

# Documented Code For glossaries v4.30

Nicola L.C. Talbot

Dickimaw Books

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

2017-06-11

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 v4.30: 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	31
1.4	Xindy	41
1.5	Loops and conditionals	50
1.6	Defining new glossaries	57
1.7	Defining new entries	61
1.8	Resetting and unsetting entry flags	87
1.9	Keeping Track of How Many Times an Entry Has Been Unset	90
1.10	Loading files containing glossary entries	95
1.11	Using glossary entries in the text	95
1.12	Adding an entry to the glossary without generating text	154
1.13	Creating associated files	156
1.14	Writing information to associated files	175
1.15	Glossary Entry Cross-References	181
1.16	Displaying the glossary	183
1.17	Acronyms	212
1.18	Predefined acronym styles	217
1.19	Predefined Glossary Styles	248
1.20	Debugging Commands	249
1.21	Compatibility with version 2.07 and below	254
<b>2</b>	<b>Prefix Support (glossaries-prefix Code)</b>	<b>256</b>
<b>3</b>	<b>Glossary Styles</b>	<b>263</b>
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	263
3.2	In-line Style (glossary-inline.sty)	265
3.3	List Style (glossary-list.sty)	267
3.4	Glossary Styles using longtable (the glossary-long package)	271
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	277
3.6	Glossary Styles using longtable (the glossary-longragged package)	282
3.7	Glossary Styles using multicol (glossary-mcols.sty)	287
3.8	Glossary Styles using supertabular environment (glossary-super package)	293
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	300
3.10	Tree Styles (glossary-tree.sty)	306

<b>4 Backwards Compatibility</b>	<b>316</b>
4.1 glossaries-compatible-207 . . . . .	316
4.2 glossaries-compatible-307 . . . . .	322
<b>5 Accessibility Support (glossaries-accsupp Code)</b>	<b>336</b>
5.1 Defining Replacement Text . . . . .	337
5.2 Accessing Replacement Text . . . . .	340
5.3 Displaying the Glossary . . . . .	356
5.4 Acronyms . . . . .	357
5.5 Debugging Commands . . . . .	372
<b>6 Multi-Lingual Support</b>	<b>374</b>
6.1 Polyglossia Captions . . . . .	374
<b>Glossary</b>	<b>376</b>
<b>Change History</b>	<b>377</b>
<b>Index</b>	<b>400</b>

# 1 Main Package Code

## 1.1 Package Definition

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

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2017/06/11 v4.30 (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.

```
32 \define@boolkey{glossaries.sty}{@gls@}{debug}[true]{%
33   \if@gls@debug
34     \renewcommand*{\GlossariesWarning}[1]{%
35       \PackageWarning{glossaries}{##1}%
36     }%
37     \renewcommand*{\GlossariesWarningNoLine}[1]{%
38       \PackageWarningNoLine{glossaries}{##1}%
39     }%
40     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
41   \else
42     \PackageInfo{glossaries}{debug mode OFF}%
43   \fi
44 }
```

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

gls@see@noindex

```
45 \newcommand*{\@gls@see@noindex}{%
46   \PackageError{glossaries}{%

```

```

47 {'\gls@xr@key' key may only be used after \string\makeglossaries\space
48 or \string\makenoidxglossaries}%
49 {You must use \string\makeglossaries\space
50 or \string\makenoidxglossaries\space before defining
51 any entries that have a '\gls@xr@key' key}%
52 }

```

seenoinindex

```

53 \define@choicekey{glossaries.sty}{seenoinindex}[\val\nr]{error,warn,ignore}{%
54   \ifcase\nr
55     \renewcommand*{\@gls@see@noindex}{%
56       \PackageError{glossaries}%
57       {'\gls@xr@key' key may only be used after \string\makeglossaries\space
58       or \string\makenoidxglossaries}%
59       {You must use \string\makeglossaries\space
60       or \string\makenoidxglossaries\space before defining
61       any entries that have a '\gls@xr@key' key}%
62     }%
63   \or
64     \renewcommand*{\@gls@see@noindex}{%
65       \GlossariesWarning{'\gls@xr@key' key ignored}%
66     }%
67   \or
68     \renewcommand*{\@gls@see@noindex}{}%
69   \fi
70 }

```

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.

```
71 \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.

```
72 \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.

```

73 \ifcsundef{chapter}%
74   {\newcommand*{\@@glossarysec}{section}}%
75   {\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.

