcommand*{\acronymfont}[1]{##1}%
11296 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11297 \renewcommand*{\acrfullfmt}[3]{%
11298 \glslink[##1]{##2}{%
11299 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11300 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11301 \renewcommand*{\Acrfullfmt}[3]{%
11302 \glslink[##1]{##2}{%
11303 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11304 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11305 \renewcommand*{\ACRfullfmt}[3]{%
11306 \glslink[##1]{##2}{%
11307 \glsshortaccessdisplay
11308 {\mfirstucMakeUppercase
11309 {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11310 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}%
11311 \renewcommand*{\acrfullplfmt}[3]{%
11312 \glslink[##1]{##2}{%
11313 \glsshortpluralaccessdisplay
11314 {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11315 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}%
11316 \renewcommand*{\Acrfullplfmt}[3]{%
11317 \glslink[##1]{##2}{%
11318 \glsshortpluralaccessdisplay
11319 {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11320 (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}%
11321 \renewcommand*{\ACRfullplfmt}[3]{%
11322 \glslink[##1]{##2}{%

```

```

11323 \glsshortpluralaccessdisplay
11324 {\mfirstucMakeUppercase
11325 {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
11326 (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

11327 \renewcommand*{\glentryfull}[1]{%
11328 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
11329 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11330 \renewcommand*{\Glsentryfull}[1]{%
11331 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
11332 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11333 \renewcommand*{\glentryfullpl}[1]{%
11334 \glsshortpluralaccessdisplay
11335 {\acronymfont{\glentryshortpl{##1}}{##1}\space
11336 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11337 \renewcommand*{\Glsentryfullpl}[1]{%
11338 \glsshortpluralaccessdisplay
11339 {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
11340 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11341 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11342 \renewacronymstyle{footnote-sc}%
11343 {%
11344 \GlsUseAcrEntryDispStyle{footnote}%
11345 }%
11346 {%
11347 \GlsUseAcrStyleDefs{footnote}%
11348 \renewcommand{\acronymentry}[1]{%
11349 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11350 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11351 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11352 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11353 \renewacronymstyle{footnote-sm}%
11354 {%
11355 \GlsUseAcrEntryDispStyle{footnote}%
11356 }%
11357 {%
11358 \GlsUseAcrStyleDefs{footnote}%
11359 \renewcommand{\acronymentry}[1]{%
11360 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11361 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11362 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11363 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11364 \renewacronymstyle{footnote-desc}%
11365 {%
11366   \GlsUseAcrEntryDisplayStyle{footnote}%
11367 }%
11368 {%
11369   \GlsUseAcrStyleDefs{footnote}%
11370   \renewcommand*{\GenericAcronymFields}{}%
11371   \renewcommand*{\acronymsort}[2]{##2}%
11372   \renewcommand*{\acronymentry}[1]{%
11373     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11374     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11375 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11376 \renewacronymstyle{footnote-sc-desc}%
11377 {%
11378   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11379 }%
11380 {%
11381   \GlsUseAcrStyleDefs{footnote-sc}%
11382   \renewcommand*{\GenericAcronymFields}{}%
11383   \renewcommand*{\acronymsort}[2]{##2}%
11384   \renewcommand*{\acronymentry}[1]{%
11385     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11386     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11387 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11388 \renewacronymstyle{footnote-sm-desc}%
11389 {%
11390   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11391 }%
11392 {%
11393   \GlsUseAcrStyleDefs{footnote-sm}%
11394   \renewcommand*{\GenericAcronymFields}{}%
11395   \renewcommand*{\acronymsort}[2]{##2}%
11396   \renewcommand*{\acronymentry}[1]{%
11397     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11398     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11399 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11400 \renewcommand*{\newacronymhook}{%
11401   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11402     \the\glskeylisttok}%
11403   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11404 }

1tNewAcronymDef Modify default style to use access text:

```
11405 \renewcommand*{\DefaultNewAcronymDef}{%
11406   \edef\@do@newglossaryentry{%
11407     \noexpand\newglossaryentry{\the\glslabeltok}%
11408     {%
11409       type=\acronymtype,%
11410       name={\the\glsshorttok},%
11411       description={\the\glslongtok},%
11412       descriptionaccess=\relax,%
11413       text={\the\glsshorttok},%
11414       access={\noexpand\@glo@textaccess},%
11415       sort={\the\glsshorttok},%
11416       short={\the\glsshorttok},%
11417       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11418       shortaccess={\the\glslongtok},%
11419       long={\the\glslongtok},%
11420       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11421       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11422       first={\noexpand\glslongaccessdisplay
11423         {\the\glslongtok}{\the\glslabeltok}\space
11424         (\noexpand\glsshortaccessdisplay
11425           {\the\glsshorttok}{\the\glslabeltok})},%
11426       plural={\the\glsshorttok\acrpluralsuffix},%
11427       firstplural={\noexpand\glslongpluralaccessdisplay
11428         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11429         (\noexpand\glsshortpluralaccessdisplay
11430           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11431       firstaccess=\relax,%
11432       firstpluralaccess=\relax,%
11433       textaccess={\noexpand\@glo@shortaccess},%
11434       \the\glskeylisttok
11435     }%
11436   }%
11437   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11438   \let\@org@gls@assign@plural\gls@assign@plural
11439   \let\@org@gls@assign@descplural\gls@assign@descplural
11440   \def\gls@assign@firstpl##1##2{%
11441     \@gls@expand@field{##1}{firstpl}{##2}%
11442   }%
11443   \def\gls@assign@plural##1##2{%
11444     \@gls@expand@field{##1}{plural}{##2}%
11445   }%
11446   \def\gls@assign@descplural##1##2{%
11447     \@gls@expand@field{##1}{descplural}{##2}%
11448   }%
11449   \@do@newglossaryentry
11450   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```



```

11451 \let\gls@assign@plural\@org@gls@assign@plural
11452 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11453 }

```

teNewAcronymDef

```

11454 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11455 \edef\@do@newglossaryentry{%
11456 \noexpand\newglossaryentry{\the\glslabeltok}%
11457 {%
11458 type=\acronymtype,%
11459 name={\noexpand\acronymfont{\the\glsshorttok}},%
11460 sort={\the\glsshorttok},%
11461 text={\the\glsshorttok},%
11462 short={\the\glsshorttok},%
11463 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11464 shortaccess={\the\glslongtok},%
11465 long={\the\glslongtok},%
11466 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11467 access={\noexpand\@glo@textaccess},%
11468 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11469 symbol={\the\glslongtok},%
11470 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11471 firstpluralaccess=\relax,
11472 textaccess={\noexpand\@glo@shortaccess},%
11473 \the\glskeylisttok
11474 }%
11475 }%
11476 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11477 \let\@org@gls@assign@plural\gls@assign@plural
11478 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11479 \def\gls@assign@firstpl##1##2{%
11480 \@@gls@expand@field{##1}{firstpl}{##2}%
11481 }%
11482 \def\gls@assign@plural##1##2{%
11483 \@@gls@expand@field{##1}{plural}{##2}%
11484 }%
11485 \def\gls@assign@symbolplural##1##2{%
11486 \@@gls@expand@field{##1}{symbolplural}{##2}%
11487 }%
11488 \do@newglossaryentry
11489 \let\gls@assign@plural\@org@gls@assign@plural
11490 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11491 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11492 }

```

onNewAcronymDef

```

11493 \renewcommand*{\DescriptionNewAcronymDef}{%
11494 \edef\@do@newglossaryentry{%
11495 \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11496 {%
11497     type=\acronymtype,%
11498     name={\noexpand
11499         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11500     access={\noexpand\@glo@textaccess},%
11501     sort={\the\glsshorttok},%
11502     short={\the\glsshorttok},%
11503     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11504     shortaccess={\the\glslongtok},%
11505     long={\the\glslongtok},%
11506     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11507     first={\the\glslongtok},%
11508     firstaccess=\relax,
11509     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11510     text={\the\glsshorttok},%
11511     textaccess={\the\glslongtok},%
11512     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11513     symbol={\noexpand\@glo@text},%
11514     symbolaccess={\noexpand\@glo@textaccess},%
11515     symbolplural={\noexpand\@glo@plural},%
11516     firstpluralaccess=\relax,
11517     textaccess={\noexpand\@glo@shortaccess},%
11518     \the\glskeylisttok}%
11519 }%
11520 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11521 \let\@org@gls@assign@plural\gls@assign@plural
11522 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11523 \def\gls@assign@firstpl##1##2{%
11524     \@gls@expand@field{##1}{firstpl}{##2}%
11525 }%
11526 \def\gls@assign@plural##1##2{%
11527     \@gls@expand@field{##1}{plural}{##2}%
11528 }%
11529 \def\gls@assign@symbolplural##1##2{%
11530     \@gls@expand@field{##1}{symbolplural}{##2}%
11531 }%
11532 \do@newglossaryentry
11533 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11534 \let\gls@assign@plural\@org@gls@assign@plural
11535 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11536 }

```

teNewAcronymDef

```

11537 \renewcommand*{\FootnoteNewAcronymDef}{%
11538     \edef\@do@newglossaryentry{%
11539         \noexpand\newglossaryentry{\the\glslabeltok}%
11540         {%
11541             type=\acronymtype,%
11542             name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11543 sort={\the\glsshorttok},%
11544 text={\the\glsshorttok},%
11545 textaccess={\the\glslongtok},%
11546 access={\noexpand\@glo@textaccess},%
11547 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11548 short={\the\glsshorttok},%
11549 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11550 long={\the\glslongtok},%
11551 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11552 description={\the\glslongtok},%
11553 descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11554 \the\glskeylisttok
11555 }%
11556 }%
11557 \let\@org@gls@assign@plural\gls@assign@plural
11558 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11559 \let\@org@gls@assign@descplural\gls@assign@descplural
11560 \def\gls@assign@firstpl##1##2{%
11561   \@gls@expand@field{##1}{firstpl}{##2}%
11562 }%
11563 \def\gls@assign@plural##1##2{%
11564   \@gls@expand@field{##1}{plural}{##2}%
11565 }%
11566 \def\gls@assign@descplural##1##2{%
11567   \@gls@expand@field{##1}{descplural}{##2}%
11568 }%
11569 \do@newglossaryentry
11570 \let\gls@assign@plural\@org@gls@assign@plural
11571 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11572 \let\gls@assign@descplural\@org@gls@assign@descplural
11573 }

```

# 11NewAcronymDef

```

11574 \renewcommand*{\SmallNewAcronymDef}{%
11575   \edef\@do@newglossaryentry{%
11576     \noexpand\newglossaryentry{\the\glslabeltok}%
11577     {%
11578       type=\acronymtype,%
11579       name={\noexpand\acronymfont{\the\glsshorttok}},%
11580       access={\noexpand\@glo@symbolaccess},%
11581       sort={\the\glsshorttok},%
11582       short={\the\glsshorttok},%
11583       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11584       shortaccess={\the\glslongtok},%
11585       long={\the\glslongtok},%
11586       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11587       text={\noexpand\@glo@short},%
11588       textaccess={\noexpand\@glo@shortaccess},%
11589       plural={\noexpand\@glo@shortpl},%

```

```

11590     first={\the\glslongtok},%
11591     firstaccess=\relax,
11592     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11593     description={\noexpand\@glo@first},%
11594     descriptionplural={\noexpand\@glo@firstplural},%
11595     symbol={\the\glsshorttok},%
11596     symbolaccess={\the\glslongtok},%
11597     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11598     \the\glskeylisttok
11599 }%
11600 }%
11601 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11602 \let\@org@gls@assign@plural\gls@assign@plural
11603 \let\@org@gls@assign@descplural\gls@assign@descplural
11604 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11605 \def\gls@assign@firstpl##1##2{%
11606   \@@gls@expand@field{##1}{firstpl}{##2}%
11607 }%
11608 \def\gls@assign@plural##1##2{%
11609   \@@gls@expand@field{##1}{plural}{##2}%
11610 }%
11611 \def\gls@assign@descplural##1##2{%
11612   \@@gls@expand@field{##1}{descplural}{##2}%
11613 }%
11614 \def\gls@assign@symbolplural##1##2{%
11615   \@@gls@expand@field{##1}{symbolplural}{##2}%
11616 }%
11617 \do@newglossaryentry
11618 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11619 \let\gls@assign@plural\@org@gls@assign@plural
11620 \let\gls@assign@descplural\@org@gls@assign@descplural
11621 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11622 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```

11623 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

```

pluralaccesskey

```

11624 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

```

lslongaccesskey

```

11625 \newcommand*{\glslongaccesskey}{\glslongkey access}%

```

pluralaccesskey

```

11626 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

## 5.5 Debugging Commands

owglonameaccess

```
11627 \newcommand*{\showglonameaccess}[1]{%  
11628   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname  
11629 }
```

owglotextaccess

```
11630 \newcommand*{\showglotextaccess}[1]{%  
11631   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname  
11632 }
```

glopluralaccess

```
11633 \newcommand*{\showglopluralaccess}[1]{%  
11634   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname  
11635 }
```

wglofirstaccess

```
11636 \newcommand*{\showglofirstaccess}[1]{%  
11637   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname  
11638 }
```

rstpluralaccess

```
11639 \newcommand*{\showglofirstpluralaccess}[1]{%  
11640   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname  
11641 }
```

glosymbolaccess

```
11642 \newcommand*{\showglosymbolaccess}[1]{%  
11643   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname  
11644 }
```

bolpluralaccess

```
11645 \newcommand*{\showglosymbolpluralaccess}[1]{%  
11646   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname  
11647 }
```

owglodescaccess

```
11648 \newcommand*{\showglodescaccess}[1]{%  
11649   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname  
11650 }
```

escpluralaccess

```
11651 \newcommand*{\showglodescpluralaccess}[1]{%  
11652   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname  
11653 }
```

wgloshortaccess

```
11654 \newcommand*{\showgloshortaccess}[1]{%  
11655   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11656 }
```

ortpluralaccess

```
11657 \newcommand*{\showgloshortpluralaccess}[1]{%  
11658   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11659 }
```

owglolongaccess

```
11660 \newcommand*{\showglolongaccess}[1]{%  
11661   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11662 }
```

ongpluralaccess

```
11663 \newcommand*{\showglolongpluralaccess}[1]{%  
11664   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11665 }
```

## 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.

```
11666 \NeedsTeXFormat{LaTeX2e}
11667 \ProvidesPackage{glossaries-babel}[2017/08/24 v4.32 (NLCT)]
```

Load tracklang to obtain language settings.

```
11668 \RequirePackage{tracklang}
11669 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11670 \AnyTrackedLanguages
11671 {%
11672   \ForEachTrackedDialect{\this@dialect}{%
11673     \IfTrackedLanguageFileExists{\this@dialect}%
11674       {glossaries-}% prefix
11675       {.ldf}%
11676       {%
11677         \RequireGlossariesLang{\CurrentTrackedTag}%
11678       }%
11679       {%
11680         \PackageWarningNoLine{glossaries}%
11681           {No language module detected for ‘\this@dialect’.\MessageBreak
11682             Language modules need to be installed separately.\MessageBreak
11683             Please check on CTAN for a bundle called\MessageBreak
11684             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11685       }%
11686     }%
11687   }%
11688 }
```

### 6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11689 \NeedsTeXFormat{LaTeX2e}
11690 \ProvidesPackage{glossaries-polyglossia}[2017/08/24 v4.32 (NLCT)]
```

Load tracklang to obtain language settings.

```
11691 \RequirePackage{tracklang}
11692 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11693 \AnyTrackedLanguages
```