```

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

```

Determine whether or not to use numbered sections.

glossarysecstar

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

glossaryseclabel

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

\glsautoprefix

Prefix to add before label if automatically generated:

```
81 \newcommand*{\glsautoprefix}{}
```

numberedsection

```
82 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
83 false,nolabel,autolabel,nameref}[nolabel]{%
84   \ifcase\nr\relax
85     \renewcommand*{\@@glossarysecstar}{*}%
86     \renewcommand*{\@@glossaryseclabel}{}%
87   \or
88     \renewcommand*{\@@glossarysecstar}{}%
89     \renewcommand*{\@@glossaryseclabel}{}%
90   \or
91     \renewcommand*{\@@glossarysecstar}{}%
92     \renewcommand*{\@@glossaryseclabel}{%
93       \label{\glsautoprefix\@glo@type}}}%
94   \or
95     \renewcommand*{\@@glossarysecstar}{*}%
96     \renewcommand*{\@@glossaryseclabel}{%
97       \protected@edef\@currentlabelname{\glossarytoctitle}%
98       \label{\glsautoprefix\@glo@type}}}%
99   \fi
100 }
```

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

```
101 \ifpackageloaded{classicthesis}
102 {\newcommand*{\@glossary@default@style}{index}}
103 {\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](#).

```
104 \define@key{glossaries.sty}{style}{%
105   \renewcommand*{\@glossary@default@style}{#1}%
106 }
```

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`

```
107 \newcommand*{\@gls@declareoption}[2]{%
108   \DeclareOptionX{#1}{#2}%
109   \DeclareOption{#1}{#2}%
110 }
```

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`

```
111 \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).

```
112 \@gls@declareoption{nonumberlist}{%
113   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
114 }
```

`savenumberlist` Provide means to store the number list for entries.

```
115 \define@boolkey{glossaries.sty}{gls}{savenumberlist}[true]{%
116   \glssavenumberlistfalse}
```

`eautionumberlist`

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

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

```
118 \@gls@declareoption{seeautonumberlist}{%
119   \renewcommand*{\@glo@seeautonumberlist}{%
120     \def\@glo@prefix{\glsnextpages}%
121   }%
122 }
```

`\@gls@loadlong`

```
123 \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.

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



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

```

125 \IfFileExists{supertabular.sty}{%
126   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
127   \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.

```

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

```

`\@gls@loadlist`

```

129 \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.

```

130 \@gls@declareoption{nolist}{\renewcommand*{\@gls@loadlist}{}

```

`\@gls@loadtree`

```

131 \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.

```

132 \@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).

```

133 \@gls@declareoption{nostyles}{%
134   \renewcommand*{\@gls@loadlong}{}%
135   \renewcommand*{\@gls@loadsuper}{}%
136   \renewcommand*{\@gls@loadlist}{}%
137   \renewcommand*{\@gls@loadtree}{}%
138   \let\@glossary@default@style\relax
139 }

```

`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.)

```

140 \newcommand*{\glspostdescription}{%
141   \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
142 }

```

`nopostdot` Boolean option to suppress post description dot

```

143 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
144 \glsnopostdotfalse

```

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

```

145 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
146 \glsnogroupskipfalse

```

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

```
147 \define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
148 \ifclassloaded{memoir}
149 {%
150   \glsucmarktrue
151 }%
152 {%
153   \glsucmarkfalse
154 }
```

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.

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

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

```
157 \define@key{glossaries.sty}{counterwithin}{%
158   \renewcommand*{\@gls@counterwithin}{#1}%
159   \glsentrycountertrue
160 }
```

s@counterwithin The default value is no parent counter:

```
161 \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.

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

default@sorttype Initialise default sort for \printnoidxglossary

```
164 \newcommand*{\@gls@default@sorttype}{standard}
```

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

```
165 \define@choicekey{glossaries.sty}{sort}{standard,def,use,none}{%
166   \renewcommand*{\@gls@default@sorttype}{#1}%
167   \csname @gls@setupsort@#1\endcsname
168 }
```

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.

```

169 \newcommand*{\glsprestandardsort}[3]{%
170   \glsdosanitizesort
171 }

eck@sortallowed
172 \newcommand*{\@glo@check@sortallowed}[1]{%

upsort@standard  Set up the macros for default sorting.
173 \newcommand*{\@gls@setupsort@standard}{%
  Store entry information when it's defined.
174   \def\do@glo@storeentry{\@glo@storeentry}%
  No count register required for standard sort.
175   \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.)
176   \def\@gls@defsort##1##2{%
177     \ifx\@glo@sort\@glsdefaultsort
178       \let\@glo@sort\@glo@name
179     \fi
180     \let\glsdosanitizesort\@gls@sanitizesort
181     \glsprestandardsort{\@glo@sort}{##1}{##2}%
182     \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
183   }%
  Don't need to do anything when the entry is used.
184   \def\@gls@setsort##1{%
  This sort option is allowed with \makeglossaries and \makenoidxglossaries.
185   \let\@glo@check@sortallowed\@gobble
186 }
  Set standard sort as the default:
187 \@gls@setupsort@standard

lssortnumberfmt  Format the number used as the sort key by sort=def and sort=use. Defaults to six digit num-
bering.
188 \newcommand*\glssortnumberfmt[1]{%
189   \ifnum#1<100000 0\fi
190   \ifnum#1<10000 0\fi
191   \ifnum#1<1000 0\fi
192   \ifnum#1<100 0\fi
193   \ifnum#1<10 0\fi
194   \number#1%
195 }

s@setupsort@def  Set up the macros for order of definition sorting.
196 \newcommand*{\@gls@setupsort@def}{%

```

Store entry information when it's defined.

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

Defined count register associated with the glossary.

```
198 \def\@gls@defsortcount##1{%
199   \expandafter\global
200   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
201 }%
```

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

```
202 \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.

```
203   \ifcsundef{glossary@##1@sortcount}%
204   {\@gls@defsortcount{##1}}%
205   {}%
206   \expandafter\global\expandafter
207   \advance\csname glossary@##1@sortcount\endcsname by 1\relax
208   \expandafter\protected@xdef\csname glo@##2@sort\endcsname{%
209     \expandafter\glssortnumberfmt
210     {\csname glossary@##1@sortcount\endcsname}}%
211 }%
```

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

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

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

```
213 \let\@glo@check@sortallowed\@gobble
214 }
```

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

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

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

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

Defined count register associated with the glossary.

```
217 \def\@gls@defsortcount##1{%
218   \expandafter\global
219   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
220 }%
```

Initialise the sort key to empty.

```
221 \def\@gls@defsort##1##2{%
222   \expandafter\gdef\csname glo@##2@sort\endcsname{}%
223 }%
```

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

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

Get the parent, if one exists

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

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

```
226 \ifx\@glo@parent\@empty
```

```
227 \else
```

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

```
229 \fi
```

Set index information for this entry

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

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

```
232 \ifx\@gls@tmp\@empty
```

```
233 \expandafter\global\expandafter
```

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

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

```
236 \expandafter\glssortnumberfmt
```

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

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

```
239 \fi
```

```
240 }%
```

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

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

```
242 }
```

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

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

Don't store entry index information.

```
244 \def\do@glo@storeentry##1{}%
```

No count register required for standard sort.

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

Don't modify sort value.

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

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

```
248 }%
```

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

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

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

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

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

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

```
253 }
```

`\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.)

```
254 \newcommand*{\glsdefmain}{%
255   \if@gls@docloaded
256     \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
257   \else
258     \newglossary{main}{gls}{glo}{\glossaryname}%
259   \fi
```

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

```
260 \newcommand*{\gls@tr@set@main@toctitle}{%
261   \translatelet{\glossarytoctitle}{Glossary}%
262 }%
263 }
```

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`

```
264 \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`

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

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

```
266 \@gls@declareoption{nomain}{%
267   \let\glsdefaulttype\relax
268   \renewcommand*{\glsdefmain}{}%
269 }
```

`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.

```
270 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
271   \ifglsacronym
272     \renewcommand{\@gls@do@acronymsdef}{%
273       \DeclareAcronymList{acronym}%
274       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
275       \renewcommand*{\acronymtype}{acronym}%
276     }%
277   }
```

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

```
276 \newcommand*{\gls@tr@set@acronym@toctitle}{%
277   \translatelet{\glossarytoctitle}{Acronyms}%
278 }
```

```

278     }%
279 }%
280 \else
281   \let\@gls@do@acronymsdef\relax
282 \fi
283 }

```

`\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.

```

284 \AtBeginDocument{%
285   \ifglsacronym
286     \ifbool{glscompatible-3.07}%
287       {}%
288     {%
289       \providecommand*\printacronyms[1][ ]{%
290         \printglossary[type=\acronymtype,#1]}%
291     }%
292 \fi
293 }

```

`@do@acronymsdef` Set default value

```

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

```

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

```

295 \@gls@declareoption{acronyms}{%
296   \glsacronymtrue
297   \renewcommand*\@gls@do@acronymsdef{%
298     \DeclareAcronymList{acronym}%
299     \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
300     \renewcommand*\acronymtype{acronym}%

```

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

```

301   \newcommand*\@gls@tr@set@acronym@toctitle{%
302     \translatelet{\glossarytoctitle}{Acronyms}%
303   }%
304 }%
305 }

```

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

```

306 \newcommand*\@glsacronymlists{}

```

`dtoacronymlists`

```

307 \newcommand*\@addtoacronymlists[1]{%
308   \ifx\@glsacronymlists\@empty
309     \protected@xdef\@glsacronymlists{#1}%
310   \else
311     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
312   \fi
313 }

```

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

```
314 \newcommand*{\DeclareAcronymList}[1]{%
315   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
316 }
```

`\IfListOfAcronyms`

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

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

```
317 \newcommand{\glsIfListOfAcronyms}[1]{%
318   \edef\@do@gls@islistofacronyms{%
319     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
320   \@do@gls@islistofacronyms
321 }
```

Internal command requires label and list to be expanded:

```
322 \newcommand{\@gls@islistofacronyms}[4]{%
323   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
324     \def\@before{##1}\def\@after{##2}}%
325   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
326   \ifx\@after\@nnil
```

Not found

```
327   #4%
328   \else
```

Found

```
329   #3%
330   \fi
331 }
```

`\glsisacronymlist` Convenient boolean.