```

11694 {%
11695     \ForEachTrackedDialect{\this@dialect}{%
11696         \IfTrackedLanguageFileExists{\this@dialect}%
11697         {glossaries-}% prefix
11698         {.ldf}%
11699         {%
11700             \RequireGlossariesLang{\CurrentTrackedTag}%
11701         }%
11702         {%
11703             \PackageWarningNoLine{glossaries}%
11704             {No language module detected for ‘\this@dialect’.\MessageBreak
11705             Language modules need to be installed separately.\MessageBreak
11706             Please check on CTAN for a bundle called\MessageBreak
11707             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11708         }%
11709     }%
11710 }%
11711 {}%

```



# Glossary

`makeindex` An indexing application. [11](#), [27](#), [28](#), [176](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [11](#), [27](#), [28](#), [176](#)

# Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 7
General: Added range facility in format key . . . . . 112	
\writeist: Added spaces after \delimN and \delimR in ist file . . . . . 158	
1.04 (2007-08-03)	1.12 (2008-03-08)
General: Added \glstextformat . . . . . 96	\@GLSpl: now uses
1.05 (2007-08-10)	\glentrydescplural and
\glossarysection: added \@mkboth to \glossarysection . . . . . 39	\glentrysymbolplural instead of
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key . . . . . 80	\glentrydesc and
1.07 (2007-09-13)	\glentrysymbol . . . . . 125
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon . . . . . 110	\@Glspl@: now uses
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon . . . . . 156	\glentrydescplural and
1.08 (2007-10-13)	\glentrysymbolplural instead of
General: Added babel support . . . . . 33	\glentrydesc and
listgroup: changed listgroup style to use \glsgetgrouptitle . . . . . 270	\glentrysymbol . . . . . 124
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle . . . . . 271	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) . . . . . 120
1.1 (2008-02-22)	descriptionplural: new . . . . . 62
\@glossarysection: numbered sections and auto label added . . . . . 40	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) . . . . . 80
\@gls@tmpb: changed \toksdef to \newtoks . . . . . 114	descriptionplural support added . . . . . 79
\@gls@toc: numberline added . . . . . 42	symbolplural support added . . . . . 80
\@p@glossarysection: numbered sections and auto label added . . . . . 41	\Glsentrydescplural: New . . . . . 149
General: amsgen now loaded (\new@ifnextchar needed) . . . . . 4	\glentrydescplural: New . . . . . 149
translate: translate option added . . . . . 24	\Glsentrysymbolplural: New . . . . . 150
\setglossarysection: new . . . . . 40	\glentrysymbolplural: New . . . . . 150
numberedsection: numberedsection package option added . . . . . 7	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink . . . . . 236
	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink . . . . . 242
	symbolplural: new . . . . . 63

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter .....	127-134
\ACRfullpl: new .....	217
\Acrfullpl: new .....	217
\acrfullpl: new .....	216
\acrpluralsuffix: New .....	214
\gls@defglossaryentry: Changed default first value .....	80
Changed default firstplural value .....	80
Removed restriction on only using \newglossaryentry in the preamble	85
\newacronym: Removed restriction on only using \newacronym in the preamble .....	214
1.14 (2008-06-17)	
\@gls@hypergroup: new .....	265
General: added nonumberlist key to \printglossary .....	200
added numberedsection key to \printglossary .....	198
\firstacronymfont: new .....	217
\glsautoprefix: new .....	7
\glsnavhyperlink: changed \edef to \protected@edef .....	264
\glsnavhypertarget: added write to aux file .....	264
\glsnavigation: changed to only use labels for groups that are present ..	266
1.15 (2008-08-15)	
\@gls@link: added \glslabel .....	110
\gls@defglossaryentry: check for \@glo@first in description .....	84
check for \@glo@text in symbol .....	84
\gls@hypergroup: new .....	265
\glsnavhypertarget: added check if rerun required .....	264
\glssettoctitle: new .....	32
\printglossary: changed the way the TOC title is set .....	184
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	123
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	125
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	122
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	125
\@gls@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	121
\@glsdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	126
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	124
\@gls@target: raised the hypertarget so the target text doesn't scroll off the top of the page .....	120
\gls@defglossaryentry: Changed def to let .....	80
1.17 (2008-12-26)	
\@do@wrglossary: new .....	179
\@do@seeglossary: new .....	182
\@glo@storeentry: new .....	86
\@gls@glossary: changed definition to use \index instead of \@index ....	177
\@glsdefaultplural: new .....	67
\@glsdefaultsort: new .....	67
\@gls@hypernumber: new .....	211
\@glsnoname: new .....	67
\@glsnonextpages: new .....	201
General: added xindy support .....	27
parent: new .....	64
see: new .....	64
\gls@defglossaryentry: added nonumberlist key .....	80
added parent key .....	80
added see key .....	80
Stored main part of entry format when entry is defined .....	84
\gls@suffixF: new .....	37
\gls@suffixFF: new .....	38
\gls@wrglossary: modified to allow for xindy support .....	177

\glshyperlink: new .....	155	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	236
\glshypernumber: modified to allow material to be attached to location	211	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller .....	242
\glshnavhyperlink: replaced \hyperlink to \@glslink .....	264	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller .....	245
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	264	2.01 (2009 May 30) \@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit .....	110
\glsssee: new .....	183	\forall glossaries: replaced \ifthenelse with \ifx .....	51
\glssseeformat: new .....	183	\forall glossentries: replaced \ifthenelse with \ifx .....	51
\glssSetSuffixF: new .....	37	\glssdefmain: new .....	14
\glssSetSuffixFF: new .....	38	\glssdescwidth: changed \linewidth to \hsize .....	272, 294
\ifglssxindy: new .....	27	\glsslistdottedwidth: changed \linewidth to \hsize .....	272
\listfilename: added xindy support ...	36	\glsspagelistwidth: changed \linewidth to \hsize .....	272, 294
\newglossarystyle: made \newglossarystyle long .....	210	nomain: added nomain package option	15
\nopostdesc: new .....	35	\writeist: removed item_02 - no such makeindex key .....	162
nonumberlist: new .....	65	2.02 (2007-07-13) \@printglossary: suppressed warning globally rather than locally .....	187
\printglossary: added check to determine if \printglossary is already defined .....	184	2.02 (2009-07-13) \glossarysection: changed \@mkboth to \glossarymark .....	39
added print language to aux file .....	184	\glssglossarymark: New .....	39
order: order package option added ...	27	2.03 (2009-09-23) \@GLS@: Added check for hyperfirst ...	123
\writeist: added xindy support .....	158	\@GLSpl: Added check for hyperfirst ...	125
1.18 (2009-01-14) \@glsl@loadlist: new .....	9	\@GLs@: Added check for hyperfirst ...	122
\@glsl@loadlong: new .....	9	\@GLspl@: Added check for hyperfirst ..	125
\@glsl@loadsuper: new .....	9	\@glsl@: Added check for hyperfirst ...	121
\@glsl@loadtree: new .....	10	\@glsl@@link: new .....	108
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort .....	80	\@glsl@link: added \leavevmode ...	110
moved sort sanitization to \newglossaryentry .....	84	Moved entry existence check to avoid duplicate code .....	110
\glstarget: new .....	204	\@glsl@disp: Added check for hyperfirst	126
\oldacronym: new .....	213	\@glspl@: Added check for hyperfirst ..	124
nolist: new .....	9	\glssglossarymark: Added check to see if it's already defined .....	39
nolong: new .....	9	hyperfirst: new .....	26
sort: moved sanitization to \newglossaryentry .....	62		
nostyles: new .....	10		
nosuper: new .....	9		
notree: new .....	10		
1.19 (2009-03-02) \glsclearpage: new .....	42		
\glssdisp: new .....	126		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	240		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms .....	123
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms .....	125
\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms .....	122
\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms .....	125
\@glossaryentryfield: new .....	86
\@glossarysubentryfield: new .....	86
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms .....	121
\@glsacronymlists: new .....	16
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms .....	126
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms .....	124
\@newglossaryentryposthook: new ..	85
\@newglossaryentryprehook: new ...	85
acronymlists: new .....	17
\DeclareAcronymList: new .....	16
\DefineAcronymSynonyms: new .....	231
\gls@defglossaryentry: added user1-6 keys .....	80
\glsadd: fixed bug that ignored counter	156
\Glsentryuseri: new .....	151
\glsentryuseri: new .....	151
\Glsentryuserii: new .....	152
\glsentryuserii: new .....	151
\Glsentryuseriii: new .....	152
\glsentryuseriii: new .....	152
\Glsentryuseriv: new .....	152
\glsentryuseriv: new .....	152
\Glsentryuserv: new .....	152
\glsentryuserv: new .....	152
\Glsentryuservi: new .....	152
\glsentryuservi: new .....	152
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined. ....	59
\SetAcronymLists: new .....	17
\SetDefaultAcronymDisplayStyle: new .....	232
\SetDefaultAcronymStyle: new ....	233
\SetDescriptionAcronymDisplayStyle: new .....	238
\SetDescriptionDUAAcronymDisplayStyle: new .....	236
\SetDescriptionFootnoteAcronymDisplayStyle: new .....	234
\SetDUADisplayStyle: new .....	246
\SetFootnoteAcronymDisplayStyle: new .....	240
\SetSmallAcronymDisplayStyle: new	243
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco .....	126
Removed spurious brace. Patch provided by Sergiu Dotenco .....	126
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco .....	163
2.06 (2010-06-14)	
\altnewglossary: new .....	60
\CustomAcronymFields: new .....	248
\CustomNewAcronymDef: new .....	248
\SetCustomDisplayStyle: new .....	248
\SetCustomStyle: new .....	249
2.07 (2010-07-10)	
General: glsadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format) .....	155
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites .....	167
\gls@wrglossary: modified to take into account savewrites .....	177
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument .....	112
3.0 (2011-04-02)	
\@do@wrglossary: added check for hyper location prefix .....	180
modified to use new format .....	179
\@@glossarysec: replaced \@ifundefined with \ifcsundef ...	7
\@do@seeglossary: Sanitize and escape cross-referencing information ....	182
\@gls@counterwithin: new .....	11

\@gls@ifinlist: new .....	43	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter .....	156
\@gls@saveentrycounter .....	110	\GlsAddXdyCounters: new .....	43
added \@gls@setsort .....	110	\glseentrycounterlabel: new .....	203
\@gls@saveentrycounter: new .....	111	\glseentryitem: new .....	203
\@gls@setupsort@def: new .....	12	\Glsentrylong: new .....	153
\@gls@setupsort@standard: new ....	11	\glseentrylong: new .....	153
\@gls@setupsort@use: new .....	13	\Glsentrylongpl: new .....	153
\@gls@xdy@locationlist: new .....	46	\glseentrylongpl: new .....	153
\@glslink: replaced \@ifundefined		\Glsentryshort: new .....	153
with \ifcsundef .....	120	\glseentryshort: new .....	153
\@glsnextpages: new .....	201	\Glsentryshorttpl: new .....	153
\@print@glossary: replaced		\glseentryshorttpl: new .....	153
\@ifundefined with \ifcsundef ..	187	\glsgetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	207
\currentglossary .....	186	\gls glossarymark: replaced	
added \glsnextpages .....	186	\@ifundefined with \ifcsundef ..	39
make toctitle default to title .....	186	\glshyperlink: changed default from	
\@xdyattributelist: new .....	43	\glseentryname to \glseentrytext ..	155
General: added prefix to hyperlink ....	212	\glshypernumber: replaced	
etoolbox now loaded .....	4	\@ifundefined with \ifcsundef ..	211
replaced \@ifundefined with		\glsnumberformat: replaced	
\ifcsundef .....	31, 34, 106, 198	\@ifundefined with \ifcsundef ..	38
\acrfootnote: new .....	234	\glsrefentry: new .....	203
\ACRfull: added starred version .....	216	\glsresetsubentrycounter: new ...	202
\Acrfull: added starred version .....	215	\glsseeitem: hyperlink uses	
\acrfull: added starred version .....	215	\glsseeitemformat instead of	
\ACRfullpl: added starred version ...	217	\glseentryname .....	184
\Acrfullpl: added starred version ...	217	\glsseeitemformat: new .....	184
\acrfullpl: added starred version ...	216	\glssortnumberfmt: new .....	12
\acrlinkfootnote: new .....	234	\glsstepentry: new .....	202
\acrlinkfootnote: new .....	234	\glsstepsubentry: new .....	202
savewrites: new .....	28	\glssubentrycounterlabel: new ...	203
see: added \@glo@seeautonumberlist ..	64	\glssubentryitem: new .....	203
seeautonumberlist: new .....	9	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef .....	203
\@ifundefined with \ifcsundef ..	39	short: new .....	66
\glossarystyle: replaced		shortplural: new .....	66
\@ifundefined with \ifcsundef ..	209	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	52
\@ifundefined with \ifcsundef ..	27	\ifglseentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	53
\@gls@defsort .....	84	\istfile: deprecated .....	175
added short and long keys .....	80	glossaryentry: new .....	201
replaced \@ifundefined with		glossarysubentry: new .....	202
\ifcsundef .....	81	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	41	\newrobustcmd .....	69

<code>\newglossarystyle</code> : replaced		<code>\showglouseriii</code> : new	251
<code>\@ifundefined</code> with <code>\ifcsundef</code>	210	<code>\showglouseriv</code> : new	252
<code>\ns@newglossary</code> : added		<code>\showglouserv</code> : new	252
<code>\@gls@defsortcount</code>	60	<code>\showglouservi</code> : new	252
replaced <code>\@ifundefined</code> with		<code>subentrycounter</code> : new	11
<code>\ifcsundef</code>	59	<code>\writeist</code> : added xindy-only macro	
<code>entrycounter</code> : new	10	definitions to glossary open tag	160
<code>entrycounterwithin</code> : new	11	modified to support new format	158
<code>\oldacronym</code> : replaced <code>\@ifundefined</code>		3.01 (2011-04-12)	
with <code>\ifcsundef</code>	213	<code>\@glswritefiles</code> : added check for	
<code>compatible-2.07</code> : <code>compatible-2.07</code>		empty glossaries	175
option added	29	General: made robust	123
<code>long</code> : new	66	<code>\ACRfull</code> : made robust	216
<code>longplural</code> : new	66	<code>\Acrfull</code> : made robust	215
<code>nonumberlist</code> : now boolean	65	<code>\acrfull</code> : made robust	215
<code>sort</code> : new	11	<code>\acrfullformat</code> : removed	
<code>counter</code> : replaced <code>\@ifundefined</code> with		<code>\acronymfont</code> as it should already be	
<code>\ifcsundef</code>	63	set in the second argument.	215
<code>\printglossary</code> : replaced		<code>\ACRfullpl</code> : made robust	217
<code>\@ifundefined</code> with <code>\ifcsundef</code>	184	<code>\Acrfullpl</code> : made robust	217
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>		<code>\acrfullpl</code> : made robust	216
expanded options link options	234	<code>\ACRlong</code> : made robust	144
<code>\setentrycounter</code> : added optional		<code>\Acrlong</code> : made robust	143
argument	208	<code>\acrlong</code> : made robust	143
<code>\showacronymlists</code> : new	254	<code>\ACRlongpl</code> : made robust	146
<code>\showglocounter</code> : new	251	<code>\Acrlongpl</code> : made robust	145
<code>\showgloDESC</code> : new	252	<code>\acrlongpl</code> : made robust	145
<code>\showgloDESCplural</code> : new	252	<code>\ACRshort</code> : made robust	140
<code>\showglofirst</code> : new	250	<code>\Acrshort</code> : made robust	140
<code>\showglofirstpl</code> : new	251	<code>\acrshort</code> : made robust	139
<code>\showgloflag</code> : new	254	<code>\ACRshortpl</code> : made robust	142
<code>\showgloindex</code> : new	253	<code>\Acrshortpl</code> : made robust	141
<code>\showglolevel</code> : new	250	<code>\acrshortpl</code> : made robust	141
<code>\showglongame</code> : new	252	<code>\Gls</code> : made robust	122
<code>\showgloparent</code> : new	250	<code>\glsadd</code> : made robust	156
<code>\showgloplural</code> : new	250	<code>\glsaddall</code> : made robust	156
<code>\showglosort</code> : new	253	<code>\GLSdesc</code> : made robust	131
<code>\showglossaries</code> : new	254	<code>\Glsdesc</code> : made robust	131
<code>\showglossarycounter</code> : new	255	<code>\glsdesc</code> : made robust	131
<code>\showglossaryentries</code> : new	255	<code>\GLSdescplural</code> : made robust	132
<code>\showglossaryin</code> : new	254	<code>\Glsdescplural</code> : made robust	132
<code>\showglossaryout</code> : new	255	<code>\glsdescplural</code> : made robust	132
<code>\showglossarytitle</code> : new	255	<code>\glsfirst</code> : made robust	128
<code>\showglosymbol</code> : new	253	<code>\GLSfirstplural</code> : made robust	130
<code>\showglosymbolplural</code> : new	253	<code>\Glsfirstplural</code> : made robust	130
<code>\showglotext</code> : new	250	<code>\glsfirstplural</code> : made robust	129
<code>\showglotype</code> : new	251	<code>\glslink</code> : made robust	108
<code>\showglouserI</code> : new	251	<code>\GLSname</code> : made robust	131
<code>\showglouserII</code> : new	251	<code>\Glsname</code> : made robust	130

\glsname: made robust .....	130	\@printglossary: add a way to fetch	
\GLSpl: made robust .....	125	current entry label .....	186
\Glspl: made robust .....	124	savenumberlist: new .....	9
\glspl: made robust .....	123	ucmark: new .....	10
\GLSplural: made robust .....	129	\gls@defglossaryentry: added	
\GLSsymbol: made robust .....	133	numberlist element .....	83
\Glsymbol: made robust .....	133	\gls@save@numberlist: new .....	184
\glssymbol: made robust .....	133	\gls@wrglossary: added check for	
\GLSsymbolplural: made robust .....	134	glossary file defined .....	177
\Glsymbolplural: made robust .....	134	\glsdisplaynumberlist: new .....	154
\glssymbolplural: made robust .....	133	\glsentrycounter: set default value ..	111
\Glstext: made robust .....	127	\Glsentryfull: fixed bug (replaced	
\glstext: made robust .....	127	\glsentryshortpl with	
\GLSuseri: made robust .....	135	\glsentryshort) .....	153
\Glsuseri: made robust .....	135	\glsentryfullpl: fixed bug (replaced	
\glsuseri: made robust .....	134	\glsentryshort with	
\GLSuserii: made robust .....	136	\glsentryshortpl) .....	154
\Glsuserii: made robust .....	135	\glsentrynumberlist: new .....	154
\glsuserii: made robust .....	135	\glsmoveentry: new .....	85
\GLSuseriii: made robust .....	136	\glsresetsubentrycounter: new ...	202
\Glsuseriii: made robust .....	136	\ifglshaschildren: new .....	54
\glsuseriii: made robust .....	136	\ifglshasparent: new .....	55
\GLSuseriv: made robust .....	137	\makeglossaries: added list parser ..	170
\Glsuseriv: made robust .....	137	indexonlyfirst: new .....	26
\glsuseriv: made robust .....	137	\renewglossarystyle: new .....	210
\GLSuserv: made robust .....	138	\showglossaryentries: fixed misspelt	
\Glsuserv: made robust .....	138	command .....	255
\glsuserv: made robust .....	137	\SmallNewAcronymDef: fixed broken	
\GLSuservi: made robust .....	139	short and long plural .....	244
\Glsuservi: made robust .....	139	3.03 (2012/09/21)	
\glsuservi: made robust .....	138	\@gls@sanitizesort: new .....	20
3.02 (2012-05-19)		\@gls@setupsort@standard: used	
\glsnumlistlastsep: new .....	155	\@gls@sanitizesort .....	12
\glsnumlistsep: new .....	155	\@printglossary: allow title to override	
3.02 (2012-05-21)		default toctitle .....	185
\@do@wrglossary: changed		General: allow title to set toctitle .....	198
\@glslocref to		\glsinlinedescformat: new .....	268
\theglsentrycounter .....	181	\glsinlineemptydescformat: new ..	268
\@do@wrglossary: changed		\glsinlinenameformat: new .....	268
\@do@wr@glossary to test for		\glsinlinepostchild: new .....	268
indexonlyfirst option; put old		\glsinlinesubdescformat: new ....	268
\@do@wr@glossary code into		\glsinlinesubnameformat: new ....	268
\@do@wrglossary .....	177	\glspostinline: replaced “.” with	
\@gls@missingnumberlist: new .....	67	\glspostdescription .....	268
\@glswritefiles: added check for		list: added check for glsnogroupskip ..	270
existence of token in case		altlongragged4col: added check for	
\makeglossaries has been		glsnogroupskip .....	287
omitted .....	175	altsuperragged4col: added check for	
		glsnogroupskip .....	306



alttree: added check for		\gls@disablepagerefexpansion: new	178
glsnogroupskip .....	315	\gls@numberpage: new .....	178
index: added check for glsnogroupskip	309	\gls@protected@pagefmts: new ....	178
nogroupskip: new .....	10	\gls@romanpage: new .....	178
long: added check for glsnogroupskip	273	\glsdefmain: added check for doc	
long3col: added check for		package .....	14
glsnogroupskip .....	275	\glsorg@endtheglossary: new .....	5
long4col: added check for		\glsorg@theglossary: new .....	5
glsnogroupskip .....	276	\PrintChanges: new .....	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip .....	284	@@do@wrglossary: add Roman case.	
longragged3col: added check for		Fixed bugs in the else statements ..	180
glsnogroupskip .....	285	@gls@link: added check for	
nopostdot: new .....	10	“nohypertypes” .....	110
tree: added check for glsnogroupskip	310	mcolalttree: replaced ‘2’ with	
treenoname: added check for		\glsmcols .....	293
glsnogroupskip .....	312	mcolindex: replaced ‘2’ with \glsmcols	289
super: added check for glsnogroupskip	295	mcolindexspannav: replaced ‘2’ with	
super3col: added check for		\glsmcols .....	290
glsnogroupskip .....	297	mcoltree: replaced ‘2’ with \glsmcols	290
super4col: added check for		mcoltreenoname: replaced ‘2’ with	
glsnogroupskip .....	299	\glsmcols .....	292
superragged: added check for		mcoltreesspannav: replaced ‘2’ with	
glsnogroupskip .....	302	\glsmcols .....	291
superragged3col: added check for		\gls@protected@pagefmts: added	
glsnogroupskip .....	304	Roman to list .....	178
3.04 (2012-11-11)		\gls@Romanpage: new .....	178
altlist: replaced \newline with		\glsgetgrouplabel: fixed bug (typo in	
paragraph break .....	270	\equal) .....	208
3.04 (2012-11-18)		\nopostdesc: made robust .....	35
@@do@wrglossary: changed		3.05 (2013/04/21)	
\theglsentrycounter back to		@gls@nohyperlist: new .....	18
@glslocref .....	181	\GlsDeclareNoHyperList: new .....	18
@@do@wrglossary: modified to		nohypertypes: new .....	18
compensate for possible incorrect		3.06 (2013/06/17)	
page number .....	179	@xdy@main@language: Changed back to	
@gls@escbsdq: unsanitize		using \language .....	27
\gls@numberpage, \gls@alphpage,		\findrootlanguage: Obsoleted .....	50
\gls@Alphpage and		3.07 (2013-07-05)	
\gls@romanpage .....	113	@gls@link: fixed bug that failed to find	
@print@glossary: Moved aux write to		entry in list .....	110
end of document to prevent		\glossarypreamble: modified to work	
unwanted whatsit occurring here. ..	187	with \setglossarypreamble .....	38
General: Added check for doc package	4	\gls@docclearpage: added check for	
added datatool-base as a required		openright .....	41
package .....	4	\glspostdescription: Added	
added local key .....	107	spacefactor code .....	10
\gls@Alphpage: new .....	178	\GlsSetXdyCodePage: Added check for	
\gls@alphpage: new .....	178	fontspec .....	50

\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	238	\ifglshasdesc: new	55
\setglossarypreamble: new	39	\ifglshassymbol: new	55
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	287
list: updated list style to use \glossentry and \subglossentry	269	alttree: updated to use \glossentry and \subglossentry	314
listdotted: updated listdotted style to use \glossentry and \subglossentry	271	index: added paragraph break at end of environment	308
altlist: updated altlist style to use \glossentry and \subglossentry	270	updated to use \glossentry and \subglossentry	308
inline: updated inline style to use \glossentry and \subglossentry	267	long: updated to use \glossentry and \subglossentry	273
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	284
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	86	longragged3col: updated to use \glossentry and \subglossentry	285
updated for \glossentry	86	tree: updated to use \glossentry and \subglossentry	310
\@glossaryentryfield: switched to \glossentry	86	\setglossarystyle: new	209
\@glossarysubentryfield: switched to \subglossentry	86	\setglossentrycompatibility: new	206
General: added nogroupskip key to \printglossary	199	superragged: updated to use \glossentry and \subglossentry	302
removed definition of \@glossaryentryfield	357	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield	357	\@gls@assign@symbolplural@field: new	20
\compatibleglossentry: new	204	\@gls@default@value: new	63
\compatiblesubglossentry: new	205	\Glsentrydesc: made robust	149
\glossaryentryfield: deprecated	206	\Glsentrydescplural: made robust	149
\Glossentrydesc: new	205	\Glsentryfirst: made robust	150
\glossentrydesc: new	205	\Glsentryfirstplural: made robust	150
\Glossentryname: new	205	\Glsentryfull: made robust	153
\glossentryname: new	204	\Glsentryfullpl: made robust	154
\Glossentrysymbol: new	205	\Glsentrylong: made robust	153
\glossentrysymbol: new	205	\Glsentrylongpl: made robust	153
\gls@assign@desc@field: new	20	\Glsentryname: made robust	148
\gls@assign@descplural@field: new	20	\Glsentryplural: made robust	149
\gls@assign@field: new	69	\Glsentryshort: made robust	153
\gls@ifnotmeasuring: new	87	\Glsentryshortpl: made robust	153
\glsaddallunused: new	156	\Glsentrysymbol: made robust	150
\glsexpandfields: new	69	\Glsentrysymbolplural: made robust	150
\glsnoexpandfields: new	69	\Glsentrytext: made robust	149
\glssee: made robust	183	\Glsentryuseri: made robust	151
\glsseeformat: made robust	183	\Glsentryuserii: made robust	152
\glsseeitem: made robust	184	\Glsentryuseriii: made robust	152
\glsseelist: made robust	183	\Glsentryuseriv: made robust	152
\ifglsdessuppressed: new	55	\Glsentryuserv: made robust	152
		\glstextup: new	214

\ifglshassymbol: changed test to check for \@gls@default@symbol .....	55	\@Gls@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	122
3.10a (2013-09-28)		change to using \glentryfmt style commands .....	122
\gls@assign@type@field: new .....	20	removed \makefirstuc (now dealt with in \glentryfmt) .....	122
3.10a (2013-10-13)		\@Glspl@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	124
\@gls@keymap: new .....	71	change to using \glentryfmt style commands .....	125
\@gls@provide@newglossary: new ...	58	removed \makefirstuc (now dealt with in \glentryfmt) .....	125
\@gls@writedef: new .....	70	\@acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	356
\@glsdefaultplural: Obsolete .....	67	\@acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	355
\@glsnodesc: new .....	67	\@gls@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	121
\@print@glossary: Added providecommand code to aux file .....	187, 188	change to using \glentryfmt style commands .....	121
\gls@defglossaryentry: Changed to using \@gls@default@value .....	80	\@gls@noexpand@fields: Fixed bug expand replaced with noexpand ....	68
new .....	79	\@glsdisp: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	126
\gls.writedefhook: new .....	78	change to using \glentryfmt style commands .....	126
\makeglossaries: Added providecommand code to aux file ..	169	\@glspl@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	124
\new@glossaryentry: new .....	70	change to using \glentryfmt style commands .....	124
\ns@newglossary: added \@gls@provide@newglossary ....	59	General: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext .....	139–146
3.11a (2013-10-15)		changed to just use \Glsentrydescplural .....	132
\@ACRlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	357	changed to just use \glentrydescplural .....	132
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	355	changed to just use \Glsentrydesc .	131
\@Acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	356	changed to just use \glentrydesc .....	131, 132
\@Acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	355		
\@GLS@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	123		
change to using \glentryfmt style commands .....	123		
removed \MakeUppercase (now moved to \glentryfmt) .....	123		
\@GLSpl: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	125		
change to using \glentryfmt style commands .....	125		
removed \MakeUppercase as now dealt with in \glentryfmt .....	125		

changed to just use		<code>\glsdisplay</code> : obsoleted	105
<code>\Glsentryfirstplural</code>	130	<code>\glsdisplayfirst</code> : obsoleted	105
changed to just use		<code>\glsgenentryfmt</code> : new	100
<code>\glsentryfirstplural</code>	129, 130	<code>\glsgetgrouptitle</code> : Added check in	
changed to just use <code>\Glsentryfirst</code>	128	case non-Latin alphabet in use	207
changed to just use <code>\glsentryfirst</code>	128	<code>\gls glossarymark</code> : replaced	
changed to just use <code>\Glsentryname</code>	131	<code>\MakeUppercase</code> with	
changed to just use		<code>\mfirstucMakeUppercase</code>	39
<code>\glsentryname</code>	130, 131	<code>\glsnavigation</code> : switched to using	
changed to just use <code>\Glsentryplural</code>	129	<code>\@gls@getgrouptitle</code>	266
changed to just use <code>\glsentryplural</code>	129	<code>\ifglshasdesc</code> : replaced <code>\ifdefempty</code>	
changed to just use		with <code>\ifcsempy</code>	55
<code>\Glsentrysymbolplural</code>	134	<code>\ifglshaslong</code> : new	55
changed to just use		<code>\ifglshasshort</code> : new	56
<code>\glsentrysymbolplural</code>	134	<code>\ifglshassymbol</code> : replaced	
changed to just use <code>\Glsentrysymbol</code>	133	<code>\ifdefempty</code> with <code>\ifcsempy</code>	55
changed to just use <code>\glsentrysymbol</code>	133	<code>\ifglused</code> : replaced <code>\ifthenelse</code> with	
Changed to just use <code>\Glsentrytext</code>	128	<code>\ifbool</code>	53
changed to just use <code>\glsentrytext</code>	127	<code>\longnewglossaryentry</code> : new	79
changed to just use		<code>\ns@newglossary</code> : replaced	
<code>\Glsentryuseriii</code>	136	<code>\glsdisplay</code> and	
changed to just use		<code>\glsdisplayfirst</code> with	
<code>\glsentryuseriii</code>	136, 137	<code>\glsentryfmt</code>	59
changed to just use <code>\Glsentryuserii</code>	135	compatible-3.07: <code>cnew</code>	29
changed to just use		<code>\SetCustomDisplayStyle</code> : updated to	
<code>\glsentryuserii</code>	135, 136	use <code>\defglsglsentryfmt</code>	248
changed to just use <code>\Glsentryuseriv</code>	137	<code>\SetDefaultAcronymDisplayStyle</code> :	
changed to just use <code>\glsentryuseriv</code>	137	changed to use <code>\defglsglsentryfmt</code>	232
changed to just use <code>\Glsentryuseri</code>	135	<code>\SetDescriptionAcronymDisplayStyle</code> :	
changed to just use		updated to use <code>\defglsglsentryfmt</code>	238
<code>\glsentryuseri</code>	134, 135	<code>\SetDescriptionDUAAcronymDisplayStyle</code> :	
changed to just use <code>\Glsentryuservi</code>	139	updated to use <code>\defglsglsentryfmt</code>	236
changed to just use		<code>\SetDescriptionFootnoteAcronymDisplayStyle</code> :	
<code>\glsentryuservi</code>	138, 139	updated to use <code>\defglsglsentryfmt</code>	234
changed to just use <code>\Glsentryuserv</code>	138	<code>\SetDUADisplayStyle</code> : updated to use	
changed to just use <code>\glsentryuserv</code>	138	<code>\defglsglsentryfmt</code>	246
Now requires <code>textcase</code>	4	<code>\SetFootnoteAcronymDisplayStyle</code> :	
acronymlists: replaced		updated to use <code>\defglsglsentryfmt</code>	240
<code>\@addtoacronymlists</code> with		<code>\SetSmallAcronymDisplayStyle</code> :	
<code>\DeclareAcronymList</code>	17	updated to use <code>\defglsglsentryfmt</code>	243
<code>\defglsglsdisplay</code> : obsoleted	105	<code>\setupglossaries</code> : new	30
<code>\defglsglsdisplayfirst</code> : obsoleted	106	<code>\showglo long</code> : new	253
<code>\defglsglsentryfmt</code> : new	58	<code>\showglo short</code> : new	253
<code>\forglsglsentries</code> : replaced <code>\ifx</code> with		numbers: new	29
<code>\ifdefempty</code>	51	symbols: new	29
<code>\gls@assign@desc</code> : new	78	3.12a (2013-10-16)	
<code>\gls@defglossaryentry</code> : Fixed default		<code>\gls@defglossaryentry</code> : added	
counter if none supplied	83	<code>\glslabel</code>	79
<code>\gls@doentryfmt</code> : new	58	<code>\glsaddkey</code> : new	73

### 3.13a (2013-11-05)

\@gls@assign@symbol@field: changed to use \glssetnoexpandfield	20
\@gls@assign@symbolplural@field: changed to use \glssetnoexpandfield	20
\@gls@link: removed \relax	110
\@gls@notranslatorhook: new	24
\@gls@setupsort@standard: moved \@gls@santizesort to \glsprestandardsort	12
ucmark: added check for memoir	10
see: added \gls@checkseeallowed	64
\glossarysection: changed \glossarymark to \gls glossarymark	39
\glossarystyle: fixed bug caused by using \ifdef instead of \ifcsdef	209
\gls@assign@desc@field: changed to use \glssetnoexpandfield	20
\gls@assign@descplural@field: changed to use \glssetnoexpandfield	20
\gls@assign@name@field: changed to use \glssetnoexpandfield	20
\gls@assign@type@field: changed to use \glssetexpandfield	20
\gls@checkseeallowed: new	64
\glsaddallunused: set default to \@glo@types	156
\Glsentryfull: changed to use \acrfullformat	153
\glsentryfull: changed to use \acrfullformat	153
\Glsentryfullpl: changed to use \acrfullformat	154
\glsentryfullpl: changed to use \acrfullformat	154
\gls glossarymark: renamed \glossarymark to \gls glossarymark to avoid conflict with memoir	39
\glsprestandardsort: new	11
\glssetexpandfield: new	19
\glssetnoexpandfield: new	19
altsuper4colheader: switched to \tabularnewline	300
altsuper4colheaderborder: switched to \tabularnewline	301

long: switched to \tabularnewline	273
long3col: switched to \tabularnewline	274
long3colheader: switched to \tabularnewline	275
long3colheaderborder: switched to \tabularnewline	275
long4col: switched to \tabularnewline	276
long4colheader: switched to \tabularnewline	276
longheader: switched to \tabularnewline	274
longheaderborder: switched to \tabularnewline	274
\SetFootnoteAcronymDisplayStyle: fixed missing argument bug	241
super: switched to \tabularnewline	295
super3col: switched to \tabularnewline	297
super3colheader: switched to \tabularnewline	297
super4col: switched to \tabularnewline	298
super4colheader: switched to \tabularnewline	299
super4colheaderborder: switched to \tabularnewline	299
superheader: switched to \tabularnewline	296
superheaderborder: switched to \tabularnewline	296

### 3.14a (2013-11-12)

\@glswritefiles: renamed \glswritefiles to \@glswritefiles and used “savewrites” option to set \glswritefiles	175
General: new	257
acronyms: new	16
\gls@def glossaryentry: added check for existence of default glossary	80
set the default for firstplural to be the value of plural	83
xindy gloss: new	28
\longprovide glossaryentry: new	79
compatible-2.07: added check for 2.07 before setting 3.07 compatibility	29
notranslate: new	24

\provideglossaryentry:new .....	70	index:new .....	30
4.0 (2013-11-14)		\newacronymstyle:new .....	220
\gls@defglossaryentry: added check		long-sc-short:new .....	223
for first key .....	83	long-sc-short-desc:new .....	224
super: fixed typo in \subglossentry		long-short:new .....	221
(\glossentrydesc) .....	295	long-short-desc:new .....	224
4.01 (2013-11-16)		long-sm-short:new .....	223
General: fixed non-value options so that		long-sm-short-desc:new .....	225
they can be passed to document class .	8	long-sp-short-desc:new .....	224
\CustomAcronymFields: inserted		footnote:new .....	228
missing comma .....	248	footnote-desc:new .....	230
4.02 (2013-12-05)		footnote-sc:new .....	229
\@acrfull: now using \acrfullfmt ..	215	footnote-sc-desc:new .....	230
\@gls@indexdef:new .....	30	footnote-sm:new .....	230
\@gls@numbersdef:new .....	29	footnote-sm-desc:new .....	230
\@gls@symbolsdef:new .....	29	\setacronymstyle:new .....	220
General: Removed \acronymfont .	143–146	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new .....	216	Moved check for empty custom text to	
\Acrfullfmt:new .....	216	prevent unwanted parenthetical	
\acrfullfmt:new .....	215	material .....	238
\ACRfullplfmt:new .....	217	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new .....	217	Moved check for empty custom text to	
\acrfullplfmt:new .....	216	prevent unwanted parenthetical	
\acronymentry:new .....	219	material .....	234
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here .....	23	Moved check for empty custom text to	
sc-short-long:new .....	223	prevent unwanted parenthetical	
sc-short-long-desc:new .....	225	material .....	240
\Genacrfullformat:new .....	104	\SetGenericNewAcronym:new .....	218
\genacrfullformat:new .....	104	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new .....	219	Moved check for empty custom text to	
\Genplacrfullformat:new .....	105	prevent unwanted parenthetical	
\genplacrfullformat:new .....	105	material .....	243
\Glsentryfull: bug fix: added missing		dua:new .....	226
\acronymfont .....	153	dua-desc:new .....	228
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont .....	153	option .....	7
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont .....	154	\makeglossaries: made preamble only	171
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont .....	154	General: changed default to \@empty	
\glsgenacfmt:new .....	102	instead of \relax .....	29
\GlsUseAcrEntryDisplayStyle:new ...	221	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new .....	221	\@do@wrglossary: added	
short-long:new .....	222	\glsdetoklabel .....	180
short-long-desc:new .....	225	\@ACRlong: removed \glslabel	
xindynoglsnumbers:new .....	28	(defined in \@gls@link) .....	357
sm-short-long:new .....	223	\@ACRshort: removed \glslabel	
sm-short-long-desc:new .....	225	(defined in \@gls@link) .....	355

\@Acrlong: removed \glslabel (defined in \@gls@link) .....	356	\genplacrfullformat: redefined to use accessibility information .....	354
\@Acrshort: removed \glslabel (defined in \@gls@link) .....	355	\glossentryname: added \glsdetoklabel .....	204
\@GLS@: removed \glslabel (defined in \@gls@link) .....	123	\gls@defglossaryentry: added \glsdetoklabel .....	79
\@GLSpl: removed \glslabel (defined in \@gls@link) .....	125	replaced #1 with \@glo@label .....	81
\@Gls@: removed \glslabel (defined in \@gls@link) .....	122	replaced \ifthenelse with \ifdefequal .....	81
\@Gls@entry@field: new .....	147	\glsadd: added \glsdetoklabel ....	156
\@Glspl@: removed \glslabel (defined in \@gls@link) .....	124	\glsaddkey: switched to using \@gls@field@link .....	74
\@acrlong: removed \glslabel (defined in \@gls@link) .....	356	\glsdetoklabel: new .....	52
\@acrshort: removed \glslabel (defined in \@gls@link) .....	355	\glsdisplaynumberlist: added \glsdetoklabel .....	154
\@gls@: removed \glslabel (defined in \@gls@link) .....	121	\glsdoifexistsorwarn: new .....	53
\@gls@access@display: new .....	343	\glsentryaccess: switched to using \@gls@entry@field .....	341
\@gls@entry@field: new .....	147	\glsentrydescaccess: switched to using \@gls@entry@field .....	342
\@gls@fetchfield: new .....	72	\glsentrydescpluralaccess: switched to using \@gls@entry@field .....	342
\@gls@field@link: new .....	127	\glsentryfirstaccess: switched to using \@gls@entry@field .....	342
\@gls@link: added \glsdetoklabel .	110	\glsentryfirstplural: added \glsdetoklabel .....	150
moved \@gls@link@opts and \@gls@link@label to \@gls@link	110	\glsentrylongaccess: switched to using \@gls@entry@field .....	343
\@gls@writedef: added \glsdetoklabel .....	70	\glsentrylongpluralaccess: switched to using \@gls@entry@field .....	343
\@glsdisp: removed \glslabel (defined in \@gls@link) .....	126	\glsentrypluralaccess: switched to using \@gls@entry@field .....	342
\@glspl@: removed \glslabel (defined in \@gls@link) .....	124	\glsentryshortaccess: switched to using \@gls@entry@field .....	342
\@printglossary: added \glsdetoklabel .....	186	\glsentryshortpluralaccess: switched to using \@gls@entry@field .....	342
General: removed \glslabel (defined in \@gls@link) .....	139	\glsentrysymbolaccess: switched to using \@gls@entry@field .....	342
sc-short-long-desc: redefined to use accessibility information .....	361	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field .....	342
\compatibleglossentry: added \glsdetoklabel .....	337	\glsentrytextaccess: switched to using \@gls@entry@field .....	342