```
332 \newif\if@glsisacronymlist
```

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

```
333 \newcommand*{\chkisacronymlist}[1]{%
334   \glsIfListOfAcronyms{#1}%
335   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
336 }
```

`\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.)

```
337 \newcommand*{\SetAcronymLists}[1]{%
338   \renewcommand*{\@glsacronymlists}{#1}%
339 }
```



acronymlists

```
340 \define@key{glossaries.sty}{acronymlists}{%
341   \DeclareAcronymList{#1}%
342 }
```

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`

```
343 \newcommand{\glscounter}{page}
```

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

```
344 \define@key{glossaries.sty}{counter}{%
345   \renewcommand*\glscounter{#1}%
346 }
```

`gls@nohyperlist`

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

`lareNoHyperList`

```
348 \newcommand*\GlsDeclareNoHyperList[1]{%
349   \ifdefempty\@gls@nohyperlist
350   {%
351     \renewcommand*\@gls@nohyperlist{#1}%
352   }%
353   {%
354     \appto\@gls@nohyperlist{,#1}%
355   }%
356 }
```

`nohypertypes`

```
357 \define@key{glossaries.sty}{nohypertypes}{%
358   \GlsDeclareNoHyperList{#1}%
359 }
```

`ossariesWarning` Prints a warning message.

```
360 \newcommand*\GlossariesWarning[1]{%
361   \PackageWarning{glossaries}{#1}%
362 }
```

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

```
363 \newcommand*\GlossariesWarningNoLine[1]{%
364   \PackageWarningNoLine{glossaries}{#1}%
365 }
```

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

```
366 \newcommand{\glosortentrieswarning}{%
367 \typeout{Using TeX to sort glossary entries---this may
368 take a while}%
369 }
```

nowarn Define package option to suppress warnings

```
370 \@gls@declareoption{nowarn}{%
371 \if@gls@debug
372 \GlossariesWarning{Warnings can't be suppressed in debug mode}%
373 \else
374 \renewcommand*{\GlossariesWarning}[1]{}%
375 \renewcommand*{\GlossariesWarningNoLine}[1]{}%
376 \renewcommand*{\glosortentrieswarning}{}%
377 \fi
378 }
```

nonglossdefined Issue a warning if overriding \printglossary

```
379 \newcommand*{\@gls@warnnonglossdefined}{%
380 \GlossariesWarning{Overriding \string\printglossary}%
381 }
```

theglossdefined Issue a warning if overriding theglossary

```
382 \newcommand*{\@gls@warnontheglossdefined}{%
383 \GlossariesWarning{Overriding 'theglossary' environment}%
384 }
```

noredefwarn Suppress warning on redefinition of \printglossary

```
385 \@gls@declareoption{noredefwarn}{%
386 \renewcommand*{\@gls@warnnonglossdefined}{}%
387 \renewcommand*{\@gls@warnontheglossdefined}{}%
388 }
```

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

```
389 \newcommand*{\@gls@sanitizedesc}{%
390 }
```

lssetexpandfield

`\glssetexpandfield{\<field>}`

Sets field to always expand.

```
391 \newcommand*{\glssetexpandfield}[1]{%
```

```

392 \csdef{gls@assign@#1@field}##1##2{%
393   \@@gls@expand@field{##1}{#1}{##2}%
394 }%
395 }

```

setnoexpandfield `\glssetnoexpandfield{<field>}`

Sets field to never expand.

```

396 \newcommand*{\glssetnoexpandfield}[1]{%
397   \csdef{gls@assign@#1@field}##1##2{%
398     \@@gls@noexpand@field{##1}{#1}{##2}%
399   }%
400 }

```

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

```
401 \glssetexpandfield{type}
```

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

```
402 \glssetnoexpandfield{desc}
```

descplural@field

```
403 \glssetnoexpandfield{descplural}
```

ls@sanitizename

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

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

```
405 \glssetnoexpandfield{name}
```

@sanitizesymbol

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

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

```
407 \glssetnoexpandfield{symbol}
```

bolplural@field

```
408 \glssetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```

409 \newcommand*{\@gls@sanitizesort}{%
410   \ifglssanitizesort
411     \@@gls@sanitizesort
412   \else
413     \@@gls@nosanitizesort
414   \fi
415 }

```

ls@sanitizesort

```
416 \newcommand*\@@gls@sanitizesort{%
417   \@onelevel@sanitize\@glo@sort
418 }
```

@nosanitizesort

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

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