\compatiblesubglossentry: added \glsdetoklabel .....	338	\glsgenacfmt: redefined to use accessibility information .....	352
\Genacrfullformat: redefined to use accessibility information .....	354	\glsgenentryfmt: redefined to use accessibility information .....	349
\genacrfullformat: redefined to use accessibility information .....	354		

<code>\glshyperlink</code> : added		
<code>\glsdetoklabel</code> .....	155	
<code>\glslocalreset</code> : added		
<code>\glsdetoklabel</code> .....	88	
<code>\glslocalunset</code> : added		
<code>\glsdetoklabel</code> .....	89	
<code>\glsmoveentry</code> : added		
<code>\glsdetoklabel</code> .....	85	
replaced <code>\ifthenelse</code> with		
<code>\ifdefequal</code> .....	85	
<code>\glsrefentry</code> : added <code>\glsdetoklabel</code>	203	
<code>\glsreset</code> : added <code>\glsdetoklabel</code> ...	88	
<code>\glsseelist</code> : added <code>\expandafter</code>		
commands .....	183	
<code>\glsstepentry</code> : added		
<code>\glsdetoklabel</code> .....	202	
<code>\glsstepsubentry</code> : added		
<code>\glsdetoklabel</code> .....	202	
<code>\glsunset</code> : added <code>\glsdetoklabel</code> ...	88	
short-long: commented spurious EOL	223	
redefined to use accessibility		
information .....	359	
short-long-desc: redefined to use		
accessibility information .....	361	
<code>\ifglshdescsuppressed</code> : added		
<code>\glsdetoklabel</code> .....	55	
fixed typo .....	55	
<code>\ifglshentryexists</code> : added		
<code>\glsdetoklabel</code> .....	53	
<code>\ifglshaschildren</code> : added		
<code>\glsdetoklabel</code> .....	54	
<code>\ifglshasdesc</code> : added		
<code>\glsdetoklabel</code> .....	55	
<code>\ifglshasfield</code> : new .....	56	
<code>\ifglshaslong</code> : added		
<code>\glsdetoklabel</code> .....	55	
<code>\ifglshasparent</code> : added		
<code>\glsdetoklabel</code> .....	55	
<code>\ifglshasshort</code> : added		
<code>\glsdetoklabel</code> .....	56	
<code>\ifglshassymbol</code> : added		
<code>\glsdetoklabel</code> .....	55	
replaced <code>\ifcempty</code> with		
<code>\ifdefempty</code> and replaced <code>\ifx</code> with		
<code>\ifdefequal</code> .....	55	
<code>\ifglshused</code> : added <code>\glsdetoklabel</code> ..	53	
sm-short-long-desc: redefined to use		
accessibility information .....	361	
long-sc-short-desc: redefined to use		
accessibility information .....	360	
long-short: redefined to use		
accessibility information .....	358	
long-short-desc: redefined to use		
accessibility information .....	360	
long-sm-short-desc: redefined to use		
accessibility information .....	360	
footnote: redefined to use accessibility		
information .....	364	
footnote-desc: redefined to use		
accessibility information .....	366	
footnote-sc: redefined to use		
accessibility information .....	366	
footnote-sc-desc: redefined to use		
accessibility information .....	367	
footnote-sm: redefined to use		
accessibility information .....	366	
footnote-sm-desc: redefined to use		
accessibility information .....	367	
<code>\renewacronymstyle</code> : new .....	220	
<code>\showglocounter</code> : added		
<code>\glsdetoklabel</code> .....	251	
<code>\showglodesc</code> : added <code>\glsdetoklabel</code>	252	
<code>\showglodescaccess</code> : added		
<code>\glsdetoklabel</code> .....	373	
<code>\showglodescplural</code> : added		
<code>\glsdetoklabel</code> .....	253	
<code>\showglodescpluralaccess</code> : added		
<code>\glsdetoklabel</code> .....	373	
<code>\showglofirst</code> : added		
<code>\glsdetoklabel</code> .....	251	
<code>\showglofirstaccess</code> : added		
<code>\glsdetoklabel</code> .....	373	
<code>\showglofirstpl</code> : added		
<code>\glsdetoklabel</code> .....	251	
<code>\showglofirstpluralaccess</code> : added		
<code>\glsdetoklabel</code> .....	373	
<code>\showgloflag</code> : added <code>\glsdetoklabel</code>	254	
<code>\showgloindex</code> : added		
<code>\glsdetoklabel</code> .....	254	
<code>\showglolevel</code> : added		
<code>\glsdetoklabel</code> .....	250	
<code>\showglolong</code> : added <code>\glsdetoklabel</code>	253	
<code>\showglolongaccess</code> : added		
<code>\glsdetoklabel</code> .....	374	
<code>\showglolongpluralaccess</code> : added		
<code>\glsdetoklabel</code> .....	374	
<code>\showglongname</code> : added <code>\glsdetoklabel</code>	252	



\showglonameaccess: added	4.04 (2014-03-04)	
\glsdetoklabel .....	373	\@gls@getcounterprefix: added
\showgloparent: added		warning if no prefix can be formed . 182
\glsdetoklabel .....	250	4.04 (2014-03-06)
\showgloplural: added		\@gls@noidx@nosanitizesort: new . 21
\glsdetoklabel .....	250	\@gls@noidx@sanitizesort: new ... 20
\showglopluralaccess: added		\@gls@nosanitizesort: new ..... 20
\glsdetoklabel .....	373	\@gls@sanitizesort: new ..... 20
\showgloshort: added		\@glo@addchildren: new ..... 189
\glsdetoklabel .....	253	\@glo@do@sortentries: new ..... 190
\showgloshortaccess: added		\@glo@grabfirst: new ..... 195
\glsdetoklabel .....	374	\@glo@sortedinsert: new ..... 190
\showgloshortpluralaccess: added		\@glo@sortentries: new ..... 188
\glsdetoklabel .....	374	\@glo@sorthandler@case: new ..... 191
\showglosort: added \glsdetoklabel	253	\@glo@sorthandler@letter: new ... 191
\showglosymbol: added		\@glo@sorthandler@nocase: new ... 191
\glsdetoklabel .....	253	\@glo@sorthandler@word: new ..... 190
\showglosymbolaccess: added		\@glo@sortmacro@case: new ..... 192
\glsdetoklabel .....	373	\@glo@sortmacro@def: new ..... 193
\showglosymbolplural: added		\@glo@sortmacro@def@do: new ..... 193
\glsdetoklabel .....	253	\@glo@sortmacro@letter: new ..... 192
\showglosymbolpluralaccess: added		\@glo@sortmacro@nocase: new ..... 193
\glsdetoklabel .....	373	\@glo@sortmacro@standard: new ... 192
\showglotext: added \glsdetoklabel	250	\@glo@sortmacro@use: new ..... 193
\showglotextaccess: added		\@glo@sortmacro@word: new ..... 191
\glsdetoklabel .....	373	\@gls@noidx@do: new ..... 195
\showglotype: added \glsdetoklabel	251	\@gls@noidx@getgrouptitle: new .. 208
\showglouser: added		\@gls@noref@warn: new ..... 175
\glsdetoklabel .....	251	\@gls@reference: new ..... 197
\showglouserii: added		\@gls@warnonglossdefined: new .... 19
\glsdetoklabel .....	251	\@gls@warnontheGLOSSdefined: new . 19
\showglouseriii: added		\@no@makeglossaries: new ..... 175
\glsdetoklabel .....	252	\@print@glossary: new ..... 187
\showglouseriv: added		\@print@noidx@glossary: new ..... 194
\glsdetoklabel .....	252	\@print@gloss@setsort: new ..... 185
\showglouserv: added		\@print@glossary: new ..... 185
\glsdetoklabel .....	252	General: added sort key to printgloss
\showglouservi: added		group ..... 200
\glsdetoklabel .....	252	\compatibleglossentry: changed
dua: fixed bug in \acrfullfmt .....	227	\newcommand to \def as is may or
fixed bug in \Acrrfullplfmt .....	227	may not be defined ..... 337
fixed bug in \acrfullplfmt .....	227	\compatiblesubglossentry: changed
redefined to use accessibility		\newcommand to \def as is may or
information .....	362	may not be defined ..... 338
dua-desc: commented spurious EOL ..	228	\defglsdisplayfirst: fixed unwanted
redefined to use accessibility		space ..... 106
information .....	364	\glo@grabfirst: new ..... 194
		\gls@defglossaryentry: replaced \ifx
		with \ifdefvoid ..... 84

\glsnoidxdisplayloc: new .....	197	\@ACRshort: added	
\glsnoidxdisplayloclisthandler:		\do@gl@link@checkfirsthyper	355
new .....	197	\@Acrlong: added	
\glsnoidxloclist: new .....	196	\do@gl@link@checkfirsthyper	356
\glsnoidxloclisthandler: new ....	197	\@Acrshort: added	
\glsnoidxstripaccents: new .....	21	\do@gl@link@checkfirsthyper	355
alttree: moved hangindent and		\@GLS@: moved \glsifhyper .....	123
parindent assignments outside level		moved check for first use to	
test .....	314	\@gl@link .....	123
\makeglossaries: Moved definition of		\@GLSpl: moved \glsifhyper .....	125
\glswrite to \makeglossaries ..	169	moved check for first use to	
\makenoidxglossaries: new .....	171	\@gl@link .....	125
\printglossary: changed to use new		\@Gls@: moved \glsifhyper .....	122
\@printglossary .....	184	moved check for first use to	
\printnoidxglossaries: new .....	185	\@gl@link .....	122
\printnoidxglossary: new .....	185	\@Glspl@: moved \glsifhyper .....	125
\showgloclist: new .....	254	moved check for first use to	
\warn@noprintglossary: Activate		\@gl@link .....	125
warning in \makeglossaries ....	184	\@acrlong: added	
\writeist: checked for definition of		\do@gl@link@checkfirsthyper	356
\glswrite .....	158, 162	\@acrshort: added	
4.06 (2014-03-12)		\do@gl@link@checkfirsthyper	354
\@GLS@: added \glsifhyper .....	123	\@closegls: new .....	168
\@GLSpl: added \glsifhyper .....	125	\@gls@: moved \glsifhyper .....	121
\@Gls@: added \glsifhyper .....	122	moved check for first use to	
\@Glspl@: added \glsifhyper .....	125	\@gl@link .....	121
\@gls@: added \glsifhyper .....	121	\@gls@automake: new .....	168
\@gls@numbersdef: added hook to set		\@gls@doautomake: new .....	28
toc title .....	30	\@gls@field@link: added assignment	
\@gls@symbolsdef: added hook to set		of	
toc title .....	29	\do@gl@link@checkfirsthyper	127
\@glsdisp: added \glsifhyper .....	126	\@gls@forbidtexext: new .....	58
\@Glspl@: added \glsifhyper .....	124	\@gls@hyp@opt: new .....	107
General: added \glsifhyper ....	139–146	\@gl@link: removed redundancy ....	110
acronym: added hook to set toc title ....	15	renamed \gls@type to \glstype ...	110
acronyms: added hook to set toc title ...	16	\@gl@link@checkfirsthyper: new .	109
\glsdefmain: added hook to set toc title	14	\@glsdisp: moved \glsifhyper .....	126
4.07 (2014-04-04)		moved check for first use to	
\@glossarysection: added optional		\@gl@link .....	126
argument when using unstarred		\@Glspl@: moved \glsifhyper .....	124
version .....	40	moved check for first use to	
\@gls@noidx@do: added \global in case		\@gl@link .....	124
it's used in a tabular-like style .....	195	\@ignored@glossaries: new .....	61
\Acrfullplfmt: fixed no case change		General: added entrycounter option to	
bug .....	217	printgloss family .....	199
\glsletentryfield: new .....	147	added nopostdot option to	
4.08 (2014-07-30)		printgloss family .....	199
\@ACRlong: added		added subentrycounter option to	
\do@gl@link@checkfirsthyper	356	printgloss family .....	200

explicitly initialise hyper key .....	107	removed \@sGLSuseriii .....	136
moved \glsifhyper .....	139–146	removed \@sGlsuseriii .....	135
removed \@sACRlongpl .....	146	removed \@sglsuseriii .....	135
removed \@sAcrlongpl .....	145	removed \@sGLSuseriv .....	137
removed \@sacrlongpl .....	145	removed \@sGlsuseriv .....	137
removed \@sACRlong .....	144	removed \@sglsuseriv .....	137
removed \@sAcrlong .....	143	removed \@sGLSuseri .....	135
removed \@sacrlong .....	143	removed \@sGlsuseri .....	135
removed \@sACRshortpl .....	142	removed \@sglsuseri .....	134
removed \@sAcrshortpl .....	142	removed \@sGLSservi .....	139
removed \@sacrshortpl .....	141	removed \@sGlsservi .....	139
removed \@sACRshort .....	140	removed \@sglsuseri .....	138
removed \@sAcrshort .....	140	removed \@sGLSserv .....	138
removed \@sacrshort .....	139	removed \@sGlsuseri .....	138
removed \@sgls@link .....	108	removed \@sglsuseri .....	138
removed \@sGLSdescplural .....	132	removed \@sGLS .....	123
removed \@sGlsdescplural .....	132	removed \@sGls .....	122
removed \@sglsdescplural .....	132	removed \@sgls .....	121
removed \@sGLSdesc .....	132	removed \@thirdofthree (defined in	
removed \@sGlsdesc .....	131	kernel) .....	121
removed \@sglsdesc .....	131	removed sPGLS .....	262
removed \@sglsdisp .....	126	removed sPgls .....	260
removed \@sGLSfirstplural .....	130	removed spgls .....	259
removed \@sGlsfirstplural .....	130	removed sPGLSpl .....	262
removed \@sglsfirstplural .....	129	removed sPglspl .....	261
removed \@sGLSfirst .....	128	removed spglspl .....	260
removed \@sGlsfirst .....	128	\ACRfull: removed \s@ACRfull .....	216
removed \@sglsfirst .....	128	switched to using \@gls@hyp@opt ..	216
removed \@sGLSname .....	131	\Acrfull: removed \@sAcrfull .....	215
removed \@sGlsname .....	130	switched to using \@gls@hyp@opt ..	215
removed \@sglsname .....	130	\acrfull: removed \@sacrfull .....	215
removed \@sGLSplural .....	129	switched to using \@gls@hyp@opt ..	215
removed \@sGlsplural .....	129	\ACRfullpl: removed \s@ACRfullpl ..	217
removed \@sglsplural .....	129	switched to using \@gls@hyp@opt ..	217
removed \@sGLSpl .....	125	\Acrfullpl: removed \s@Acrfullpl ..	217
removed \@sGlspl .....	124	switched to using \@gls@hyp@opt ..	217
removed \@sglspl .....	123	\acrfullpl: removed \s@acrfullpl ..	216
removed \@sGLSsymbolplural .....	134	switched to using \@gls@hyp@opt ..	216
removed \@sGlsymbolplural .....	134	\ACRlong: switched to using	
removed \@sglsymbolplural .....	133	\@gls@hyp@opt .....	144
removed \@sGLSsymbol .....	133	\Acrlong: switched to using	
removed \@sGlsymbol .....	133	\@gls@hyp@opt .....	143
removed \@sglsymbol .....	133	\acrlong: switched to using	
removed \@sGLStext .....	127	\@gls@hyp@opt .....	143
removed \@sGlstext .....	128	\ACRlongpl: switched to using	
removed \@sglstext .....	127	\@gls@hyp@opt .....	146
removed \@sGLSuseriii .....	136	\Acrlongpl: switched to using	
removed \@sGlsuseriii .....	136	\@gls@hyp@opt .....	145
removed \@sglsuseriii .....	136		

\acrlongpl: switched to using \@gls@hyp@opt .....	145	\GLSfirst: switched to using \@gls@hyp@opt .....	128
\ACRshort: switched to using \@gls@hyp@opt .....	140	\Glsfirst: switched to using \@gls@hyp@opt .....	128
\Acrshort: switched to using \@gls@hyp@opt .....	140	\glsfirst: switched to using \@gls@hyp@opt .....	128
\acrshort: switched to using \@gls@hyp@opt .....	139	\GLSfirstplural: switched to using \@gls@hyp@opt .....	130
\ACRshortpl: switched to using \@gls@hyp@opt .....	142	\Glsfirstplural: switched to using \@gls@hyp@opt .....	130
\Acrshortpl: switched to using \@gls@hyp@opt .....	141	\glsfirstplural: switched to using \@gls@hyp@opt .....	129
\acrshortpl: switched to using \@gls@hyp@opt .....	141	\glsifhyper: deprecated .....	107
\forallacronyms: new .....	51	\glslink: switched to using \@gls@hyp@opt .....	108
\GLS: switched to using \@gls@hyp@opt	123	\glslinkcheckfirsthyperhook: new	109
\Gls: switched to using \@gls@hyp@opt	122	\glslinkvar: new .....	107
\gls: switched to using \@gls@hyp@opt	121	\GLSname: switched to using \@gls@hyp@opt .....	131
\gls@defglossaryentry: added check for ignored glossary .....	81	\Glsname: switched to using \@gls@hyp@opt .....	130
\gls@istfilebase: new .....	36	\glsname: switched to using \@gls@hyp@opt .....	130
\glsaddkey: removed \@sGLS@user@<key> .....	75	\GLSpl: switched to using \@gls@hyp@opt .....	125
removed \@sGls@user@<key> .....	75	\Glspl: switched to using \@gls@hyp@opt .....	124
removed \@sgls@user@<key> .....	74	\glspl: switched to using \@gls@hyp@opt .....	123
switched to using \@gls@hyp@opt	74, 75	\GLSplural: switched to using \@gls@hyp@opt .....	129
\GLSdesc: switched to using \@gls@hyp@opt .....	131	\Glsplural: switched to using \@gls@hyp@opt .....	129
\Glsdesc: switched to using \@gls@hyp@opt .....	131	\glsplural: switched to using \@gls@hyp@opt .....	129
\glsdesc: switched to using \@gls@hyp@opt .....	131	\glsspace: new .....	215
\GLSdescplural: switched to using \@gls@hyp@opt .....	132	\GLSsymbol: switched to using \@gls@hyp@opt .....	133
\Glsdescplural: switched to using \@gls@hyp@opt .....	132	\Glsymbol: switched to using \@gls@hyp@opt .....	133
\glsdescplural: switched to using \@gls@hyp@opt .....	132	\glssymbol: switched to using \@gls@hyp@opt .....	133
\glsdisablehyper: added \KV@glslink@hyperfalse to definition .....	120	\GLSsymbolplural: switched to using \@gls@hyp@opt .....	134
\glsdisp: switched to using \@gls@hyp@opt .....	126	\Glsymbolplural: switched to using \@gls@hyp@opt .....	134
\glsdohyperlink: new .....	120	\glssymbolplural: switched to using \@gls@hyp@opt .....	133
\glsdohypertarget: new .....	119		
\glsenablehyper: added \KV@glslink@hypertrue to definition .....	120		

\GLStext: switched to using \@gls@hyp@opt .....	127	\ns@newglossary: added \@gls@hyp@opt: new .....	59
\Glstext: switched to using \@gls@hyp@opt .....	127	\p@gls@hyp@opt: new .....	108
\glstext: switched to using \@gls@hyp@opt .....	127	\PGLS: changed to use \@gls@hyp@opt	262
\glstreenamefmt: new .....	307	\Pgls: changed to use \@gls@hyp@opt	260
\GLSuseri: switched to using \@gls@hyp@opt .....	135	\pgls: changed to use \@gls@hyp@opt	259
\Glsuseri: switched to using \@gls@hyp@opt .....	135	\PGLSpl: changed to use \@gls@hyp@opt .....	262
\glsuseri: switched to using \@gls@hyp@opt .....	134	\Pglspl: changed to use \@gls@hyp@opt .....	261
\GLSuserii: switched to using \@gls@hyp@opt .....	136	\pglspl: changed to use \@gls@hyp@opt .....	260
\Glsuserii: switched to using \@gls@hyp@opt .....	135	\s@gls@hyp@opt: new .....	108
\glsuserii: switched to using \@gls@hyp@opt .....	135	\s@newglossary: new .....	59
\GLSuseriii: switched to using \@gls@hyp@opt .....	136	automake: new .....	28
\Glsuseriii: switched to using \@gls@hyp@opt .....	136	4.09 (2014-08-12)	
\glsuseriii: switched to using \@gls@hyp@opt .....	136	\glsaddkey: fixed bug in user commands	74
\GLSuseriv: switched to using \@gls@hyp@opt .....	137	4.10 (2014-08-27)	
\Glsuseriv: switched to using \@gls@hyp@opt .....	137	\@Gls@acrentryname: new .....	148
\glsuseriv: switched to using \@gls@hyp@opt .....	137	\@Gls@entryname: new .....	148
\GLSuserv: switched to using \@gls@hyp@opt .....	138	\@gls@glossary: Renamed \@glossary to \@gls@glossary .....	177
\Glsuserv: switched to using \@gls@hyp@opt .....	138	\glspercentchar: new .....	157
\glsuserv: switched to using \@gls@hyp@opt .....	137	\glstildechar: new .....	157
\GLSuservi: switched to using \@gls@hyp@opt .....	139	alttree: moved space after symbol	314, 315
\Glsuservi: switched to using \@gls@hyp@opt .....	139	4.11 (2014-09-01)	
\ifignoredglossary: new .....	61	\@do@wrglossary: added hook .....	180
altlongragged4col: fixed bug that displayed description instead of symbol .....	287	sanitize: none option .....	23
\newglossary: added starred version ..	59	\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	177
\newignoredglossary: new .....	61	\glsaddprotectedpagefmt: new ....	179
		\glsbackslash: new .....	157
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute .....	45
		\@gls@adjustmode: new .....	156
		\@gls@notranslatorhook: removed ...	24
		\@gls@toc: added \protect to \numberline .....	42
		\@gls@usetranslator: new .....	24
		\glsacrpluralsuffix: new .....	33
		\glsadd: added check for vertical mode	156
		\glsaddallunused: replaced @gobble with glsignore .....	156
		\glsifusedtranslatordict: new ....	24
		\glsignore: new .....	157
		\glsupacrpluralsuffix: new .....	33
		\ProvidesGlossariesLang: new .....	33

\RequireGlossariesLang: new	33	4.16 (2015-07-08)	
4.13 (2015-02-03)		\@ACRlong: added \glspostlinkhook	357
\indexspace: new	269, 289, 307	\@ACRshort: added \glspostlinkhook	356
4.14 (2015-02-28)		\@Acrlong: added \glspostlinkhook	356
\@glslocalreset: new	89	\@Acrshort: added \glspostlinkhook	355
\@glslocalunset: new	89	\@GLS@: added \glspostlinkhook	123
\@glsreset: new	90	\@GLSpl: added \glspostlinkhook	126
\@glsunset: new	89	\@Gls@: added \glspostlinkhook	123
\@newglossaryentry@defcounters:		\@GLspl@: added \glspostlinkhook	125
new	91	\@acrlong: added \glspostlinkhook	356
\cGls: new	94	\@acrshort: added \glspostlinkhook	355
\cGls@: new	94	\@gls@: added \glspostlinkhook	122
\cGlspl@: new	95	\@gls@link: added	
\cgls: new	94	\glspostlinkhook	109
\cgls@: new	94	\@gls@field@link: added	
\cglspl: new	95	\glspostlinkhook	127
\cglspl@: new	95	\@gls@link: moved definition of	
\@gls@entry@count: new	93	\glsifhyperon outside of this	
\@gls@increment@currcount: new	93	macro	110
\@gls@local@increment@currcount:		\@glsdisp: added \glspostlinkhook	126
new	93	\@glspl@: added \glspostlinkhook	124
\@gls@write@entrycounts: new	93	General: added \glspostlinkhook	140–146
\@glslocalreset: new	89	\glsacspace: new	222
\@glslocalunset: new	89	\glsadd: changed \@do@wrglossary to	
\@glsreset: new	89	\@do@wrglossary	156
\@glsunset: new	89	\glsfielddef: new	77
\@newglossaryentry@defcounters:		\glsfieldedef: new	76
new	85	\glsfieldfetch: new	77
\cGls: new	94	\glsfieldgdef: new	76
\cgls: new	93	\glsfieldxdef: new	75
\cGlsformat: new	94	\glsifhyperon: moved definition of	
\cglsformat: new	94	\glsifhyperon	109
\cGlspl: new	95	\glslinkpostsetkeys: new	109
\cglspl: new	94	\glspostlinkhook: new	109
\cGlsplformat: new	95	\glswriteentry: new	178
\cglsplformat: new	95	\ifglsfieldcseq: new	78
\@gls@defdocnewglossaryentry: new	70	\ifglsfielddefeq: new	78
\@glsenableentrycount: new	91	\ifglsfieldefeq: new	77
\glslocalreset: switched to		long-sp-short: new	221
\@glslocalreset	88	\showglofield: new	254
\glslocalunset: switched to		4.18 (2015-09-09)	
\@glslocalunset	89	General: split mfirstuc into separate	
\glsreset: switched to \@glsreset	88	bundle	4
\glsunset: switched to \@glsunset	88	4.19 (2015-10-31)	
4.15 (2015-03-16)		\glstreenamibox: new	313
General: bug fix replaced \@glo@type		4.19 (2015-11-22)	
with \glstype	146	\@gls@link@nocheckfirsthyper: new	127
4.16 (2015-06-18)		\@gls@preglossaryhook: new	185
\glsaddstoragekey: new	72		

\@printglossary: added		\glslistgroupheaderfmt: new	269
\@gls@preglossaryhook	187	\glslistnavigationitem: new	269
\do@glsglisablehyperinlist: new	109	\glstreegroupheaderfmt: new	307
\doifglossarynoexistsordo: new	54	\glstreenavigationfmt: new	307
\gls@gobbleopt: new	58	\ifglswrallowprimitivemods: new	179
\glsdoifexistsordo: new	54	list: fixed missing space before	
4.20 (2015-11-30)		description	269
\@gls@link: added		long: fixed typo in \glossentrydesc	273
\@gls@setdefault@glslink@opts	110	super4col: fixed bug in \glossentry	298
added \glsdonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed	110	\glscurrentfieldvalue: new	57
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new	109	\glscurrentfieldvalue	56, 57
\gls@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	64	nogroupskip changed	287
\glsdonohyperlink: new	120	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	306
\@printglossary: warn if no style has		long: check for nogroupskip changed	273
been set	185	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	279
assignment	139–146	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed	275
already set	209	long3col-booktabs: check for	
\glsLTpenaltycheck: new	282	nogroupskip changed	280
\glspatchLToutput: new	282	long4col: check for nogroupskip	
\glspenaltygroupskip: new	282	changed	276
altlong4col-booktabs: new	280	long4col-booktabs: check for	
altlongragged4col-booktabs: new	281	nogroupskip changed	280
long-booktabs: new	279	longragged: check for nogroupskip	
long3col-booktabs: new	279	changed	284
long4col-booktabs: new	280	longragged3col: check for nogroupskip	
longragged-booktabs: new	281	changed	285
longragged3col-booktabs: new	281	super: check for nogroupskip changed	295
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set	209	changed	297
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@wrglossary: added check for		changed	299
\@arabic	179	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	302
modifications and added arabic case	180	superragged3col: check for	
mcolalttreespannav: new	294	nogroupskip changed	304
mcolindexspannav: new	290	4.24 (2016-05-27)	
mcoltreenonamespannav: new	292	\@gls@extramakeindexopts: new	167
mcoltreespannav: new	291	\@gls@glossary: added check for debug	
\gls@arabicpage: new	178	mode	177
\gls@protected@pagefmts: added		\@gls@see@noindex: new	6
arabic to list	178	debug: new	5
\glsentrytitlecase: new	151	seenoinindex: new	6
\glsfindwidesttoplevelname: new	313	\glsnomakeindexwarning: new	42

\GlsSetQuote: new .....	164	\glsnavhyperlinkname: new .....	264
\GlsSetWriteIstHook: new .....	164	4.30 (2017-06-11)	
4.25 (2016-06-09)		\@glo@autosee: new .....	85
\@gls@enablesavenonumberlist: new	65	\@glo@autoseehook: new .....	85
\@gls@initnonumberlist: new .....	65	\@glo@check@sortallowed: new .....	11
\@gls@savenonumberlist: new .....	65	\@gls@noidx@do: letter group	
4.26 (2016-10-12)		assignment made global .....	196
\@glossary@default@style: added		\@gls@setupsort@def: added check for	
check for classicthesis .....	8	register .....	12
mcolindex: replaced \@idxitem with		\@gls@setupsort@none: new .....	14
\glstreeitem .....	289	\@xdycrossrefhook: new .....	47
mcolindexspannav: replaced \@idxitem		\@xdylocationclassorder: bug fix:	
with \glstreeitem .....	290	changed \edef to \def .....	48
\glstreechildpredesc: new .....	308	\glosortentrieswarning: new .....	18
\glstreeitem: new .....	307	\gls@set@xr@key: new .....	64
\glstreepredesc: new .....	308	\gls@xr@key: new .....	64
\glstreesubitem: new .....	308	\GlsAddXdyLocation: bug fix: changed	
\glstreesubsubitem: new .....	308	#1 to #2 .....	48
4.28 (2017-01-07)		\glsnoidxstripaccents: added \a ...	21
\glspatchtabularx: new .....	88	added \TH, \dh and \DH .....	22
4.29 (2017-01-19)		4.31 (2017-08-10)	
\@gls@noidx@do: current letter group		nolist: added check for “list” style .....	9
assignment made global .....	196	4.31 (2017-09-10)	
\@print@noidx@glossary: moved		style: changed \renewcommand to \def .	8
definition of		4.32 (2017-08-24)	
\@gls@currentlettergroup outside		\@glsnavhypertarget: new .....	264
of theglossary environment .....	194	\@glsshowtarget: new .....	6
General: added check for		\glsshowtarget: new .....	6
\@glsxtr@doaccsupp .....	337		



# 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	
\!	116
\"	21, 113, 115, 116, 118
\#	160
\%	157, 163, 321, 322
\&	33, 155
\'	21
\.	10, 21
\=	21
\?	113, 115, 165
\@@delimN	211
\@@do@wrglossary	172, 180
\@@do@wrglossary	156, 178
\@@glo@assign@sortkey	172
\@@glo@list	52
\@@glo@sort	21
\@@glo@type	185
\@@glossarysec	7, 40, 41
\@@glossaryseclabel	7, 8, 41, 198, 199
\@@glossarysecstar	7, 8, 40, 41, 198, 199
\@@gls@checkactual	118
\@@gls@checkbar	117
\@@gls@checkescactual	115
\@@gls@checkescbar	116
\@@gls@checkesclevel	116, 117
\@@gls@checkescquote	115, 166
\@@gls@checklevel	117, 118
\@@gls@checkquote	114, 164, 165
\@@gls@default@entryfmt	96, 105, 106
\@@gls@expand@field	19, 68, 69, 72, 73, 233, 235, 237, 239, 242, 244–246, 368–372
\@@gls@extramakeindexopts	164, 169
\@@gls@fixbraces	182
\@@gls@noexpand@field	19, 67, 68
\@@gls@noidx@no@sanitizesort	21
\@@gls@noidx@nosanitizesort	174
\@@gls@nosanitizesort	20, 174
\@@gls@sanitizesort	20, 174
\@@gls@xdycheckbackslash	119
\@@gls@xdycheckquote	118, 119
\@@gls@localreset	89, 92
\@@gls@localunset	89, 92
\@@gls@reset	89, 92
\@@gls@unset	89, 91
\@@newglossaryentry@defcounters	91
\@@this@glo@	52
\@ACRfull	216
\@ACRfullpl	217
\@ACRlong	144, 216
\@ACRlongpl	146, 217
\@ACRshort	140, 216
\@ACRshortpl	142, 217
\@Acrfull	215
\@Acrfullpl	217
\@Acrlong	143, 216
\@Acrlongpl	145, 217
\@Acrshort	140
\@Acrshortpl	142
\@Alph	178–180
\@GLS	123
\@GLS@	123, 262
\@GLSdesc	131, 132
\@GLSdesc@	132
\@GLSdescplural	132
\@GLSdescplural@	132
\@GLSfirst	128
\@GLSfirst@	128
\@GLSfirstplural	130
\@GLSfirstplural@	130
\@GLSname	131
\@GLSname@	131
\@GLSpl	125
\@GLSpl@	125, 263
\@GLSplural	129

\@GLSplural@	129	\@Glsuseriii@	136
\@GLSsymbol	133	\@Glsuseriv	137
\@GLSsymbol@	133	\@Glsuseriv@	137
\@GLSsymbolplural	134	\@Glsuserv	138
\@GLSsymbolplural@	134	\@Glsuserv@	138
\@GLStext	127	\@Glsuservi	139
\@GLStext@	127	\@Glsuservi@	139
\@GLSuseri	135	\@Mi	282
\@GLSuseri@	135	\@PGLS	262
\@GLSuserii	136	\@PGLS@	262
\@GLSuserii@	136	\@PGLSpl	262
\@GLSuseriii	136	\@PGLSpl@	262
\@GLSuseriii@	136, 137	\@Pgls	260
\@GLSuseriv	137	\@Pgls@	260
\@GLSuseriv@	137	\@Pglspl	261
\@GLSuserv	138	\@Pglspl@	261
\@GLSuserv@	138	\@Roman	178–180
\@GLSuservi	139	\@acrfull	215
\@GLSuservi@	139	\@acrfullpl	216
\@Gls	122	\@acrlong	143, 215
\@Gls@	92, 94, 122, 261	\@acrlongpl	145, 216
\@Gls@acrentryname	218	\@acrshort	139, 215, 216
\@Gls@entry@field	74, 148–153	\@acrshortpl	141, 216, 217
\@Gls@entryname	148, 218	\@addtoacronymlists	16
\@Glsdesc	131	\@after	17
\@Glsdesc@	131	\@afterheading	270, 325
\@Glsdescplural	132	\@alph	178–180
\@Glsdescplural@	132	\@arabic	178–180
\@Glsfirst	128	\@auxout	58,
\@Glsfirst@	128		59, 93, 169, 170, 172, 175, 184, 187, 188, 265
\@Glsfirstplural	130	\@backslashchar	113, 119
\@Glsfirstplural@	130	\@before	17
\@Glsname	130	\@bsphack	177
\@Glsname@	130, 131	\@cGls	94
\@Glspl	124	\@cGls@	92, 94
\@Glspl@	92, 95, 124, 261, 262	\@cGlspl	95
\@Glsplural	129	\@cGlspl@	92, 95
\@Glsplural@	129	\@cclv	282, 283
\@Glsymbol	133	\@cgls	93
\@Glsymbol@	133	\@cgls@	92, 94
\@Glsymbolplural	134	\@cglspl	94
\@Glsymbolplural@	134	\@cglspl@	92, 95
\@Glstext	127, 128	\@chapter	31
\@Glstext@	128	\@classoptionslist	30
\@Glsuseri	135	\@closegls	168, 169
\@Glsuseri@	135	\@colht	282
\@Glsuserii	135	\@colroom	282, 283
\@Glsuserii@	135	\@currentlabelname	8, 199
\@Glsuseriii	136	\@curroptions	30

\@declaredoptions .....	30	\@glo@counter .....	64, 80, 83
\@delimN .....	211	\@glo@counterprefix .....	175, 180–182, 209, 212
\@delimR .....	211	\@glo@default@sorttype ..	11, 172, 191–193
\@disable@onlypremakeg .....	170	\@glo@defaultcounter .....	83
\@disable@premakecs .....	32	\@glo@desc .....	62, 78, 79, 81, 84
\@disabled@gl\$addxdycounters .....	45	\@glo@descaccess .....	339–341
\@do@addcounter .....	43	\@glo@descplural .....	62, 78, 79
\@do@auxoutstuff .....	187, 188	\@glo@descpluralaccess .....	339–341
\@do@glossentry .....	204, 337, 338	\@glo@do@sortentries .....	189
\@do@gl\$@getcounterprefix .....	180	\@glo@entry .....	156
\@do@gl\$@islistofacronyms .....	17	\@glo@entryprefix .....	257
\@do@gl\$see .....	85	\@glo@entryprefixfirst .....	257
\@do@ifinlist .....	43	\@glo@entryprefixfirstplural ..	257, 258
\@do@newglossaryentry .....	218, 219, 233, 235, 237, 239–242, 244–249, 368–372	\@glo@entryprefixplural .....	257
\@do@seeglossary .....	172, 183	\@glo@esclabel .....	86, 87
\@do@subglossentry .....	206, 338	\@glo@etext .....	97, 99
\@do@wrglossary .....	110	\@glo@first .....	63, 80, 83, 84, 244, 372
\@do@writeaux@info .....	184	\@glo@firstaccess .....	338, 340, 341
\@ehc .....	282	\@glo@firstplural .....	63, 80, 83, 372
\@empty .....	13, 16, 28–30, 32, 43, 44, 48, 50, 51, 81, 86, 87, 111, 112, 121–125, 139–146, 158, 161, 163, 168, 169, 176, 177, 181, 182, 199, 201, 209, 234, 236, 238, 240– 243, 245, 247, 249, 319, 321, 323, 355–357	\@glo@firstpluralaccess .....	339–341
\@end@fixbraces .....	182, 183	\@glo@grabfirst .....	195
\@end@fortrue .....	25, 54, 72, 265	\@glo@label .....	67, 73, 74, 76– 79, 81–85, 91, 154, 155, 257, 258, 313, 341
\@esphack .....	177	\@glo@list .....	85
\@expandtwoargs .....	30	\@glo@long .....	55, 66, 80, 83
\@first@ofone .....	21	\@glo@longaccess .....	339–341
\@first@ofthree .....	108, 121, 124, 126, 139, 141, 143, 145, 355–357	\@glo@longpl .....	66, 80, 83, 233, 235, 237, 239, 241, 244, 246, 368
\@first@oftwo .....	24, 25, 71, 72, 108, 124, 125, 141, 142, 145, 146	\@glo@longpluralaccess .....	339–341
\@for .....	25, 30, 32, 43, 44, 51, 52, 71, 72, 113, 158–160, 170, 171, 178, 183, 189, 220, 233, 236, 238, 240, 242, 245, 247, 249, 265, 266, 319	\@glo@name .....	12, 62, 67, 79, 82, 83
\@glo@@desc .....	84	\@glo@no@assign@sortkey .....	171
\@glo@@symbol .....	84	\@glo@nonumberlist .....	65
\@glo@access .....	338, 340, 341, 343	\@glo@numfmt .....	181, 319
\@glo@addchildren .....	189, 193	\@glo@parent .....	13, 64, 80–82, 86, 87, 190
\@glo@assign@sortkey .....	171, 172, 200	\@glo@plural .....	63, 80, 82, 83, 370
\@glo@autosee .....	84	\@glo@pluralaccess .....	339–341
\@glo@autoseehook .....	85	\@glo@prefix .....	9, 65, 80, 86, 87, 112, 181, 318, 319
\@glo@check@mkidxrangechar .....	112, 181, 318, 319	\@glo@range .....	181, 318, 319
\@glo@check@sortallowed ..	12–14, 171, 174	\@glo@see .....	64, 80, 85
\@glo@childlist .....	189	\@glo@seeautonumberlist .....	9, 64
		\@glo@short .....	56, 66, 80, 83, 371
		\@glo@shortaccess .....	339–341, 368–371
		\@glo@shortpl .....	66, 80, 83, 233, 235, 237, 239, 241, 244, 246, 368, 371
		\@glo@shortpluralaccess .....	339–341
		\@glo@sort ...	12, 14, 20, 21, 62, 80, 82, 86, 87
		\@glo@sortedinsert .....	190
		\@glo@sortentries .....	191–193
		\@glo@sorthandler@case .....	192

<code>\@glo@sorthandler@letter</code> .....	192	<code>\@gls@adjustmode</code> .....	156
<code>\@glo@sorthandler@nocase</code> .....	193	<code>\@gls@automake</code> .....	171
<code>\@glo@sorthandler@word</code> .....	191	<code>\@gls@between</code> .....	266
<code>\@glo@sortinghandler</code> .....	189, 190	<code>\@gls@body</code> .....	148
<code>\@glo@sortinglist</code> .....	189, 190, 193	<code>\@gls@checkactual</code> .....	114, 165
<code>\@glo@sorttype</code> .....	172, 194, 195, 201	<code>\@gls@checkbar</code> .....	114, 165
<code>\@glo@storeentry</code> .....	11, 12, 14	<code>\@gls@checkedmkidx</code> .....	113–119, 164–166
<code>\@glo@suffix</code> .....	112, 181, 319	<code>\@gls@checkescactual</code> .....	113, 165
<code>\@glo@symbol</code> .....		<code>\@gls@checkescbar</code> .....	114, 165
.....	55, 63, 80, 84, 238, 239, 243, 244, 340	<code>\@gls@checkescquote</code> .....	113, 165, 166
<code>\@glo@symbolaccess</code> .....	339–341, 371	<code>\@gls@checklevel</code> .....	114, 166
<code>\@glo@symbolplural</code> .....	63, 80, 84	<code>\@gls@checkmkidxchars</code> .....	
<code>\@glo@symbolpluralaccess</code> .....	339–341	.....	86, 112, 165, 172, 180, 182, 318, 319
<code>\@glo@text</code> .....	62,	<code>\@gls@checkquote</code> .....	113, 164, 165
80, 82, 84, 121–126, 147, 148, 239, 258, 370		<code>\@gls@classI</code> .....	159
<code>\@glo@textaccess</code> ...	338, 340, 341, 368–371	<code>\@gls@classII</code> .....	159
<code>\@glo@thislabel</code> .....	85	<code>\@gls@codepage</code> .....	188
<code>\@glo@thislettergrp</code> .....	194–196	<code>\@gls@counter</code> .....	
<code>\@glo@thisvalue</code> .....	56, 57	107, 110–112, 155, 156, 175, 181, 182, 319	
<code>\@glo@tmp</code> .....	73, 74, 181	<code>\@gls@counterwithin</code> .....	11, 199, 201
<code>\@glo@type</code> .....	7, 8, 13, 14, 63,	<code>\@gls@ctr</code> .....	43
80, 81, 83, 84, 155, 156, 170, 175, 176,		<code>\@gls@currentlettergroup</code> .....	194, 196
185–190, 193, 194, 198, 199, 218, 234,		<code>\@gls@debugfalse</code> .....	5
236, 238, 240, 242, 245, 247, 249, 264, 266		<code>\@gls@debugtrue</code> .....	5
<code>\@glo@types</code> .....		<code>\@gls@declareoption</code> .....	
.....	51, 52, 59, 90, 156, 170, 171, 254, 313	.....	8–10, 15, 16, 18, 19, 24, 27–30
<code>\@glo@useri</code> .....	65, 80, 83	<code>\@gls@default</code> .....	96
<code>\@glo@userii</code> .....	66, 80, 83	<code>\@gls@default@value</code> .....	
<code>\@glo@useriii</code> .....	66, 80, 83	.....	55–57, 68, 69, 79, 80, 82–84, 243, 257
<code>\@glo@useriv</code> .....	66, 80, 83	<code>\@gls@deffile</code> .....	70, 71
<code>\@glo@userv</code> .....	66, 80, 83	<code>\@gls@defsort</code> .....	12–14, 84
<code>\@glo@uservi</code> .....	66, 80, 83	<code>\@gls@defsortcount</code> .....	11–14, 60
<code>\@glodesc</code> .....	84	<code>\@gls@do@acronymsdef</code> .....	15, 16, 31, 61
<code>\@glolist@</code> .....	81	<code>\@gls@do@indexdef</code> .....	30, 31, 61
<code>\@gloname</code> .....	83	<code>\@gls@do@numbersdef</code> .....	29, 31, 61
<code>\@glossary@default@style</code> .....		<code>\@gls@do@symbolsdef</code> .....	29, 61
.....	8–10, 185, 209, 250	<code>\@gls@do@symbolssdef</code> .....	31
<code>\@glossaryentryfield</code> .....	87	<code>\@gls@doautomake</code> .....	28, 171
<code>\@glossarysection</code> .....	39	<code>\@gls@docheckquotedef</code> .....	164–167
<code>\@glossarystyle</code> .....	185, 186, 198	<code>\@gls@docloadedfalse</code> .....	4
<code>\@glossarysubentryfield</code> .....	87	<code>\@gls@docloadedtrue</code> .....	4
<code>\@gls</code> .....	121	<code>\@gls@dodeflistparser</code> .....	170
<code>\@gls@</code> .....	92, 94, 121, 260, 261	<code>\@gls@doentrydef</code> .....	105, 106
<code>\@gls@@link</code> .....	108	<code>\@gls@dolast</code> .....	183
<code>\@gls@Hcounter</code> .....	111, 112	<code>\@gls@donext</code> .....	183
<code>\@gls@ReturnAfterFi</code> .....	212	<code>\@gls@donext@def</code> .....	154, 155
<code>\@gls@access@display</code> .....	343, 344	<code>\@gls@dothiswrite</code> .....	168, 169
<code>\@gls@actualchar</code> ...	87, 115, 118, 162, 322	<code>\@gls@elem</code> .....	265
<code>\@gls@addpredefinedattributes</code> .	158, 167	<code>\@gls@enablesavenonumberlist</code> .....	70

<code>\@gls@encapchar</code> .....	<code>\@gls@loclist</code> .....
..... 116, 117, 162, 181, 182, 319, 322	173, 174, 195, 196
<code>\@gls@entry@count</code> .....	<code>\@gls@map</code> .....
93	71, 72
<code>\@gls@entry@field</code> .....	<code>\@gls@missingnumberlist</code> .....
..... 73, 74, 91, 148–154, 341–343	83
<code>\@gls@escbsdq</code> .....	<code>\@gls@noaccess</code> .....
113, 163, 323	343
<code>\@gls@expand@fields</code> .....	<code>\@gls@noexpand@fields</code> .....
68, 69	69
<code>\@gls@expandonce</code> .....	<code>\@gls@nohyperlist</code> .....
69	18, 61, 109
<code>\@gls@extramakeindexopts</code> .....	<code>\@gls@noidx@do</code> .....
169	194
<code>\@gls@fetchfield</code> .....	<code>\@gls@noidx@getgrouptitle</code> .....
57	172
<code>\@gls@field@link</code> .....	<code>\@gls@noidx@sanitizesort</code> .....
74, 75, 127–139	20, 174
<code>\@gls@firsttok</code> .....	<code>\@gls@noidx@setsanitizesort</code> ....
194, 195	23, 174
<code>\@gls@fixbraces</code> .....	<code>\@gls@noidx@loclist@finalsep</code> .....
85	173
<code>\@gls@forbidtexext</code> .....	<code>\@gls@noidx@loclist@prev</code> ....
59	173, 196, 197
<code>\@gls@get@counterprefix</code> .....	<code>\@gls@noidx@loclist@sep</code> ....
181, 182	173, 196, 197
<code>\@gls@getbody</code> .....	<code>\@gls@noref@warn</code> .....
148	171, 194
<code>\@gls@getcounterprefix</code> .....	<code>\@gls@numberlink</code> .....
180	211, 212
<code>\@gls@getgrouptitle</code> .....	<code>\@gls@numbersdef</code> .....
172, 207, 266	29
<code>\@gls@glossary</code> .....	<code>\@gls@numlist@lastsep</code> .....
176, 177	154, 155
<code>\@gls@gobbleopt</code> .....	<code>\@gls@numlist@nextsep</code> .....
58	154, 155
<code>\@gls@grptitle</code> .....	<code>\@gls@numlist@sep</code> .....
207, 264, 266	154, 155
<code>\@gls@hyp@opt</code> .....	<code>\@gls@old@chapter</code> .....
74,	31
75, 93–95, 108, 121–146, 215–217, 259–262	<code>\@gls@oldnewglossaryentryposthook</code> .
<code>\@gls@hyp@opt@cs</code> .....	<code>\@gls@oldnewglossaryentryprehook</code> ..
108	340
<code>\@gls@hypergroup</code> .....	<code>\@gls@onlypremakeg</code> .....
265	32
<code>\@gls@ifinlist</code> .....	<code>\@gls@order</code> .....
43	168, 169
<code>\@gls@ifnotmeasuring</code> .....	<code>\@gls@org@LT@output</code> .....
87, 88	282
<code>\@gls@igtype</code> .....	<code>\@gls@org@glsnoidxdisplayloc</code> .....
62	174
<code>\@gls@increment@currcount</code> .....	<code>\@gls@org@glsseeformat</code> .....
91	174
<code>\@gls@indexdef</code> .....	<code>\@gls@patchtabularx</code> .....
30	88
<code>\@gls@initnonumberlist</code> .....	<code>\@gls@preglossaryhook</code> .....
65, 80	187
<code>\@gls@islistofacronyms</code> .....	<code>\@gls@prevlevel</code> .
17	293, 294, 313–316, 331, 332
<code>\@gls@keylist</code> .....	<code>\@gls@provide@newglossary</code> .....
367	59
<code>\@gls@keymap</code> .....	<code>\@gls@quotechar</code> .
65, 71–74, 257, 340	114–118, 162, 164–166, 322
<code>\@gls@label</code> .....	<code>\@gls@reference</code> .....
172, 175, 180, 181	172, 175
<code>\@gls@langmod</code> .....	<code>\@gls@removespaces</code> .....
168	212
<code>\@gls@levelchar</code> ....	<code>\@gls@renewglossary</code> .....
87, 116, 117, 162, 322	167
<code>\@gls@link</code> ..	<code>\@gls@replacementtext</code> .....
108, 121–127, 139–146, 355–357	343
<code>\@gls@link@checkfirsthyper</code> ....	<code>\@gls@rest</code> .....
121–126	148
<code>\@gls@link@label</code> .....	<code>\@gls@roman</code> .....
110, 235, 241	46, 319, 320
<code>\@gls@link@nocheckfirsthyper</code> 127, 139–146	<code>\@gls@sanitized@tmp</code> .....
<code>\@gls@link@opts</code> .....	113
110, 235, 241	<code>\@gls@sanitizedesc</code> .....
<code>\@gls@list</code> .....	26
265, 266	<code>\@gls@sanitizesort</code> .....
<code>\@gls@listsuffix</code> .....	12
43	<code>\@gls@sanitizesymbol</code> .....
<code>\@gls@loadlist</code> .....	26
9, 10, 249	<code>\@gls@saveentrycounter</code> .....
<code>\@gls@loadlong</code> .....	110, 156
9, 10, 249	<code>\@gls@savenonumberlist</code> .....
<code>\@gls@loadsuper</code> .....	65
9, 10, 249	<code>\@gls@see@noindex</code> .....
<code>\@gls@loadtree</code> .....	6, 64
10, 250	<code>\@gls@setacrstyle</code> .....
<code>\@gls@local@increment@currcount</code> ....	26, 31
92	<code>\@gls@setcounter</code> .....
	60
	<code>\@gls@setdefault@glslink@opts</code> ....
	110
	<code>\@gls@setsort</code> .....
	12–14, 110

\@gls@setupshortcuts	31	\@glsfirstletter	51, 158
\@gls@sort	196	\@glsfirstplural	129
\@gls@sort@A	190, 191	\@glsfirstplural@	129
\@gls@sort@B	190, 191	\@glshypernumber	211
\@gls@startswithexpandonce	68	\@glsisacronymlistfalse	17