```
420 \newcommand*\@gls@noidx@sanitizesort{%
421   \ifdefvoid\@glo@sort
422   {}%
423   {%
424     \expandafter\@@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
425   }%
426 }
427 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
428   \def\@glo@sort{#1#2}%
429   \@onelevel@sanitize\@glo@sort
430 }
```

@nosanitizesort

```
431 \newcommand*\@@gls@noidx@nosanitizesort{%
432   \ifdefvoid\@glo@sort
433   {}%
434   {%
435     \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
436   }%
437 }
438 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
439   \bgroup
440     \glsnoidxstripaccents
441     \protected@xdef\@glo@sort{#1#2}%
442   \egroup
443   \let\@glo@sort\@glo@sort
444 }
```

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.

```
445 \newcommand*\glsnoidxstripaccents{%
446   \let\IeC\@firstofone
447   \let\'\@firstofone
448   \let\'\@firstofone
449   \let\~\@firstofone
450   \let\"@firstofone
451   \let\u\@firstofone
452   \let\t\@firstofone
```

```

453 \let\d\@firstofone
454 \let\r\@firstofone
455 \let\=\@firstofone
456 \let\.\@firstofone
457 \let\~\@firstofone
458 \let\v\@firstofone
459 \let\H\@firstofone
460 \let\c\@firstofone
461 \let\b\@firstofone

462 \let\a\@secondoftwo
463 \def\AE{AE}%
464 \def\ae{ae}%
465 \def\OE{OE}%
466 \def\oe{oe}%
467 \def\AA{AA}%
468 \def\aa{aa}%
469 \def\L{L}%
470 \def\l{l}%
471 \def\O{O}%
472 \def\o{o}%
473 \def\SS{SS}%
474 \def\ss{ss}%
475 \def\th{th}%

476 \def\TH{TH}%
477 \def\dh{dh}%
478 \def\DH{DH}%
479 }

```

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.

```

480 \define@boolkey[glS]{sanitize}{description}[true]{%
481   \GlossariesWarning{sanitize={description} package option deprecated}%
482   \ifglS@sanitize@description
483     \glSsetnoexpandfield{desc}%
484     \glSsetnoexpandfield{descplural}%
485   \else
486     \glSsetexpandfield{desc}%
487     \glSsetexpandfield{descplural}%
488   \fi
489 }

490 \define@boolkey[glS]{sanitize}{name}[true]{%
491   \GlossariesWarning{sanitize={name} package option deprecated}%
492   \ifglS@sanitize@name
493     \glSsetnoexpandfield{name}%
494   \else
495     \glSsetexpandfield{name}%
496   \fi

```

```

497 }
498 \define@boolkey[gl]{sanitize}{symbol}[true]{%
499   \GlossariesWarning{sanitize={symbol} package option deprecated}%
500   \ifgl@sanitize@symbol
501     \glsssetnoexpandfield{symbol}%
502     \glsssetnoexpandfield{symbolplural}%
503   \else
504     \glsssetexpandfield{symbol}%
505     \glsssetexpandfield{symbolplural}%
506   \fi
507 }

```

sanitizesort

```

508 \define@boolkey{glossaries.sty}[gl]{sanitizesort}[true]{%
509   \ifglssanitizesort
510     \glsssetnoexpandfield{sortvalue}%
511     \renewcommand*{\@gl@noidx@setsanitizesort}{%
512       \glssanitizesorttrue
513       \glsssetnoexpandfield{sortvalue}%
514     }%
515   \else
516     \glsssetexpandfield{sortvalue}%
517     \renewcommand*{\@gl@noidx@setsanitizesort}{%
518       \glssanitizesortfalse
519       \glsssetexpandfield{sortvalue}%
520     }%
521   \fi
522 }

```

Default setting:

```

523 \glssanitizesorttrue
524 \glsssetnoexpandfield{sortvalue}%

```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```

525 \newcommand*{\@gl@noidx@setsanitizesort}{%
526   \glssanitizesortfalse
527   \glsssetexpandfield{sortvalue}%
528 }

529 \define@choicekey[gl]{sanitize}{sort}{true,false}[true]{%
530   \setbool{glssanitizesort}{#1}%
531   \ifglssanitizesort
532     \glsssetnoexpandfield{sortvalue}%
533   \else
534     \glsssetexpandfield{sortvalue}%
535   \fi
536   \GlossariesWarning{sanitize={sort} package option
537     deprecated. Use sanitizesort instead}%
538 }

```

```

sanitize
539 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
540   \ifthenelse{\equal{#1}{none}}{%
541     {%
542       \GlossariesWarning{sanitize package option deprecated}%
543       \glssetexpandfield{name}%
544       \glssetexpandfield{symbol}%
545       \glssetexpandfield{symbolplural}%
546       \glssetexpandfield{desc}%
547       \glssetexpandfield{descplural}%
548     }%
549     {%
550       \setkeys[gls]{sanitize}{#1}%
551     }%
552 }

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

otranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator
554 \newcommand*\@gls@usetranslator{%
    polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
    polyglossia as well.
555   \@ifpackageloaded{polyglossia}%
556   {%
557     \let\glsifusetranslator\@secondoftwo
558   }%
559   {%
560     \@ifpackageloaded{babel}%
561     {%
562       \IfFileExists{translator.sty}%
563       {%
564         \RequirePackage{translator}%
565         \let\glsifusetranslator\@firstoftwo
566       }%
567     }%
568   }%
569   {}%
570 }%
571 }

dtranslatordict Checks if given translator dictionary has been loaded.
572 \newcommand{\glsifusedtranslatordict}[3]{%
573   \glsifusetranslator
574   {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%

```

```

575  {#3}%
576 }

```

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

```

577 \@gls@declareoption{nottranslate}{%
578   \glstranslatefalse
579   \let\@gls@usetranslator\relax
580   \let\glsifusetranslator\@secondoftwo
581 }

```

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

```

582 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
583 {true,false,babel}[true]%
584 {%
585   \ifcase\nr\relax
586     \glstranslatetrue
587     \renewcommand*\@gls@usetranslator{%
588       \@ifpackageloaded{polyglossia}%
589       {%
590         \let\glsifusetranslator\@secondoftwo
591       }%
592       {%
593         \@ifpackageloaded{babel}%
594         {%
595           \IfFileExists{translator.sty}%
596           {%
597             \RequirePackage{translator}%
598             \let\glsifusetranslator\@firstoftwo
599           }%
600           {}%
601         }%
602         {}%
603       }%
604     }%
605   \or
606     \glstranslatefalse
607     \let\@gls@usetranslator\relax
608     \let\glsifusetranslator\@secondoftwo
609   \or
610     \glstranslatetrue
611     \let\@gls@usetranslator\relax
612     \let\glsifusetranslator\@secondoftwo
613   \fi
614 }

```

Set the default value:

```

615 \glstranslatefalse
616 \let\glsifusetranslator\@secondoftwo
617 \@ifpackageloaded{translator}%

```



```

618 {%
619   \glstranslatetrue
620   \let\glsifusetranslator\@firstoftwo
621 }%
622 {%
623   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
624   {
625     \@ifpackageloaded{\gls@thissty}%
626     {%
627       \glstranslatetrue
628       \@endfortrue
629     }%
630   }%
631 }
632 }

```

**indexonlyfirst** Set whether to only index on first use.

```

633 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
634 \glsindexonlyfirstfalse

```

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

```

635 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
636 \glshyperfirsttrue

```

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

```

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

```

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

```

638 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}%
639 \ifbool{glsacrdescription}%
640 {}%
641 {%
642   \renewcommand*{\@gls@sanitizedesc}{}%
643 }%
644 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
645 }

```

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

```

646 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}%
647 \renewcommand*{\@gls@sanitizesymbol}{}%
648 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
649 }

```

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

```

650 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}%
651 \renewcommand*{\@gls@sanitizesymbol}{}%
652 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
653 }

```

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

```
654 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
655   \renewcommand*{\@gls@sanitizesymbol}{}}%
656   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
657 }
```

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

```
658 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
659   \renewcommand*{\@gls@sanitizesymbol}{}}%
660   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
661 }
```

**shortcuts** Define acronym shortcuts.

```
662 \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.

```
663 \newcommand*{\glsorder}{word}
```

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

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

**order**

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

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

```
667 \newif\ifglxindy
```

The default is `makeindex`:

```
668 \glxindyfalse
```

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

```
669 \@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.

```
670 \define@boolkey[glx]{xindy}{glsnumbers}[true]{%
671 \glx@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.)

```
672 \def\@xdy@main@language{\language}
```

Define key to set the language

```
673 \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.

```
674 \ifcsundef{inputencodingname}{%
675   \def\glS@codepage{}}{%
676   \def\glS@codepage{\inputencodingname}
677 }
```

Define a key to set the code page.

```
678 \define@key[glS]{xindy}{codepage}{\def\glS@codepage{#1}}
```

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

```
679 \define@key{glossaries.sty}{xindy}[]{%
680   \glSxindytrue
681   \setkeys[glS]{xindy}{#1}%
682 }
```

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

```
683 \@glS@declareoption{xindygloss}{%
684   \glSxindytrue
685 }
```

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

```
686 \@glS@declareoption{xindynoglsnumbers}{%
687   \glSxindytrue
688   \glS@xindy@glSnumbersfalse
689 }
```