\@gls@storenonumberlist	65, 84	\@glsisacronymlisttrue	17
\@gls@symbolsdef	29	\@glslink	110, 120, 121, 155, 264
\@gls@this	178	\@glslocalreset	88, 92
\@gls@thisHloc	181	\@glslocalunset	89, 92
\@gls@thisfield	57	\@glslocref	175, 180, 181, 318, 319
\@gls@thislabel	54, 183, 193	\@glsminrange	158, 159, 320
\@gls@thislist	154, 155	\@glsname	130
\@gls@thisloc	181	\@glsname@	130
\@gls@thisval	72	\@glsnavhypertarget	264
\@gls@title	39	\@glsnextpages	186
\@gls@tmp	13, 34, 48, 69, 113, 177, 265, 266	\@glsnodesc	79, 81, 84
\@gls@tmpb	114–119, 164–166	\@glsnoname	79, 82, 83
\@gls@toc	41	\@glsnonextpages	186
\@gls@type	171, 220, 233, 236, 238, 240, 242, 245, 247, 249, 313	\@glsnumberformat	107, 110, 155, 156, 175, 181, 318, 319
\@gls@updatechecked	113, 114, 165, 166	\@glsopenfile	167, 176
\@gls@usetranslator	24, 25, 34	\@glsorder	169, 170
\@gls@value	68, 151	\@glspl	123
\@gls@warnonglossdefined	19, 184	\@glspl@	92, 95, 123, 260–262
\@gls@warnontheGLOSSdefined	19, 204	\@glsplural	129
\@gls@write@entrycounts	93	\@glsplural@	129
\@gls@writedef	70	\@glsreset	88, 92
\@gls@writeisthook	162, 164	\@glssee	85, 183
\@gls@xdy@locationlist	159	\@glsshowtarget	5, 6, 120
\@gls@xdycheckbackslash	113	\@glsymbol	133
\@gls@xdycheckquote	113	\@glsymbol@	133
\@gls@xref	182	\@glsymbolplural	133
\@gls@Alphacompositor	37, 47, 320	\@glsymbolplural@	133, 134
\@glsHlocref	180	\@glstarget	120, 121, 204, 265
\@glsacronymlists	16, 17, 51, 218, 220, 233, 234, 236, 238, 240, 242, 245, 247, 249, 254	\@glstext	127
\@glsaddkey	73, 74	\@glstext@	127
\@glsaddstoragekey	72, 73	\@glsunset	89, 91
\@glsaddxdyattribute	44	\@glsuseri	134
\@glsdefaultsort	12	\@glsuseri@	134
\@glsdesc	131	\@glsuserii	135
\@glsdesc@	131	\@glsuserii@	135
\@glsdescplural	132	\@glsuseriii	136
\@glsdescplural@	132	\@glsuseriii@	136
\@glsdisp	126	\@glsuseriv	137
\@glsentry	90, 93	\@glsuseriv@	137
\@glsentrytitlecase	151	\@glsuserv	137, 138
\@glsfirst	128	\@glsuserv@	138
\@glsfirst@	128	\@glsuservi	138
		\@glsuservi@	138

\@glswidestname .....	313–315, 331	\@pgls .....	259
\@glswritefiles .....	28	\@pgls@ .....	259
\@glxtr@doaccsupp .....	337	\@pglspl .....	260
\@gobble .....	5, 12– 14, 71, 88, 113, 157, 160, 172, 317, 321, 322	\@pglspl@ .....	260
\@idxitem .....	307	\@plus .....	269, 289, 307
\@ifclassloaded .....	4, 10, 40	\@print@glossary .....	185
\@ifnextchar .....	60, 108	\@print@noidx@glossary .....	185
\@ifpackageloaded .....	..... 4, 8, 24, 25, 34, 50, 87, 154, 164, 337	\@printgloss@setsort .....	171, 172, 186
\@ifstar .....	59, 72, 73, 108, 213	\@printglossary .....	185
\@ifundefined .....	33, 265, 272, 283, 294, 301, 314, 315, 331, 345	\@roman .....	46, 319
\@ignored@glossaries .....	61, 62	\@secondofthree .....	..... 108, 121, 122, 124, 140, 142, 144, 146, 355
\@input@ .....	187	\@secondoftwo . . . . .	21, 24, 25, 34, 71, 72, 120– 123, 126, 139–141, 143, 144, 355–357, 375
\@istfilename .....	169, 170	\@set@glo@numformat .....	181, 319
\@makecol .....	282, 283	\@sglsaddkey .....	73
\@makeglossary .....	170	\@sglsaddstoragekey .....	72
\@minus .....	269, 289, 307	\@thirdofthree .....	..... 108, 123, 125, 141, 142, 144, 146, 355
\@mkboth .....	40	\@this@attr .....	160
\@newglossary .....	58, 59	\@this@childlabel .....	189
\@newglossaryentry@defcounters . .	85, 91	\@this@counter .....	44
\@newglossaryentryposthook .....	..... 73, 74, 85, 257, 340	\@this@ctr .....	160
\@newglossaryentryprehook .....	..... 73, 74, 79, 80, 257, 340	\@this@key .....	72
\@nil .....	17, 85, 112–114, 148, 165, 166, 181, 182, 194–196, 211, 212, 318, 319	\@this@label .....	189
\@nnil .....	17, 183	\@thiscs .....	32
\@no@makeglossaries .....	171, 172	\@tmp .....	46, 320
\@no@post@desc .....	324	\@use@option .....	30
\@nopostdesc .....	186	\@warn@nomakeglossaries .....	169, 188
\@onelevel@sanitize .....	20, 21, 46, 71, 87, 113, 161, 182, 184, 195, 320, 321	\@wrglossary@pageformat .....	179
\@onlypreamble . . . . .	60, 70, 79, 93, 96, 171, 174	\@wrglossarynumberhook .....	179, 180
\@onlypremakeg . . . . .	36, 37, 43, 45, 48, 60, 164	\@xdy@main@language .....	27, 168, 188
\@org@glossaryentrynumbers . . . .	186, 187	\@xdy@attributelist .....	44, 160
\@org@gl@s@assign@descplural .....	..... 233, 242, 244–247, 368, 371, 372	\@xdy@attributes .....	44, 159, 317, 319
\@org@gl@s@assign@firstpl .....	233, 235–237, 239, 240, 242, 244–247, 368–372	\@xdy@counters .....	43, 44, 160
\@org@gl@s@assign@plural .....	..... 233, 235, 237, 239–242, 244–247, 368–372	\@xdy@crossrefhook .....	160
\@org@gl@s@assign@symbolplural . .	233, 235–237, 239, 240, 244–247, 369, 370, 372	\@xdy@language .....	188
\@org@gl@snumberformat .....	154	\@xdylettergroups .....	51, 162, 322
\@org@newglossaryentryprehook .....	79	\@xdy@locationclassorder .....	49, 160, 321
\@outputpage .....	282, 283	\@xdy@locref .....	44, 162, 317, 321
\@p@glossarysection .....	39	\@xdy@requiredstyles .....	49, 158, 319
		\@xdysortrules .....	49, 162, 322
		\@xdystyle .....	158, 319
		\@xdyuseralphabets .....	46, 159, 319
		\@xdyuserlocationdefs . . . . .	48, 160, 318, 320
		\@xdyuserlocationnames .....	48, 318
		\@xfor@nextelement .....	183
		\\ .....	86, 113, 157, 163, 211, 212, 322, 323, 325–327, 335, 336



<code>\{</code> .....	71, 157, 163, 317, 322, 323	<code>\Acrshortpl</code> .....	231
<code>\}</code> .....	71, 157, 163, 317, 323	<code>\acrshortpl</code> .....	231
<code>\^</code> .....	21	<code>\addcontentsline</code> .....	42
<code>\‘</code> .....	21	<code>\addglossarytocaptions</code> .....	34
<code>\ </code> .....	114, 116, 166	<code>\addtolength</code> .....	315, 331
<code>\~</code> .....	21	<code>\advance</code> .....	13, 82, 111, 282
<b>A</b>		<code>\AE</code> .....	21
<code>\a</code> .....	21	<code>\ae</code> .....	21
<code>\AA</code> .....	22	amsgen package .....	4, 106
<code>\aa</code> .....	22	amsmath package .....	87
accsupp package .....	337	<code>\andname</code> .....	183
<code>\accsuppglossaryentryfield</code> .....	337	<code>\AnyTrackedLanguages</code> .....	34, 375
<code>\accsuppglossarysubentryfield</code> .....	338	<code>\appto</code> .....	18, 65, 73, 74, 257, 340
<code>\acrfootnote</code> .....	235, 241	array package .....	279, 283, 301
<code>\Acrfull</code> .....	232	article class .....	181
<code>\acrfull</code> .....	232	<code>\AtBeginDocument</code> ....	15, 50, 70, 87, 156, 172
<code>\ACRfullfmt</code> ....	216, 219, 227, 229, 363, 365	<code>\AtEndDocument</code> .....	28, 70, 93, 171, 175, 187, 188, 265
<code>\Acrfullfmt</code> ....	216, 219, 227, 229, 363, 365	<b>B</b>	
<code>\acrfullfmt</code> ....	215, 219, 227, 229, 363, 365	<code>\b</code> .....	21
<code>\acrfullformat</code> ....	153, 154, 215, 233, 248	babel package .....	24, 32, 34, 50
<code>\Acrfullpl</code> .....	232	<code>\begin</code> ...	161, 194, 269, 273–278, 281–307, 321
<code>\acrfullpl</code> .....	232	<code>\BeginAccSupp</code> .....	343
<code>\ACRfullplfmt</code> ...	217, 219, 227, 229, 363, 365	<code>\begingroup</code> .....	5, 177, 179, 212
<code>\Acrfullplfmt</code> ...	217, 219, 227, 229, 363, 365	<code>\bfseries</code> .....	274–277, 279, 280, 284–286, 288, 296–301, 303–307
<code>\acrfullplfmt</code> ...	216, 219, 227, 229, 363, 365	<code>\bgroup</code> .....	21, 79, 154, 186, 189
<code>\acrlinkfootnote</code> .....	234	booktabs package .....	278–281
<code>\acrlinkfullformat</code> .....	215–217	<code>\boolean</code> .....	247
<code>\Acrlong</code> .....	231	<code>\boolfalse</code> .....	29
<code>\acrlong</code> .....	231	<code>\booltrue</code> .....	29
<code>\Acrlongpl</code> .....	231	<code>\bottomrule</code> .....	279, 280
<code>\acrlongpl</code> .....	231	<code>\box</code> .....	282
<code>\acrnameformat</code> .....	239, 370	<b>C</b>	
<code>\acronymentry</code> .....	218, 221–226, 228–231, 359–361, 364–367	<code>\c</code> .....	21
<code>\acronymfont</code> .....	103, 139–142, 148, 153, 154, 217–219, 221–231, 234–236, 238, 240–245, 352, 353, 355–357, 359–361, 363–367, 369–371	<code>\c@equation</code> .....	111
<code>\acronymname</code> .....	15, 16, 35	<code>\c@glossarysubentry</code> .....	200
<code>\acronymsort</code> .....	218, 221–226, 228–231, 359–361, 364, 365, 367	<code>\c@page</code> .....	178–180
<code>\acronymtype</code> .....	15, 16, 218, 220, 233–242, 244–249, 368–371	<code>\cGls</code> .....	94
<code>\acrpluralsuffix</code> .....	219, 221– 224, 228–230, 233–237, 239–242, 244– 246, 248, 249, 359, 360, 364–366, 368–372	<code>\cGls</code> .....	94
<code>\Acrshort</code> .....	231	<code>\cGlsformat</code> .....	92
<code>\acrshort</code> .....	231	<code>\cGlsformat</code> .....	92
		<code>\cGlspl</code> .....	95
		<code>\cGlspl</code> .....	95
		<code>\cGlsplformat</code> .....	92
		<code>\cGlsplformat</code> .....	92
		<code>\char</code> .....	208
		classiethesis package .....	8



<code>\cleardoublepage</code> .....	41	154–156, 158, 162, 164–166, 168–170,
<code>\clearpage</code> .....	41	172, 173, 175, 176, 179–183, 185, 186,
<code>\closeout</code> .....	70, 162, 164, 168, 176	189, 193–202, 208–213, 215–218, 233–
<code>\compatglossarystyle</code> .....	324–336	240, 242–247, 249, 257, 259–263, 266–
<code>\compatibleglossentry</code> .....	206	268, 293, 294, 313–316, 318, 319, 322,
<code>\compatiblesubglossentry</code> .....	206	324, 331, 332, 337–340, 354–357, 368–372
<code>\copy</code> .....	282, 283	<code>\def@gl@xdycheckbackslash</code> .....
<code>\count@</code> .....	195	119
<code>\csdef</code> .....	19,	<code>\DefaultNewAcronymDef</code> .....
	73–75, 84, 85, 91, 92, 189, 190, 210, 220, 323	234
<code>\csedef</code> .....	93, 179	<code>\defgl@entryfmt</code> .....
<code>\csgdef</code> .....	39, 58, 61, 92, 93, 184, 197, 198	60, 61, 105, 106,
<code>\cslet</code> .....	65, 79, 85, 193	220, 232, 234, 237, 238, 240, 243, 246, 248
<code>\csname</code> .....	11–14, 30, 32, 34, 35, 40, 41, 44, 46,	<code>\define@boolkey</code> .....
	47, 50, 52, 54, 59–61, 67, 68, 72–78, 81–	..... 7, 9–11, 15, 22, 23, 26–29, 107, 200
	87, 89, 90, 105, 106, 110–112, 121–126,	<code>\define@choicekey</code> .....
	139–147, 154, 156, 159, 160, 165–168,	..... 5–7, 11, 23, 24, 27, 65, 198–200
	172, 175–177, 179, 181, 182, 185–188,	<code>\define@key</code> .....
	190, 198, 204, 206, 209, 210, 213, 250–	8, 11, 17, 18, 23, 27, 28, 62–66,
	258, 265, 266, 313–315, 317–319, 331,	73, 74, 106, 107, 155, 198, 200, 257, 338, 339
	337, 338, 341, 342, 345, 355–357, 373, 374	<code>\DefineAcronymSynonyms</code> .....
<code>\csshow</code> .....	254	31, 232
<code>\csuse</code> .....	35, 38, 58, 68,	<code>\delimN</code> .....
	74, 75, 105, 106, 168, 169, 190, 192, 194,	161, 170, 197, 211, 212, 321
	195, 197, 199, 200, 209, 221, 258, 324–336	<code>\delimR</code> .....
<code>\csxdef</code> .....	83, 93	161, 211, 321
<code>\currentglossary</code> .....	38, 186, 199, 201	<code>\DescriptionDUANewAcronymDef</code> .....
<code>\currentglssubentry</code> .....	200, 202	238
<code>\CurrentOption</code> .....	30, 257, 337	<code>\DescriptionFootnoteNewAcronymDef</code> .....
<code>\CurrentTrackedLanguage</code> .....	35, 375, 376	236
<code>\CurrentTrackedTag</code> .....	35, 375, 376	<code>\descriptionname</code> .....
<code>\CustomAcronymFields</code> .....	249	35, 274–277,
<code>\CustomNewAcronymDef</code> .....	249	279, 280, 284–286, 288, 296–301, 303–307
<b>D</b>		
<code>\d</code> .....	21	<code>\DescriptionNewAcronymDef</code> .....
<code>\datatool package</code> .....	190	240
<code>\day</code> .....	158, 162, 319, 322	<code>\DH</code> .....
<code>\DeclareAcronymList</code> .....	15–17, 218,	22
	220, 234, 236, 238, 240, 242, 245, 247, 249	<code>\dh</code> .....
<code>\DeclareListParser</code> .....	170	22
<code>\DeclareOption</code> .....	8, 257, 337	<code>\dimen@</code> .....
<code>\DeclareOptionX</code> .....	8	222, 282, 313
<code>\DeclareRobustCommand</code> .....		<code>\disable@keys</code> .....
	35, 183, 184, 243, 343–345	30
<code>\def</code> .....	8, 9, 11–14, 17,	<code>\do</code> .....
	21, 22, 27, 31, 32, 35, 36, 39, 43, 45, 47–	25, 30, 32,
	51, 54, 58–60, 62–66, 69, 77, 79–85, 88,	43, 44, 51, 52, 71, 72, 113, 154, 158–160,
	92, 94–96, 105–107, 110–119, 121–146,	170, 171, 178, 183, 189, 220, 233, 236,
		238, 240, 242, 245, 247, 249, 265, 266, 319
		<code>\do@glo@storeentry</code> .....
		11–14, 84
		<code>\do@gl@link@checkfirsthyper</code> .....
		..... 108, 110, 121–127, 139–146, 355, 356
		<code>\do@gl@xdycheckbackslash</code> .....
		113
		<code>\do@gl@disablehyperinlist</code> .....
		110
		<code>\do@gl@shaschildren</code> .....
		54
		<code>\doc package</code> .....
		4, 5, 14
		<code>\doifglossarynoexistsordo</code> .....
		59
		<code>\dtl@ifsingle</code> .....
		208
		<code>\dtl@insertinto</code> .....
		190
		<code>\dtl@sortresult</code> .....
		190, 191
		<code>\dtlcompare</code> .....
		191
		<code>\dtlicompare</code> .....
		191
		<code>\DTLifinlist</code> .....
		62, 109
		<code>\DTLifint</code> .....
		208
		<code>\dtlletterindexcompare</code> .....
		191
		<code>\DTLsubstituteall</code> .....
		113

<code>\dtlwordindexcompare</code> .....	190	68, 71–78, 81–90, 105, 106, 109, 111–	
<code>\DUANewAcronymDef</code> .....	247	119, 148, 155, 157, 160, 164–167, 176–	
<b>E</b>			
<code>\eappto</code> .....	61, 85, 179	178, 180, 181, 183, 186, 190, 195, 196,	
<code>\edef</code> .. 13, 17, 32, 34, 42–44, 46, 48, 49, 52,		204–206, 210, 212, 213, 235, 241, 250–	
54, 59, 61, 62, 68, 72, 76–79, 81, 85, 86,		255, 257, 258, 265, 266, 313, 317–319,	
105, 106, 110, 111, 113–119, 154, 156,		321, 322, 337, 338, 341, 343, 367, 373, 374	
157, 162, 164–166, 168–170, 172, 176,			
180, 181, 184, 187–191, 195, 200, 202,		<code>\expandonce</code> .....	
208, 212, 213, 233, 235, 237, 239, 241,		68, 69,	
244, 246, 264, 317, 318, 320, 322, 367–371		113, 165, 166, 179, 191, 204, 206, 218,	
<code>\egroup</code> .....	21, 79, 155, 187, 189	233, 235, 237, 239, 241, 244, 246, 337, 338	
<code>\else</code> .....	10, 13–18, 20, 22,	<b>F</b>	
23, 28, 30–32, 36, 37, 40–46, 48–51, 65,		<code>\fi</code> .....	5, 6, 8, 10, 12–17, 19, 20, 22, 23,
67, 81, 83, 86–88, 92, 109–119, 122–126,		25, 28, 30–32, 35–37, 40–51, 60, 65, 67,	
148, 158, 161–168, 176–183, 186, 195,		81–84, 86–88, 92, 93, 109–119, 122–126,	
199–203, 209, 211, 212, 222, 236, 238,		148, 156–158, 161, 163–167, 169–171,	
240, 242, 243, 245–247, 250, 265, 270,		176–184, 186, 188, 195, 199–203, 209–	
273, 275, 276, 279, 280, 282, 284, 286,		212, 222, 232, 234, 236, 238, 240, 242,	
287, 295, 297, 299, 302, 304, 306, 309,		243, 245–250, 256, 265, 270, 273, 275,	
310, 312, 314, 315, 318–324, 329–332, 343		276, 279, 280, 282–284, 286, 287, 295,	
<code>\emph</code> .....	183, 213	297, 299, 302, 304, 306, 309, 310, 312–	
<code>\empty</code> .....	212, 337	315, 317–321, 323, 324, 329–332, 337, 343	
<code>\end</code> .....	161, 194, 269, 273–278, 281–307, 321	<b>file types</b>	
<code>\end@doifinlist</code> .....	43	<code>.aux</code> .....	187
<code>\end@getprefix</code> .....	181, 182	<code>.glo</code> .....	86
<code>\end@glis@islistofacronyms</code> .....	17	<code>.ist</code> .....	157, 167
<code>\EndAccSupp</code> .....	343	<code>.toc</code> .....	42
<code>\endcsname</code> .....	11–	<code>.xdy</code> .....	36
14, 30, 32, 34, 35, 40, 41, 44, 46, 47,		<code>glo</code> .....	255
50, 52, 54, 59–61, 67, 68, 72–78, 81–		<code>\firstacronymfont</code> .....	
87, 89, 90, 105, 106, 110–112, 121–126,		.....	104, 105, 221–223, 228, 229,
139–147, 154, 156, 159, 160, 165–168,		234, 239, 241, 244, 354, 358–360, 364, 365	
172, 175–177, 179, 181, 182, 185–188,		<code>\footnote</code> .....	228, 229, 234, 365
190, 198, 204, 206, 209, 210, 213, 250–		<code>\FootnoteNewAcronymDef</code> .....	242
258, 265, 266, 313–315, 317–319, 331,		<code>\forall glossaries</code> .....	52, 175, 185, 313
337, 338, 341, 342, 345, 355–357, 373, 374		<code>\forall glossentries</code> .....	90, 93, 156
<code>\endfoot</code> .. 273–275, 277, 279, 280, 284–286, 288		<code>\ForEachTrackedDialect</code> .....	35, 375, 376
<code>\endgroup</code> .....	5, 177, 180, 212	<code>\forall glossentries</code> .....	52, 54, 85, 193, 313
<code>\endhead</code> .. 273–275, 277, 279, 280, 284–286, 288		<code>\forall listcsloop</code> .....	189, 194
<code>\endtheglossary</code> .....	5	<code>\forall listloop</code> .....	173, 174, 196
<code>\entryname</code> .....	35, 274–277,	<b>G</b>	
279, 280, 284–286, 288, 296–301, 303–307		<code>garamondx package</code> .....	214
<code>\equal</code> .....	23, 31, 41, 111, 170, 208, 265	<code>\gdef</code> .....	13, 44, 59, 76, 81, 82, 177, 201, 265
<code>equation (counter)</code> .....	111	<code>\Genacrformat</code> .....	
<code>etoolbox package</code> .....	4	104, 219, 221, 222, 228, 354, 358, 359, 365	
<code>\expandafter</code> .....	12–14, 21, 30, 32,	<code>\genacrformat</code> .....	104, 105,
34, 44, 46, 47, 49–52, 54, 59, 60, 62, 67,		219, 221, 222, 228, 353, 354, 358, 359, 364	
		<code>\GenericAcronymFields</code> ..	219, 221, 222,
		224–228, 230, 231, 358–361, 363, 364, 367	

<code>\Genplacrfullformat</code> .....		
.....	<a href="#">104</a> , <a href="#">219</a> , <a href="#">221–223</a> , <a href="#">228</a> , <a href="#">353</a> , <a href="#">359</a> , <a href="#">365</a>	
<code>\genplacrfullformat</code> .....		
.....	<a href="#">103–105</a> , <a href="#">219</a> , <a href="#">221</a> , <a href="#">222</a> , <a href="#">228</a> , <a href="#">353</a> , <a href="#">359</a> , <a href="#">365</a>	
<code>\glo@desc</code> .....	<a href="#">324</a>	
<code>\glo@do@compare</code> .....	<a href="#">190</a> , <a href="#">191</a>	
<code>\glo@grabfirst</code> .....	<a href="#">196</a>	
<code>\glo@label</code> .....	<a href="#">54</a> , <a href="#">85</a>	
<code>\glo@list</code> .....	<a href="#">85</a>	
<code>\glo@name</code> .....	<a href="#">205</a>	
<code>\glo@parent</code> .....	<a href="#">54</a>	
<code>\glo@type</code> .....	<a href="#">85</a>	
<code>\glo@value</code> .....	<a href="#">71</a>	
<code>\global</code> .....	<a href="#">12–14</a> , <a href="#">67</a> , <a href="#">70</a> , <a href="#">79</a> , <a href="#">84</a> , <a href="#">89</a> , <a href="#">90</a> , <a href="#">177</a> , <a href="#">187</a> , <a href="#">195</a> , <a href="#">196</a> , <a href="#">201</a> , <a href="#">282</a> , <a href="#">283</a>	
<code>\glolinkprefix</code> .....	<a href="#">110</a> , <a href="#">155</a> , <a href="#">204</a>	
<code>\glosortentrieswarning</code> .....	<a href="#">19</a> , <a href="#">189</a>	
<code>glossareentry (counter)</code> .....	<a href="#">202</a>	
<code>glossaries package</code> .....		
.....	<a href="#">30</a> , <a href="#">50</a> , <a href="#">158</a> , <a href="#">249</a> , <a href="#">257</a> , <a href="#">269</a> , <a href="#">317</a> , <a href="#">337</a>	
<code>glossaries-accsupp package</code> .....	<a href="#">86</a> , <a href="#">337</a>	
<code>glossaries-extra package</code> .....	<a href="#">160</a> , <a href="#">337</a>	
<code>\GlossariesWarning</code> .....	<a href="#">5</a> , <a href="#">6</a> , <a href="#">18</a> , <a href="#">19</a> , <a href="#">22</a> , <a href="#">23</a> , <a href="#">39</a> , <a href="#">42</a> , <a href="#">53</a> , <a href="#">57</a> , <a href="#">64</a> , <a href="#">67</a> , <a href="#">94</a> , <a href="#">95</a> , <a href="#">105–107</a> , <a href="#">154</a> , <a href="#">168</a> , <a href="#">169</a> , <a href="#">171</a> , <a href="#">173–175</a> , <a href="#">177</a> , <a href="#">182</a> , <a href="#">185</a> , <a href="#">206</a> , <a href="#">209</a> , <a href="#">317</a> , <a href="#">337</a>	
<code>\GlossariesWarningNoLine</code> .....		
.....	<a href="#">5</a> , <a href="#">6</a> , <a href="#">18</a> , <a href="#">170</a> , <a href="#">172</a> , <a href="#">176</a> , <a href="#">188</a> , <a href="#">265</a>	
<code>\glossary</code> .....	<a href="#">318</a> , <a href="#">319</a>	
<code>glossary package</code> .....	<a href="#">1</a> , <a href="#">213</a>	
<code>glossary styles:</code>		
<code>altlist</code> .....	<a href="#">270</a> , <a href="#">271</a> , <a href="#">325</a>	
<code>altlistgroup</code> .....	<a href="#">271</a> , <a href="#">325</a>	
<code>altlisthypergroup</code> .....	<a href="#">271</a> , <a href="#">325</a>	
<code>altlong4col</code> .....	<a href="#">277</a> , <a href="#">278</a> , <a href="#">286</a> , <a href="#">327</a>	
<code>altlong4col-booktabs</code> .....	<a href="#">280</a> , <a href="#">282</a>	
<code>altlong4colborder</code> .....	<a href="#">278</a> , <a href="#">327</a>	
<code>altlong4colheader</code> .....	<a href="#">278</a> , <a href="#">280</a> , <a href="#">327</a>	