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

```
690 \define@boolkey{glossaries.sty}[glS]{automake}[true]{%
691   \ifglSautomake
692     \renewcommand*{\@glS@doautomake}{%
693       \PackageError{glossaries}{You must use
694         \string\makeglossaries\space with automake=true}
695       {%
696         Either remove the automake=true setting or
697         add \string\makeglossaries\space to your document preamble.%
698       }%
699     }%
700   \else
701     \renewcommand*{\@glS@doautomake}{}%
702   \fi
703 }
704 \glSautomakefalse
```

@gls@doautomake

```
705 \newcommand*{\@gls@doautomake}{}
706 \AtEndDocument{\@gls@doautomake}
```

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

```
707 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
708   \ifglssavewrites
709     \renewcommand*{\glswritefiles}{\@glswritefiles}%
710   \else
711     \let\glswritefiles\@empty
712   \fi
713 }
```

Set default:

```
714 \glssavewritesfalse
715 \let\glswritefiles\@empty
```

compatible-3.07

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

compatible-2.07

```
718 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
  Also set 3.07 compatibility if this option is set.
719   \ifbool{glscompatible-2.07}%
720   {%
721     \booltrue{glscompatible-3.07}%
722   }%
723   {}%
724 }
725 \boolfalse{glscompatible-2.07}
```

symbols Create a “symbols” glossary type

```
726 \@gls@declareoption{symbols}{}
727 \let\@gls@do@symbolsdef\@gls@symbolsdef
728 }
```

Default is not to define the symbols glossary:

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

@gls@symbolsdef
```

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

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

```
733 \newcommand*{\gls@tr@set@symbols@toctitle}{}
734 \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
735 }%
736 }%
```

numbers Create a “symbols” glossary type

```
737 \@gls@declareoption{numbers}{%  
738   \let\@gls@do@numbersdef\@gls@numbersdef  
739 }
```

Default is not to define the numbers glossary:

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

@gls@numbersdef

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

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

```
744 \newcommand*{\gls@tr@set@numbers@toctitle}{%  
745   \translatelet{\glossarytoctitle}{Numbers (glossaries)}%  
746 }%  
747 }%
```

index Create an “index” glossary type

```
748 \@gls@declareoption{index}{%  
749   \let\@gls@do@indexdef\@gls@indexdef  
750 }
```

Default is not to define index glossary:

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

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

```
752 \newcommand*{\@gls@indexdef}{%  
753   \newglossary[ilg]{index}{ind}{idx}{\indexname}%  
754   \newcommand*{\printindex}[1] [] {\printglossary[type=index,##1]}%  
755   \newcommand*{\newterm}[2] [] {%  
756     \newglossaryentry{##2}%  
757     {type={index},name={##2},description={\nopostdesc},##1}}  
758 }%
```

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

```
759 \@for\CurrentOption :=\@declaredoptions\do{%  
760   \ifx\CurrentOption\@empty  
761     \else  
762       \@expandtwoargs  
763       \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%  
764       \ifin@  
765         \@use@ption  
766         \expandafter \let\csname ds@\CurrentOption\endcsname\@empty  
767       \fi  
768     \fi  
769 }
```

Now process options passed to the package:

```
770 \ProcessOptionsX
```

Load backward compatibility stuff:

```
771 \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:

```
772 \disable@keys{glossaries.sty}{compatible-2.07,%
```

```
773 xindy,xindygloss,xindynoglsnumbers,makeindex,%
```

```
774 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,nomain}
```

Now define `\setupglossaries`:

```
775 \newcommand*{\setupglossaries}[1]{%
```

```
776 \renewcommand*{\@gls@setacrstyle}{}%
```

```
777 \ifglsacrshortcuts
```

```
778 \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
```

```
779 \else
```

```
780 \def\@gls@setupshortcuts{%
```

```
781 \ifglsacrshortcuts
```

```
782 \DefineAcronymSynonyms
```

```
783 \fi
```

```
784 }%
```

```
785 \fi
```

```
786 \glsacrshortcutsfalse
```

```
787 \let\@gls@do@numbersdef\relax
```

```
788 \let\@gls@do@symbolssdef\relax
```

```
789 \let\@gls@do@indexdef\relax
```

```
790 \let\@gls@do@acronymsdef\relax
```

```
791 \setkeys{glossaries.sty}{#1}%
```

```
792 \@gls@setacrstyle
```

```
793 \@gls@setupshortcuts
```

```
794 \@gls@do@acronymsdef
```

```
795 \@gls@do@numbersdef
```

```
796 \@gls@do@symbolssdef
```

```
797 \@gls@do@indexdef
```

```
798 }
```

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

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

```
799 \ifthenelse{\equal{\glscounter}{section}}{%
```

```
800 {%
```

```
801 \ifcsundef{chapter}{}%
```

```
802 {%
```

```

803 \let\@gls@old@chapter\@chapter
804 \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
805 \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}{}}}%
806 }%
807 }%
808 {}