<code>altlong4colheaderborder</code> ....	<a href="#">278</a> , <a href="#">327</a>	
<code>altlongragged4col</code> ...	<a href="#">281</a> , <a href="#">286–288</a> , <a href="#">328</a>	
<code>altlongragged4col-booktabs</code> ....	<a href="#">281</a>	
<code>altlongragged4colborder</code> ....	<a href="#">288</a> , <a href="#">328</a>	
<code>altlongragged4colheader</code> ....	<a href="#">287</a> , <a href="#">328</a>	
<code>altlongragged4colheaderborder</code>	<a href="#">288</a> , <a href="#">329</a>	
<code>altsuper4col</code> .....	<a href="#">300</a> , <a href="#">305</a> , <a href="#">336</a>	
<code>altsuper4colborder</code> .....	<a href="#">300</a> , <a href="#">336</a>	
<code>altsuper4colheader</code> .....	<a href="#">300</a> , <a href="#">336</a>	
<code>altsuper4colheaderborder</code> ...	<a href="#">300</a> , <a href="#">336</a>	
<code>altsuperragged4col</code> .....	<a href="#">305</a> , <a href="#">306</a> , <a href="#">334</a>	
<code>altsuperragged4colborder</code> ...	<a href="#">306</a> , <a href="#">334</a>	
<code>altsuperragged4colheader</code> ...	<a href="#">306</a> , <a href="#">334</a>	
<code>altsuperragged4colheaderborder</code> .	<a href="#">306</a> , <a href="#">334</a>	
<code>alttree</code> .....	<a href="#">293</a> , <a href="#">308</a> , <a href="#">313</a> , <a href="#">315</a> , <a href="#">331</a>	
<code>alttreegroup</code> .....	<a href="#">316</a> , <a href="#">332</a>	
<code>alttreehypergroup</code> .....	<a href="#">316</a> , <a href="#">332</a>	
<code>index</code> .....	<a href="#">8</a> , <a href="#">289</a> , <a href="#">307–310</a> , <a href="#">329</a>	
<code>indexgroup</code> .....	<a href="#">309</a> , <a href="#">329</a>	
<code>indexhypergroup</code> .....	<a href="#">309</a> , <a href="#">329</a>	
<code>inline</code> .....	<a href="#">324</a>	
<code>list</code> .....	<a href="#">8</a> , <a href="#">9</a> , <a href="#">269–271</a> , <a href="#">324</a>	
<code>listdotted</code> .....	<a href="#">271</a> , <a href="#">272</a> , <a href="#">325</a>	
<code>listgroup</code> .....	<a href="#">270</a> , <a href="#">324</a>	
<code>listhypergroup</code> .....	<a href="#">270</a> , <a href="#">325</a>	
<code>long</code> .....	<a href="#">273</a> , <a href="#">274</a> , <a href="#">279</a> , <a href="#">283</a> , <a href="#">325</a> , <a href="#">327</a>	
<code>long-booktabs</code> .....	<a href="#">279</a> , <a href="#">281</a>	
<code>long3col</code> .....	<a href="#">274</a> , <a href="#">275</a> , <a href="#">279</a> , <a href="#">326</a>	
<code>long3col-booktabs</code> .....	<a href="#">279</a> , <a href="#">281</a>	
<code>long3colborder</code> .....	<a href="#">275</a> , <a href="#">326</a>	
<code>long3colheader</code> .....	<a href="#">275</a> , <a href="#">279</a> , <a href="#">326</a>	
<code>long3colheaderborder</code> .....	<a href="#">275</a> , <a href="#">326</a>	
<code>long4col</code> .....	<a href="#">276</a> , <a href="#">277</a> , <a href="#">280</a> , <a href="#">326</a>	
<code>long4col-booktabs</code> .....	<a href="#">280</a>	
<code>long4colborder</code> .....	<a href="#">277</a> , <a href="#">327</a>	
<code>long4colheader</code> .....	<a href="#">276</a> , <a href="#">280</a> , <a href="#">327</a>	
<code>long4colheaderborder</code> .....	<a href="#">277</a> , <a href="#">327</a>	
<code>longborder</code> .....	<a href="#">273</a> , <a href="#">326</a>	
<code>longheader</code> .....	<a href="#">273</a> , <a href="#">279</a> , <a href="#">326</a>	
<code>longheaderborder</code> .....	<a href="#">274</a> , <a href="#">326</a>	
<code>longragged</code> .....	<a href="#">281</a> , <a href="#">283–285</a>	
<code>longragged-booktabs</code> .....	<a href="#">281</a>	
<code>longragged3col</code> .....	<a href="#">281</a> , <a href="#">285</a> , <a href="#">286</a> , <a href="#">328</a>	
<code>longragged3col-booktabs</code> .....	<a href="#">281</a>	
<code>longragged3colborder</code> .....	<a href="#">286</a> , <a href="#">328</a>	
<code>longragged3colheader</code> .....	<a href="#">286</a> , <a href="#">328</a>	
<code>longragged3colheaderborder</code> .	<a href="#">286</a> , <a href="#">328</a>	
<code>longraggedborder</code> .....	<a href="#">284</a> , <a href="#">327</a>	
<code>longraggedheader</code> .....	<a href="#">284</a> , <a href="#">328</a>	
<code>longraggedheaderborder</code> ....	<a href="#">285</a> , <a href="#">328</a>	
<code>mcolalttree</code> .....	<a href="#">293</a> , <a href="#">333</a>	
<code>mcolalttreegroup</code> .....	<a href="#">293</a> , <a href="#">333</a>	
<code>mcolalttreehypergroup</code> ...	<a href="#">293</a> , <a href="#">294</a> , <a href="#">333</a>	
<code>mcolindex</code> .....	<a href="#">289</a> , <a href="#">332</a>	
<code>mcolindexgroup</code> .....	<a href="#">289</a> , <a href="#">332</a>	
<code>mcolindexhypergroup</code> ....	<a href="#">289</a> , <a href="#">290</a> , <a href="#">332</a>	
<code>mcoltree</code> .....	<a href="#">290</a> , <a href="#">332</a>	
<code>mcoltreegroup</code> .....	<a href="#">332</a>	

mcoltreehypergroup .....	291, 332	\glossarymark .....	39
mcoltreenoname .....	292, 333	\glossaryname .....	14, 34, 35
mcoltreenonamegroup .....	292, 333	\glossarypostamble .....	161, 194, 321
mcoltreenonamehypergroup ...	292, 333	\glossarypreamble .....	160, 194, 321
sublistdotted .....	325	\glossarysection .....	160, 194, 321
super .....	295, 296, 303, 335	glossarysubentry (counter) ...	11, 202, 203
super3col .....	296–298, 335	\glossarysubentryfield .....	
super3colborder .....	297, 335	.....	206, 324–331, 333–336, 358
super3colheader .....	297, 335	\glossarytitle ..	160, 185, 186, 194, 198, 321
super3colheaderborder .....	298, 335	\glossarytoctitle .....	8, 14–16, 29,
super4col .....	298–300, 336	30, 32, 35, 39, 160, 185, 194, 198, 199, 321	
super4colborder .....	299, 336	\glossentry .....	86, 186, 187, 196, 206,
super4colheader .....	299, 336	267, 269–274, 276, 284, 285, 287, 295,	
super4colheaderborder .....	299, 336	297, 298, 302, 303, 305, 308, 310, 311, 314	
superborder .....	295, 335	\Glossentrydesc .....	357
superheader .....	296, 335	\glossentrydesc .....	
superheaderborder .....	296, 335	. 267–274, 276, 284, 285, 287, 295, 297,	
superragged .....	301, 303, 333	298, 302–305, 308–310, 312, 314, 315, 357	
superragged3col .....	303–305, 334	\glossentryname .....	
superragged3colborder .....	304, 334	. 267–274, 276, 284, 285, 287, 295, 297,	
superragged3colheader .....	304, 334	298, 302, 303, 305, 308–311, 314, 315, 357	
superragged3colheaderborder	305, 334	\Glossentrysymbol .....	358
superraggedborder .....	302, 333	\glossentrysymbol .....	267, 268, 276,
superraggedheader .....	303, 333	287, 298, 305, 308–310, 312, 314, 315, 358	
superraggedheaderborder ....	303, 334	\Gls .....	94, 213, 232
tree .....	290, 310, 311, 313, 329	\gls .....	94, 171, 203, 213, 232
treegroup .....	291, 311, 330	\gls@Alphpage .....	178, 180
treehypergroup .....	311, 330	\gls@alphpage .....	178, 180
treenoname .....	291, 308, 311, 312, 330	\gls@arabicpage .....	178, 180
treenonamegroup .....	312, 331	\gls@assign@desc .....	79, 84
treenonamehypergroup .....	312, 331	\gls@assign@descplural .....	
glossary-hypernav package .....	157	.....	233, 242, 244–247, 368, 371, 372
glossary-list package .....	8, 9, 269	\gls@assign@field .....	
glossary-long package ....	9, 272, 286, 294, 295	.....	69, 73, 74, 78, 81–84, 257, 258
glossary-longragged package .....	283	\gls@assign@firstpl .....	233,
glossary-mcols package .....	288	235–237, 239, 240, 242, 244–247, 368–372	
glossary-super package ...	9, 272, 294, 301, 305	\gls@assign@plural .....	
glossary-superragged package .....	301	233, 235, 237, 239–242, 244–247, 368–372	
glossary-tree package .....	10, 307	\gls@assign@symbolplural .....	233,
\glossaryentry .....	181, 182, 319	235–237, 239, 240, 244–247, 369, 370, 372	
glossaryentry (counter) ....	10, 11, 202, 203	\gls@checkisacronymlist .....	109
\glossaryentryfield .....		\gls@checkseeallowed ....	64, 70, 170, 172
.....	204, 324–331, 333–336, 358	\gls@checkseeallowed@preambleonly ..	70
\glossaryentrynumbers .....		\gls@codepage .....	50, 168, 188
.....	9, 161, 186, 187, 196, 200, 201, 321	\gls@defdocnewglossaryentry .....	70, 91
\glossaryheader .....		\gls@defglossaryentry .....	69–71, 79
.....	161, 194, 267, 269–271, 273–	\gls@disablepagerefexpansion ..	177, 180
	277, 279, 280, 283–289, 291–293, 295,	\gls@do@addxdyattribute .....	44
	296, 298, 302, 303, 305, 308–313, 316, 321	\gls@doclearpage .....	42

<code>\gls@dosubst</code> .....	113	<code>\glsadd</code> .....	156
<code>\gls@dotocitle</code> .....	186, 198	<code>\glsadd options</code>	
<code>\gls@end@sanitizesort</code> .....	21	counter .....	155
<code>\gls@endcheck</code> .....	68, 69	format .....	155, 210
<code>\gls@glossary</code> .....	176, 181, 182	<code>\glsaddall options</code>	
<code>\gls@gobbleopt</code> .....	60	types .....	155, 156
<code>\gls@grplabel</code> .....	264	<code>\GlsAddXdyAttribute</code> .....	43–45, 317, 318
<code>\gls@hypergroup rerun</code> .....	265	<code>\GlsAddXdyCounters</code> .....	44, 45, 60
<code>\gls@ifnotmeasuring</code> .....	88, 89	<code>\glsautomakefalse</code> .....	28
<code>\gls@inlinepostchild</code> .....	267, 268, 324	<code>\glsautoprefix</code> .....	7, 8, 199
<code>\gls@inlineseq</code> .....	267, 324	<code>\glsascapscase</code> .....	96, 98, 100–
<code>\gls@inlinesubseq</code> .....	267, 268, 324	104, 121–126, 139–146, 226, 239, 243,	
<code>\gls@islistofacronyms</code> .....	17	345, 347, 349, 350, 352, 353, 355–357, 362	
<code>\gls@istfilebase</code> .....	36, 168	<code>\glsclearpage</code> .....	41
<code>\gls@label</code> .....	213	<code>\glsclsebrace</code> .....	48, 161, 162, 321, 322
<code>\gls@level</code> .....	81, 82, 195	<code>\glscompositor</code> .....	37, 47, 163, 320, 323
<code>\gls@noidxglossary</code> .....	172	<code>\glscounter</code> .....	18, 31, 43, 60, 83, 111, 317
<code>\gls@nosetquote</code> .....	79, 162, 164, 167	<code>\glscurrententrylabel</code> .....	184, 187
<code>\gls@numberpage</code> .....	178, 180	<code>\glscurrentfieldvalue</code> .....	56, 57
<code>\gls@org@glossaryentryfield</code> ...	186, 187	<code>\glscustomtext</code> .....	
<code>\gls@org@glossarysubentryfield</code>	186, 187	.. 96, 100, 102, 104, 121–126, 139–146,	
<code>\gls@org@insert</code> .....	238, 241, 243	226, 227, 234, 235, 238–241, 243, 244,	
<code>\gls@protected@pagefmts</code> ....	113, 178, 179	345, 348, 349, 351, 352, 354–357, 362, 363	
<code>\gls@Romanpage</code> .....	178, 180	<code>\GlsDeclareNoHyperList</code> .....	18
<code>\gls@romanpage</code> .....	178, 180	<code>\glsdefaulttype</code> .....	15, 39, 50,
<code>\gls@save@numberlist</code> .....	8, 9	52, 58, 59, 80, 81, 96, 105, 106, 176, 184, 185	
<code>\gls@set@xr@key</code> .....	64	<code>\glsdefmain</code> .....	15, 61
<code>\gls@suffixF</code> .....	37, 161, 163, 321, 323	<code>\glsdescriptionaccessdisplay</code> .....	
<code>\gls@suffixFF</code> .....	38, 161, 163, 321, 323	..... 347–349, 357, 358	
<code>\gls@text</code> .....	105	<code>\glsdescriptionpluralaccessdisplay</code>	
<code>\gls@thissty</code> .....	25	..... 345, 346	
<code>\gls@tmp</code> .....	176, 243	<code>\glsdescwidth</code> .....	273–
<code>\gls@tmplen</code> ....	119, 120, 313–315, 331, 332	275, 277, 278, 281–288, 295–298, 300–307	
<code>\gls@tr@set@acronym@tocitle</code> ....	15, 16	<code>\glsdetoklabel</code> .....	
<code>\gls@tr@set@main@tocitle</code> .....	14	..... 53–57, 65, 71, 76–79, 85, 89–93,	
<code>\gls@tr@set@numbers@tocitle</code> .....	30	110, 147, 148, 154–156, 172–174, 180,	
<code>\gls@tr@set@symbols@tocitle</code> .....	29	187, 190, 191, 195, 196, 198–200, 202,	
<code>\gls@wrglossary</code> .....	177	203, 205, 250–254, 313, 337, 338, 373, 374	
<code>\gls@xdystring</code> .....	113	<code>\glsdisplay</code> .....	96, 106
<code>\gls@xindy@glslnumbersfalse</code> .....	28	<code>\glsdisplayfirst</code> .....	96, 105
<code>\gls@xindy@glslnumberstrue</code> .....	27	<code>\glsdisplaynumberlist</code> .....	173
<code>\gls@xr@key</code> .....	6, 64	<code>\glsdohyperlink</code> .....	120, 121
<code>\glsaccsupp</code> .....	343	<code>\glsdohypertarget</code> .....	120, 121
<code>\glsacronymtrue</code> .....	16	<code>\glsdoifexists</code> .....	
<code>\glsacrpluralsuffix</code> .....		.. 54–56, 75–78, 88, 89, 121–127, 139–	
..... 33, 214, 223, 224, 228–230, 234		146, 154, 156, 173, 174, 259–263, 354–358	
<code>\glsacrshortcutsfalse</code> .....	31	<code>\glsdoifexistsordo</code> .....	108, 147
<code>\glsacrshortcutstrue</code> .....	31	<code>\glsdoifexistsorwarn</code> .....	197, 204, 205
<code>\glsacspace</code> .....	222, 224	<code>\glsdoifnoexists</code> .....	69, 79

<code>\glsdonohyperlink</code> .....	110, 120	<code>\Glsentryprefix</code> .....	261
<code>\glsdosanitizesort</code> .....	11, 12	<code>\glsentryprefix</code> .....	259, 262
<code>\glsentryaccess</code> .....	343	<code>\Glsentryprefixfirst</code> .....	261
<code>\glsentrycounter</code> .....	209, 212	<code>\glsentryprefixfirst</code> .....	260, 262
<code>\glsentrycounterfalse</code> .....	10	<code>\Glsentryprefixfirstplural</code> .....	262
<code>\glsentrycounterlabel</code> .....	199, 203	<code>\glsentryprefixfirstplural</code> ....	260, 263
<code>\glsentrycountertrue</code> .....	11	<code>\Glsentryprefixplural</code> .....	261
<code>\glsentrycurrcount</code> .....	91, 93	<code>\glsentryprefixplural</code> .....	260, 263
<code>\Glsentrydesc</code> .....	131, 205, 357	<code>\glsentryprevcount</code> .....	91, 92
<code>\glsentrydesc</code> .....	..... 98–100, 131, 132, 205, 347–349, 357	<code>\Glsentryshort</code> .....	103, 140, 148, 222, 228, 229, 353–355, 359, 365, 366
<code>\glsentrydescaccess</code> .....	344	<code>\glsentryshort</code> 103, 104, 139, 141, 148, 153, 219, 221–231, 352–355, 358–361, 363–367	
<code>\Glsentrydescplural</code> .....	132	<code>\glsentryshortaccess</code> .....	344
<code>\glsentrydescplural</code> ..	97, 98, 132, 345, 346	<code>\Glsentryshorttpl</code> .....	..... 103, 142, 223, 229, 352, 359, 365, 366
<code>\glsentrydescpluralaccess</code> .....	344	<code>\glsentryshorttpl</code> .....	..... 103, 105, 141, 142, 154, 221, 222, 227–229, 248, 352, 354, 359, 363–366
<code>\Glsentryfirst</code> ....	94, 99, 102, 128, 348, 351	<code>\glsentryshortpluralaccess</code> .....	344
<code>\glsentryfirst</code> 94, 98–102, 128, 347, 348, 351		<code>\Glsentrysymbol</code> .....	133, 205, 358
<code>\glsentryfirstaccess</code> .....	344	<code>\glsentrysymbol</code> .....	98– 100, 133, 205, 235, 239, 243, 347–349, 358
<code>\Glsentryfirstplural</code> .....	..... 95, 98, 101, 130, 346, 350	<code>\glsentrysymbolaccess</code> .....	344
<code>\glsentryfirstplural</code> .....	95, 97, 98, 100, 101, 129, 130, 345, 346, 349, 350	<code>\Glsentrysymbolplural</code> .....	134
<code>\glsentryfirstpluralaccess</code> .....	344	<code>\glsentrysymbolplural</code> .....	..... 97, 98, 134, 235, 238, 243, 345, 346
<code>\glsentryfmt</code> .....	60, 61	<code>\glsentrysymbolpluralaccess</code> .....	344
<code>\Glsentryfull</code> .....	219, 227, 229, 364, 366	<code>\Glsentrytext</code> .....	99, 102, 128, 347, 351
<code>\glsentryfull</code> .....	219, 227, 229, 363, 366	<code>\glsentrytext</code> .....	98, 99, 101, 102, 127, 155, 184, 347, 348, 350, 351
<code>\Glsentryfullpl</code> ....	219, 228, 229, 364, 366	<code>\glsentrytextaccess</code> .....	343
<code>\glsentryfullpl</code> ....	219, 228, 229, 364, 366	<code>\glsentrytype</code> .....	81
<code>\glsentryitem</code> ...	199, 267, 269–274, 276, 284, 285, 287, 295, 297, 298, 302, 303, 305, 308, 310, 311, 314, 324–331, 333–336	<code>\Glsentryuseri</code> .....	135
<code>\Glsentrylong</code> .....	94, 144, 148, 153, 221, 222, 227, 354, 356, 359, 362–364	<code>\glsentryuseri</code> .....	134, 135
<code>\glsentrylong</code> ....	94, 104, 143, 144, 148, 153, 221, 222, 224–231, 241, 354, 356–367	<code>\Glsentryuserii</code> .....	135
<code>\glsentrylongaccess</code> .....	344	<code>\glsentryuserii</code> .....	135, 136
<code>\Glsentrylongpl</code> .....	95, 146, 154, 221, 222, 226–228, 354, 359, 362–364	<code>\Glsentryuseriii</code> .....	136
<code>\glsentrylongpl</code> .....	..... 95, 105, 145, 146, 154, 221–223, 226–229, 241, 248, 354, 359, 360, 362–366	<code>\glsentryuseriii</code> .....	136, 137
<code>\glsentrylongpluralaccess</code> .....	344	<code>\Glsentryuseriv</code> .....	137
<code>\Glsentryname</code> .....	131, 205, 357	<code>\glsentryuseriv</code> .....	137
<code>\glsentryname</code> .....	130, 131, 313, 357	<code>\Glsentryuserv</code> .....	138
<code>\glsentrynumberlist</code> .....	154, 173	<code>\glsentryuserv</code> .....	138
<code>\Glsentryplural</code> ....	97, 101, 129, 346, 350	<code>\Glsentryuservi</code> .....	139
<code>\glsentryplural</code> .....	.. 97, 98, 100, 101, 129, 345, 346, 349, 350	<code>\glsentryuservi</code> .....	138, 139
<code>\glsentrypluralaccess</code> .....	343	<code>\glsfieldfetch</code> .....	151
		<code>\glsfirstaccessdisplay</code> ....	347, 348, 351
		<code>\glsfirstpluralaccessdisplay</code> .....	..... 345, 346, 349, 350



<code>\glsfirstpluralaccessdisplay</code> .....	350	<code>hyper</code> .....	107, 109, 121
<code>\glsgenacfmt</code> ....	221, 222, 228, 358, 359, 364	<code>local</code> .....	107
<code>\glsgenentryfmt</code> .....		<code>\glslinkcheckfirsthyperhook</code> .....	109
.....	221, 222, 227, 228, 232, 234, 237, 238, 241, 243, 246, 248, 358, 359, 363, 364	<code>\glslinkpostsetkeys</code> .....	110
<code>\glsgetgrouptitle</code> .....		<code>\glslinkvar</code> .....	107, 108
.....	266, 270, 271, 289–294, 309–312, 316	<code>\glslistdottedwidth</code> .....	271, 272, 325
<code>\gls glossarymark</code> .....	39	<code>\glslistgroupheaderfmt</code> .....	270, 271
<code>\gls groupheading</code> 162, 196, 267, 269–271, 273, 274, 276, 284, 285, 287, 289–296, 298, 302, 303, 305, 308–313, 315, 316, 322		<code>\glslistnavigationitem</code> .....	270, 271
<code>\gls groupskip</code> 161, 162, 196, 268, 270, 273, 275, 276, 279, 280, 284–287, 295, 297, 299, 302, 304, 306, 309, 310, 312, 315, 321		<code>\glslocalreset</code> .....	90
<code>\glshyperfirstfalse</code> .....	228, 364	<code>\glslocalunset</code> .....	90, 122–126
<code>\glshyperfirsttrue</code> .....	26	<code>\gls longaccessdisplay</code> .....	354, 356–368
<code>\glshyperlink</code> .....	184	<code>\gls longkey</code> .....	372
<code>\glshypernavsep</code> .....	266	<code>\gls longpluralaccessdisplay</code> .....	
<code>\glshypernumber</code> .....	38, 212, 213	.....	354, 359, 360, 362–366, 368
<code>\glsifhyperon</code> .....	107	<code>\gls longpluralkey</code> .....	372
<code>\glsIfListOfAcronyms</code> .....	16, 17	<code>\gls longtok</code> .....	
<code>\glsifplural</code> .....	96, 100, 102, 103, 121–126, 139–146, 226, 235, 238, 241, 243, 345, 349, 352, 353, 355–357, 362	.....	218, 219, 221, 222, 227, 228, 233– 242, 244–249, 358, 359, 363, 364, 367–372
<code>\glsifusetranslator</code> ....	24, 25, 34, 35, 375	<code>\glsLTpenaltycheck</code> .....	282, 283
<code>\glsindexonlyfirstfalse</code> .....	26	<code>\gls mcols</code> .....	289–294
<code>\glsinlinedescformat</code> .....	267, 324	<code>\glsnameaccessdisplay</code> .....	357, 358
<code>\glsinlinedopostchild</code> .....	267, 324	<code>\glsnamefont</code> .....	204–206, 337, 338, 357
<code>\glsinlineemptydescformat</code> .....	267, 324	<code>\glsnavhyperlink</code> .....	266
<code>\glsinlinenameformat</code> .....	267, 324	<code>\glsnavhyperlinkname</code> .....	264, 265
<code>\glsinlineparentchildseparator</code> 267, 324		<code>\glsnavhypertarget</code> .....	
<code>\glsinlinepostchild</code> .....	267, 324	.....	270, 271, 290–294, 310–312, 316
<code>\glsinlineseparator</code> .....	267, 324	<code>\gls navigation</code> .....	
<code>\glsinlinesubdescformat</code> .....	268, 324	.....	270, 271, 289–294, 309, 311, 312, 316
<code>\glsinlinesubnameformat</code> .....	267, 324	<code>\glsnextpages</code> .....	9, 65, 186
<code>\glsinlinesubseparator</code> .....	268, 324	<code>\glsnogroupskipfalse</code> .....	10
<code>\glsinsert</code> .....	97– 104, 121–126, 139–146, 226, 227, 235, 238, 239, 241, 243, 244, 345–357, 362, 363	<code>\glsnoidxdisplayloc</code> .....	174, 175
<code>\glskeylisttok</code> .....		<code>\glsnoidxdisplaylocclsthandler</code> ....	173
.....	218, 219, 233–242, 244–247, 249, 367–372	<code>\glsnoidxloclist</code> .....	173, 196
<code>\glslabel</code> ....	79, 97–104, 109, 110, 140– 146, 221, 222, 226–228, 234, 235, 238, 239, 241, 243, 345–354, 358, 359, 362–364	<code>\glsnoidxloclisthandler</code> .....	196
<code>\glslabeltok</code> 218, 233–242, 244–249, 368–371		<code>\glsnoidxnumberlistloophandler</code> ....	174
<code>\glslink</code> .....	219, 227, 229, 234, 363, 365	<code>\glsnoidxstripaccents</code> .....	21
<code>\glslink options</code>		<code>\glsnomakeindexwarning</code> .....	164
<code>counter</code> .....	106, 121, 256	<code>\glsnonextpages</code> .....	65, 186
<code>format</code> .....	107, 121, 210	<code>\glsnopostdotfalse</code> .....	10
		<code>\glsnoxindywarning</code> 37, 43, 45, 46, 48–50, 158	
		<code>\glsnumberformat</code> .....	154
		<code>\glsnumberlistloop</code> .....	174
		<code>\glsnumbersgroupname</code> .....	29, 35, 208
		<code>\glsnumlistlastsep</code> .....	155, 173
		<code>\glsnumlistparser</code> .....	155, 170
		<code>\glsnumlistsep</code> .....	155, 173
		<code>\glsopenbrace</code> .....	48, 161, 162, 321, 322
		<code>\glsorder</code> .....	27, 168–170, 192
		<code>\glsorg@endtheglossary</code> .....	5

<code>\glsorg@PrintChanges</code> .....	5	<code>\glsstepentry</code> .....	199, 203
<code>\glsorg@theglossary</code> .....	5	<code>\glsstepsubentry</code> .....	200, 203
<code>\glspagelistwidth</code> 274, 275, 277, 278, 281, 282, 285–288, 296–298, 300, 301, 303–307		<code>\glssubentrycounterfalse</code> .....	11
<code>\glspatchLToutput</code> .....	279–281	<code>\glssubentrycounterlabel</code> .....	200, 203
<code>\glspenaltygroupskip</code> .....	279, 280	<code>\glssubentryitem</code> .....	
<code>\glspersentchar</code> .....	70, 71, 161, 162	..... 200, 268, 269, 271–274, 276, 284, 285, 287, 295, 297, 298, 302, 304, 305, 309, 310, 312, 314, 324–331, 333–336	
<code>\Glspl</code> .....	95, 232	<code>\glssymbolaccessdisplay</code> ...	347–349, 358
<code>\glspl</code> .....	95, 232	<code>\glssymbolpluralaccessdisplay</code> .	345, 346
<code>\glspluralaccessdisplay</code> 345, 346, 349, 350		<code>\glssymbolsgroupname</code> .....	29, 35, 208
<code>\glspluralsuffix</code> .....		<code>\glstarget</code> .....	206, 207, 268–274, 276, 284, 285, 287, 295, 297, 298, 302–305, 308–312, 314, 315, 324–336
..... 33, 83, 221–223, 359, 360, 364–366		<code>\glstextaccessdisplay</code> ..	347, 348, 350, 351
<code>\glspostdescription</code> .....		<code>\glstextformat</code> .....	108, 110
. 36, 268–271, 273, 284, 295, 302, 308– 310, 312, 314, 315, 324–327, 329–333, 335		<code>\glstextup</code> .....	33, 366
<code>\glspostinline</code> .....	267	<code>\glstildechar</code> .....	44, 161, 162
<code>\glspostlinkhook</code> .....		<code>\glstranslatefalse</code> .....	24, 25
..... 109, 122–127, 140–146, 355–357		<code>\glstranslatetrue</code> .....	25
<code>\glsprestandardsort</code> .....	12	<code>\glstreechildpredesc</code> .....	309, 310
<code>\glreset</code> .....	90	<code>\glstreegroupheaderfmt</code> .....	
<code>\glresetentrycounter</code> .....	199, 202	..... 289–294, 309, 311, 312, 316	
<code>\glresetentrylist</code> ....	161, 194, 201, 321	<code>\glstreeindent</code> ..	310, 312, 314, 315, 330–332
<code>\glresetsubentrycounter</code> .....		<code>\glstreeitem</code> .....	289, 290, 308
..... 199, 200, 203, 267, 324		<code>\glstreenamebox</code> .....	314, 315
<code>\glssanitizesortfalse</code> .....	23	<code>\glstreenamefmt</code> .....	307–311, 313–315
<code>\glssanitizesorttrue</code> .....	23	<code>\glstreenavigationfmt</code> .....	
<code>\glssavenumberlistfalse</code> .....	9	..... 289–294, 309, 311, 312, 316	
<code>\glssavewritesfalse</code> .....	29	<code>\glstreepredesc</code> .....	308, 310, 312
<code>\glseeformat</code> .....	160, 172, 174, 321	<code>\glstreesubitem</code> .....	289, 308
<code>\glseeitem</code> .....	183	<code>\glstreesubsubitem</code> .....	289, 308
<code>\glseeitemformat</code> .....	184	<code>\glstype</code> .	109, 110, 121–126, 139–146, 355–357
<code>\glseeelastsep</code> .....	183	<code>\glsucmarkfalse</code> .....	10
<code>\glseeelist</code> .....	183	<code>\glsucmarktrue</code> .....	10
<code>\glseeesep</code> .....	183	<code>\glunset</code> .....	88, 90, 92, 122–126
<code>\glssetexpandfield</code> .....	20, 22, 23	<code>\glsupacrpluralsuffix</code> .....	
<code>\glssetnoexpandfield</code> .....	20, 22, 23	..... 223, 230, 236, 240, 242, 245	
<code>\GlsSetQuote</code> .....	79, 162	<code>\GlsUseAcrEntryDisplayStyle</code> .....	220, 223–226, 228–230, 360, 361, 364, 366, 367
<code>\glssettocitle</code> .....	35, 186	<code>\GlsUseAcrStyleDefs</code> .....	220, 223–226, 228–231, 360, 361, 364, 366, 367
<code>\glsshortaccessdisplay</code> .....		<code>\glswrallowprimitivemodstrue</code> .....	179
..... 352–355, 358–361, 363–368		<code>\glswrite</code> ...	158–164, 169, 175, 176, 319–323
<code>\glsshortkey</code> .....	372	<code>\glswritedefhook</code> .....	71
<code>\glsshortpluralaccessdisplay</code> .....		<code>\glswriteentry</code> .....	178
..... 352, 354, 359, 363–366, 368		<code>\glswritefiles</code> .....	28, 29, 175
<code>\glsshortpluralkey</code> .....	372	<code>\glsxindyfalse</code> .....	27
<code>\glsshorttok</code> .....		<code>\glsxindytrue</code> .....	28
.... 218, 219, 233–242, 244–249, 368–372			
<code>\glsshowtarget</code> .....	6		
<code>\glssortnumberfmt</code> .....	13, 14		
<code>\glsspace</code> .....	215		



H	
\H	21
\hangindent	293, 294, 307, 310–312, 314–316, 329–332
\hbox	88, 271, 272, 325
\hfill	271, 272, 325
\hline	273–275, 277, 284–286, 288, 295–307
\hsize	272, 283, 294, 301
\hspace	308
\hss	271, 272, 325
\ht	282
\hyperdef	31
\hyperlink	107, 120, 212
hyperref package	181, 184, 211, 256
\hypertarget	120
I	
\IeC	21
\if	112, 181, 318
\if@endfor	265
\if@gl@debug	5, 18, 177
\if@gl@docloaded	4, 14, 176
\if@gl@isacronymlist	109
\if@openright	41
\ifbool	15, 26, 29, 53, 97, 99
\ifboolexpr	34, 58, 208
\ifcase	5–7, 25, 65, 198, 309, 329
\ifcsdef	24, 35, 41, 68, 74–78, 105, 106, 177, 189, 192–194, 209, 220
\ifcsempy	55, 259
\ifcsequal	55
\ifcsstrequal	78
\ifcsstring	77
\ifcsundef	7, 12, 27, 31, 34, 38, 39, 41, 52, 53, 60, 61, 63, 81, 83, 91, 106, 111, 112, 120, 168, 175, 184, 187, 190, 197, 198, 204, 208–211, 213, 220, 266, 323
\ifdef	56, 57, 65, 70, 71, 88, 107, 147, 151, 173, 174, 214, 307, 308
\ifdefempty	18, 40, 41, 52, 55–57, 61, 96, 100, 102, 171, 194–196, 218, 220, 226, 234, 238, 240, 243, 345, 349, 352, 362
\ifdefequal	54–57, 68, 69, 72, 81, 85, 196
\ifdefstrequal	78
\ifdefstring	9, 34, 58, 168–170, 191–193, 195, 197
\ifdefvoid	20, 21, 85, 195, 196
\ifdim	222, 282, 313
\iffalse	84, 89
\IfFileExists	9, 24, 25, 187
\ifglossaryexists	39, 50, 54, 167–169
\ifgl@sanitize@description	22
\ifgl@sanitize@name	22
\ifgl@sanitize@symbol	22
\ifgl@xindy@gl@numbers	51
\ifgl@sacrdescription	247
\ifgl@sacrdua	236, 243, 245, 247, 248
\ifgl@sacrfootnote	109, 247
\ifgl@sacronym	15
\ifgl@sacrshortcuts	31, 232
\ifgl@sacrsmalls	236, 237, 240, 242, 245
\ifgl@sacrsmalls	236, 238, 240, 242
\ifgl@automake	28, 171
\ifgl@descsuppressed	267
\ifgl@sentrycounter	199–203
\ifgl@sentryexists	53, 54, 70, 79, 82
\ifgl@shaschildren	267, 324
\ifgl@shasdesc	267
\ifgl@shaslong	94, 95, 148, 221, 222, 226, 228, 241, 358, 359, 362, 364
\ifgl@shasparent	190, 195, 313
\ifgl@shasprefix	261
\ifgl@shasprefixfirst	261
\ifgl@shasprefixfirstplural	261
\ifgl@shasprefixplural	261
\ifgl@shassymbol	234, 238, 243, 308–310, 312, 314, 315
\ifgl@shyperfirst	109
\ifgl@indexonlyfirst	178
\ifgl@nogroupskip	270, 273, 275, 276, 279, 280, 284, 285, 287, 295, 297, 299, 302, 304, 306, 309, 310, 312, 315
\ifgl@nonnumberlist	200
\ifgl@nopostdot	10
\ifgl@numberline	42
\ifgl@ssanitizesort	20, 23
\ifgl@ssavenumberlist	67, 170, 184
\ifgl@ssavewrites	28, 167, 177
\ifgl@ssubentrycounter	200, 202, 203
\ifgl@stoc	42
\ifgl@stranlate	34
\ifgl@sucmark	40
\ifgl@sused	93, 97–102, 109, 156, 178, 234, 238, 241, 243, 259–263, 345–352
\ifgl@swrallowprimitivemods	180
\ifgl@xindy	36, 37, 42–51, 60, 86, 113, 157, 158, 164, 168, 181, 182, 187, 317–319
\ifignoredglossary	81, 84, 177

<code>\ifin@</code> .....	30	113, 120–127, 139–146, 148, 154, 155,
<code>\ifinlistcs</code> .....	193, 198	162, 164, 167–172, 174, 177–179, 183,
<code>\ifKV@glslink@hyper</code> .....	109, 110	185–187, 196, 198, 201, 206, 218, 231–
<code>\ifKV@glslink@local</code> .....	122–126	233, 235–247, 257, 265, 266, 282, 289,
<code>\ifmeasuring@</code> .....	88	290, 308, 323, 340, 355–357, 368–372, 375
<code>\ifnum</code> .	12, 92, 195, 282, 310, 312, 314, 330, 331	<code>\letcs</code> .....
<code>\ifstrempy</code> .....	324	54–
<code>\ifstrequal</code> .....	208	57, 71, 73, 74, 77, 82, 83, 147, 148, 168,
<code>\ifthenelse</code>	23, 31, 41, 111, 170, 208, 247, 265	173, 174, 189–191, 195, 196, 205, 208, 313
<code>\IfTrackedLanguage</code> .....	164	link text .....
<code>\IfTrackedLanguageFileExists</code>	35, 375, 376	96
<code>\iftrue</code> .....	84, 89	<code>\listcsadd</code> .....
<code>\ifundef</code> .....	59, 70, 80, 158, 162, 169, 200	193
<code>\ifvmode</code> .....	156	<code>\listcsgadd</code> .....
<code>\ifvoid</code> .....	282	198
<code>\ifx</code> ...	12, 13, 16, 17, 30, 32, 43, 44, 46, 48,	<code>\listcsxadd</code> .....
	50, 51, 81–84, 86, 87, 111, 112, 114–119,	189
	148, 158, 161, 163–166, 176, 179–183,	<code>\listead</code> .....
	185, 186, 199, 201, 209, 211, 212, 234,	193
	236, 238, 240, 242, 243, 245, 247, 249,	<code>\loadglentries</code> .....
	250, 319–321, 323, 324, 329–332, 337, 343	96
<code>\immediate</code> ....	70, 71, 93, 168, 176, 187, 188	<code>\long</code> .....
<code>\in@</code> .....	30	79, 212
<code>\index</code> .....	177	<code>\longnewglossaryentry</code> .....
<code>\indexname</code> .....	30	79
<code>\indexspace</code> .	270, 289–294, 309–312, 315, 316	longtable package .....
<code>\input</code> .....	33, 96	272, 279, 283
<code>\inputencodingname</code> .....	27	<code>\LT@end@open</code> .....
<code>\InputIfFileExists</code> .....	70	282
<code>\istfilename</code> .	36, 158, 162, 169, 170, 319, 322	<code>\LT@err</code> .....
<code>\item</code>	269–272, 289, 290, 308, 309, 324, 325, 329	282
		<code>\LT@foot</code> .....
		282, 283
		<code>\LT@head</code> .....
		282, 283
		<code>\LT@lastfoot</code> .....
		282
		<code>\LT@output</code> .....
		282
		<b>M</b>
		<code>\makeatletter</code> .....
		70, 187
		<code>\makeatother</code> .....
		70
		<code>\makebox</code> ....
		271, 272, 313–315, 325, 331, 332
		makeglossaries ..
		27, 36, 50, 59, 164, 170, 187
		<code>\makeglossaries</code>
		6, 28, 32, 64, 171–173, 175, 188
		<code>\makeglossary</code> .....
		168, 170
		makeindex .....
		377
		makeindex .....
		11, 27, 28, 33,
		36–38, 42, 58–60, 62, 87, 112, 115, 157,
		160, 162, 164, 167, 176, 181, 207, 318, 319
		<code>delim_n</code> .....
		38
		<code>delim_r</code> .....
		38
		<code>page_compositor</code> .....
		37
		special characters .....
		113, 114, 157
		<code>\makenoidxglossaries</code> ..
		6, 64, 171, 174, 175
		<code>\MakeTextUppercase</code> .....
		4
		<code>\MakeUppercase</code> .....
		346, 348, 355, 357
		<code>\marginpar</code> .....
		6
		<code>\markboth</code> .....
		40
		<code>\mbox</code> .....
		156, 270, 293, 294, 313, 325
		memoir class .....
		177
		<code>\memUHead</code> .....
		40
		<code>\MessageBreak</code> .
		35, 58, 185, 186, 337, 375, 376
		mfirstuc package .....
		1

`\mfirstucMakeUppercase` .....  
     ..... 4, 40, 75, 98–104, 127–139, 141,  
     142, 144, 146, 219, 226, 227, 229, 239,  
     244, 262, 263, 350–354, 362, 363, 365, 366  
`\midrule` ..... 279, 280  
`\month` ..... 158, 162, 319, 322  
`multicol` package ..... 288

**N**

`\n` ..... 163, 322  
`\NeedsTeXFormat` ... 4, 257, 317, 323, 337, 375  
`\new@glossaryentry` ..... 70, 173  
`\new@ifnextchar` ..... 58, 74, 75, 94,  
     95, 121–125, 127–146, 215–217, 259–262  
`\newacronym` ..... 213,  
     218, 234, 236, 238, 240, 242, 245, 247, 249  
`\newacronymhook` ..... 218,  
     234, 236, 238, 240, 242, 245, 247, 249, 367  
`\newacronymstyle` ..... 221–226, 228–230  
`\newcommand` ..... 6–21,  
     23, 24, 26–34, 36–65, 67–79, 85, 86, 88–  
     91, 93–96, 100, 102, 104–109, 111, 113,  
     119–158, 164, 167–169, 171, 174–185,  
     187–197, 200–210, 212–222, 231–255,  
     258–262, 264–266, 268, 269, 282, 289,  
     307, 308, 313, 323, 341–343, 358, 372–374  
`\newcount` ..... 12, 13, 67  
`\newcounter` ..... 199–202  
`\newenvironment` ..... 204  
`\newglossary` ..... 14–16, 29, 30, 60, 170  
`\newglossaryentry` 30, 67, 70, 91, 218, 233,  
     235, 237, 239, 241, 244, 246, 248, 368–371  
`\newglossaryentry` options  
   `access` ..... 340, 341  
   `counter` ..... 63  
   `description` .....  
     . 26, 62, 67, 69, 79, 131, 149, 214, 242, 339  
   `descriptionaccess` ..... 342, 344  
   `descriptionplural` ..... 132, 339  
   `descriptionpluralaccess` ..... 342, 344  
   `first` .... 63, 83, 121, 128, 150, 240, 245, 338  
   `firstaccess` ..... 342, 344  
   `firstplural` ..... 63, 129, 150, 339  
   `firstpluralaccess` ..... 342, 344  
   `format` ..... 159  
   `long` ..... 102, 153, 339  
   `longaccess` ..... 343, 344  
   `longplural` ..... 153, 339  
   `longpluralaccess` ..... 343, 344  
   `name` ..... 62, 67, 69, 79, 130, 147, 184, 338  
   `nonumberlist` ..... 65  
   `parent` ..... 64, 69  
   `plural` ..... 63, 83, 128, 339  
   `pluralaccess` ..... 342, 343  
   `prefix` ..... 257  
   `prefixfirst` ..... 257  
   `prefixfirstplural` ..... 258  
   `prefixplural` ..... 258  
   `see` ..... 6, 9, 64, 70, 170, 172  
   `short` ..... 102, 153, 339  
   `shortaccess` ..... 342, 344  
   `shortplural` ..... 153, 339  
   `shortpluralaccess` ..... 342, 344  
   `sort` ..... 62, 151, 207  
   `symbol` ..... 62, 63,  
     133, 236, 237, 240, 245, 276, 298, 338–340  
   `symbolaccess` ..... 342, 344  
   `symbolplural` ..... 133, 339  
   `symbolpluralaccess` ..... 342, 344  
   `text` .... 62, 63, 121, 127, 149, 236, 240, 338  
   `textaccess` ..... 342, 343  
   `type` ..... 15, 63, 96, 151  
   `user1` ..... 134, 151, 340  
   `user2` ..... 135, 151  
   `user3` ..... 136, 152  
   `user4` ..... 137, 152  
   `user5` ..... 137, 152  
   `user6` ..... 138, 152, 340  
`\newglossarystyle` .....  
     266, 269–281, 283–306, 308–313, 315, 316  
`\newif` ..... 4, 5, 17, 24, 27, 179  
`\newlength` ..... 119, 272, 283, 294, 301, 311  
`\newrobustcmd` .....  
     ..... 69, 70, 74, 75, 93–95, 108, 121–  
     146, 148–154, 156, 214–217, 258–262, 313  
`\newterm` ..... 30  
`\newtoks` ..... 114, 167, 218  
`\newwrite` ..... 70, 158, 162, 167, 169  
`ngerman` package ..... 164  
`\noalign` ..... 282  
`\nobreak` ..... 270, 283, 325  
`\noexpand` .....  
     .. 17, 32, 43, 44, 83–85, 105, 106, 111–  
     113, 119, 155, 164–166, 168–170, 179,  
     180, 184, 187, 188, 190, 191, 204, 206,  
     213, 218, 219, 233, 235, 237, 239, 241,  
     244, 246, 248, 249, 317, 337, 338, 368–372  
`\nohyperpage` ..... 211

`\noindent` ..... 206, 290–292, 294, 311, 312  
`\noist` ..... 322, 323  
`\nopostdesc` ..... 30, 36, 79, 186, 324  
`\normalbaselineskip` ..... 282  
`\nr` ..... 5–7, 24, 25, 65, 198  
`\ns@ACRfull` ..... 216  
`\ns@Acrfull` ..... 215  
`\ns@acrfull` ..... 215  
`\ns@ACRfullpl` ..... 217  
`\ns@Acrfullpl` ..... 217  
`\ns@acrfullpl` ..... 216  
`\ns@ACRlong` ..... 144  
`\ns@Acrlong` ..... 143  
`\ns@acrlong` ..... 143  
`\ns@ACRlongpl` ..... 146  
`\ns@Acrlongpl` ..... 145  
`\ns@acrlongpl` ..... 145  
`\ns@ACRshort` ..... 140  
`\ns@Acrshort` ..... 140  
`\ns@acrshort` ..... 139  
`\ns@ACRshortpl` ..... 142  
`\ns@Acrshortpl` ..... 141, 142  
`\ns@acrshortpl` ..... 141  
`\ns@newglossary` ..... 59  
`\null` ..... 113–119, 164–166, 187  
`\number` ..... 12, 82, 93, 178–180, 206, 338  
`\numberline` ..... 42  
`\numexpr` ..... 93

## O

`\O` ..... 22  
`\o` ..... 22  
`\OE` ..... 21  
`\oe` ..... 22  
`\openout` ..... 70, 158, 162, 168, 319, 322  
`\OR` ..... 247  
`\or` ..... 5–7, 25, 198, 199, 309, 329  
`\org@glossaryentrynumbers` ..... 186, 201  
`\org@glossarytitle` ..... 185, 186  
`\org@glspostdescription` ..... 36  
`\org@ifKV@glslink@hyper` ..... 110  
`\orgAlph` ..... 179, 180  
`\orgalph` ..... 179, 180  
`\orgarabic` ..... 179, 180  
`\orgnumber` ..... 179, 180  
`\orgRoman` ..... 179, 180  
`\orgromannumeral` ..... 179, 180  
`\orgthe` ..... 179, 180  
`\outputpenalty` ..... 282

## P

`\p@` ..... 269, 289, 307, 308  
`\p@glsl@hyp@opt` ..... 108  
package options:  
`acronym` ..... 15, 32, 185, 214  
`true` ..... 16  
`counter` ..... 18  
`debug`  
`showtargets` ..... 6  
`description` ..... 240  
`dua` ..... 238, 240  
`entrycounter` ..... 200, 201  
`true` ..... 11  
`footnote` ..... 121–126, 236, 238, 240, 242  
`hyperfirst`  
`false` ..... 121–126  
`indexonlyfirst` ..... 384  
`makeindex` ..... 160, 256  
`nogroupskip` .... 273, 275, 276, 279, 280,  
   284, 285, 287, 295, 297, 299, 302, 304, 306  
`nolist` ..... 249  
`nolong` ..... 249, 272  
`nomain` ..... 14, 15  
`nonumberlist` ..... 8  
`nosuper` ..... 249  
`notree` ..... 250  
`nowarn` ..... 5  
`numberline` ..... 7  
`sanitize` ..... 22, 62, 147, 149  
`sanitizesort` ..... 19  
`savewrites` ..... 28, 381  
`false` ..... 167  
`true` ..... 169, 175  
`section` ..... 7, 40  
`sort`  
`def` ..... 11, 12  
`standard` ..... 11  
`use` ..... 11, 12  
`style` ..... 8, 249, 250  
`subentrycounter` ..... 200, 202  
`toc` ..... 7  
`true` ..... 7  
`translate` ..... 24  
`false` ..... 24  
`translator` ..... 24  
`xindy` ..... 27, 28, 160, 256  
`\PackageError` ..... 6,  
   14, 28, 32, 44, 50, 53, 54, 58, 64, 67,

73–78, 80–82, 91, 106, 147, 167, 168, 170, 173–175, 192–194, 198, 200, 209, 210, 220, 236–238, 243, 245, 246, 323, 345	\protected@xdef .....
\PackageInfo ..... 5, 6, 168, 177	..... 12, 13, 16, 21, 68, 86, 87, 180, 341
\PackageWarning ..... 5, 18	\providecommand 15, 32, 33, 40, 58, 93, 121, 160, 169, 172, 188, 204, 206, 269, 289, 307
\PackageWarningNoLine 5, 6, 18, 35, 375, 376	\ProvidesFile ..... 33
\pagegoal ..... 282	\ProvidesPackage .....
\pagelistname ..... 35, 275, 277, 279, 280, 286, 288, 297–301, 304–307	..... 4, 257, 264, 266, 269, 272, 278, 283, 288, 294, 301, 307, 317, 323, 337, 375
\par ..... 36, 206, 207, 269–271, 289, 291–294, 307, 308, 310–316, 325, 330–332	<b>R</b>
\parindent ..... 289–294, 308, 310–312, 314–316, 330–332	\r ..... 21
\parskip ..... 289–292, 308, 310, 311	\raggedright ..... 281–288, 302–307
\PassOptionsToPackage ..... 257, 337	\raisebox ..... 120
\penalty ..... 282	\ref ..... 203
\phantomsection ..... 41	\refstepcounter ..... 199, 200, 202
polyglossia package ..... 24, 34	\relax .... 5, 7, 9, 10, 13, 15, 24, 25, 31, 45, 58, 63, 65, 68, 81, 82, 84, 88, 92, 93, 108, 111–119, 148, 161, 162, 164, 166, 167, 170, 172, 174, 178, 180–183, 185, 187, 195, 198, 208, 209, 250, 265, 269, 282, 289, 293, 294, 307, 309–316, 318, 321– 323, 329–332, 340, 355, 356, 368–370, 372
\printglossaries ..... 170	\renewacronymstyle . 358–362, 364, 366, 367
\printglossary .. 16, 19, 29, 30, 170, 185, 200	\renewcommand ..... 4–11, 14–19, 23, 25, 26, 28, 31, 34–38, 50, 61, 64, 65, 79, 91–93, 154, 156, 158, 164, 165, 170– 174, 176, 187, 188, 198–200, 218, 219, 221–231, 234, 236, 238, 240, 242, 245, 247, 249, 267–277, 279, 280, 282–299, 302–306, 308–318, 323–331, 333–336, 340, 345, 349, 352, 354, 357–361, 363–371
\printglossary options	\renewenvironment ..... 204, 266, 269, 273–278, 281–308, 310, 311, 313
entrycounter ..... 199	\RequireGlossariesLang ..... 35, 375, 376
nogroupskip ..... 199	\RequirePackage .....
nonumberlist ..... 200	.. 4, 9, 10, 24, 25, 30, 34, 249, 256, 257, 272, 278, 279, 283, 289, 294, 301, 338, 375
nopostdot ..... 199	\restorecounters@ ..... 111
numberedsection ..... 198	\romannumeral ..... 178–180, 313–315, 331
style ..... 198	<b>S</b>
subentrycounter ..... 200	\s@glshyp@opt ..... 108
title ..... 198	\s@newglossary ..... 59
toctitle ..... 198	\savecounters@ ..... 111
type ..... 15, 184, 198	\seename ..... 183
\printindex ..... 30	\SetAcronymStyle ..... 26
\printnoidxglossaries ..... 172	\setbool ..... 23
\printnoidxglossary .....	\setbox ..... 282, 283
..... 171, 172, 175, 185, 192, 193, 201	\setcounter ..... 199, 200, 202
\printnoidxglossary options	\SetCustomDisplayStyle ..... 249
sort ..... 200	
\printnumbers ..... 29	
\printsymbols ..... 29	
\ProcessOptions ..... 257, 337	
\ProcessOptionsX ..... 30	
\protect ..... 42, 104, 105, 221–223, 228, 229, 235, 239, 241, 354, 358, 359, 364, 365	
\protected@edef ..... 8, 44, 46, 49, 51, 81, 85, 86, 97, 99, 105, 111, 112, 154, 177, 180, 199, 204, 206, 209, 218, 243, 248, 258, 264, 318, 319, 337, 338, 343	
\protected@write ..... 58, 59, 158– 160, 169, 170, 172, 175, 177, 184, 265, 319	