```

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

```

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

```

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

```

810 \newcommand*{\@onlypremakeg}[1]{%
811 \ifx\@gls@onlypremakeg\@empty
812 \def\@gls@onlypremakeg{#1}%
813 \else
814 \expandafter\toks@ \expandafter{\@gls@onlypremakeg}%
815 \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
816 \fi
817 }

```

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

```

818 \newcommand*{\@disable@onlypremakeg}{%
819 \@for\@thiscs:=\@gls@onlypremakeg\do{%
820 \expandafter\@disable@premakecs\@thiscs%
821 }}

```

`\sable@premakecs` Disables the given command.

```

822 \newcommand*{\@disable@premakecs}[1]{%
823 \def#1{\PackageError{glossaries}{\string#1\space may only be
824 used before \string\makeglossaries}{You can't use
825 \string#1\space after \string\makeglossaries}}%
826 }

```

## 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`

```

827 \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`  
828 `\providecommand*{\acronymname}{Acronyms}`

`\glsettoctitle` Sets the TOC title for the given glossary.  
829 `\newcommand*{\glsettoctitle}[1]{%`  
830 `\def\glossarytoctitle{\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`  
831 `\providecommand*{\entryname}{Notation}`

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

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

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

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

`\symbolsgroupname`  
835 `\providecommand*{\glsymbolsgroupname}{Symbols}`

`\numbersgroupname`  
836 `\providecommand*{\glnumbersgroupname}{Numbers}`

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

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

`\supacrpluralsuffix`  
839 `\newcommand*{\glsupacrpluralsuffix}{\glstextup{\glsacrpluralsuffix}}`

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

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

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



eGlossariesLang

```
842 \newcommand*{\RequireGlossariesLang}[1]{%
843   \@ifundefined{ver@glossaries-#1.1df}{\input{glossaries-#1.1df}}{}%
844 }
```

sGlossariesLang

```
845 \newcommand*{\ProvidesGlossariesLang}[1]{%
846   \ProvidesFile{glossaries-#1.1df}%
847 }
```

ssarytocaptions Does nothing if translator hasn't been loaded.

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

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

```
849 \ifglstranslate
```

Load tracklang

```
850 \RequirePackage{tracklang}
```

Load translator if required.

```
851 \@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.)

```
852 \@ifpackageloaded{translator}
```

```
853 {%
```

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.

```
854 \ifboolexpr
```

```
855 {
```

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

```
857   and not
```

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

```
859 }
```

```
860 {%
```

```
861 \let\glsifusetranslator\@secondoftwo
```

```
862 }%
```

```
863 {%
```

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

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

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

```
867   {%
```

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

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

```

870         \renewcommand*{\glossaryname}{\translate{Glossary}}}%
871     }%
872     \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
873 }%
874 }%
875 }%
876 }%
877 {}%

Check for tracked languages
878 \AnyTrackedLanguages
879 {%
880     \ForEachTrackedDialect{\this@dialect}{%
881         \IfTrackedLanguageFileExists{\this@dialect}%
882         {glossaries-}% prefix
883         {.ldf}%
884         {%
885             \RequireGlossariesLang{\CurrentTrackedTag}%
886         }%
887         {%
888             \PackageWarningNoLine{glossaries}%
889             {No language module detected for ‘\this@dialect’.\MessageBreak
890             Language modules need to be installed separately.\MessageBreak
891             Please check on CTAN for a bundle called\MessageBreak
892             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
893         }%
894     }%
895 }%
896 {}%

if using translator use translator interface.
897 \glsifusetranslator
898 {%
899     \renewcommand*{\glssettoctitle}[1]{%
900         \ifcsdef{gls@tr@set@#1@toctitle}%
901         {%
902             \csuse{gls@tr@set@#1@toctitle}%
903         }%
904         {%
905             \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
906         }%
907     }%
908     \renewcommand*{\glossaryname}{\translate{Glossary}}}%
909     \renewcommand*{\acronymname}{\translate{Acronyms}}}%
910     \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
911     \renewcommand*{\descriptionname}{%
912         \translate{Description (glossaries)}}}%
913     \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
914     \renewcommand*{\pagelistname}{%
915         \translate{Page List (glossaries)}}}%

```

```

916 \renewcommand*{\glssymbolsgroupname}{%
917 \translate{Symbols (glossaries)}}%
918 \renewcommand*{\glsnumbersgroupname}{%
919 \translate{Numbers (glossaries)}}%
920 }{}%
921 \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.

```
922 \DeclareRobustCommand*{\nopostdesc}{}

```

`\@nopostdesc` Suppress next description terminator.

```

923 \newcommand*{\@nopostdesc}{%
924 \let\org@glspostdescription\glspostdescription
925 \def\glspostdescription{%
926 \let\glspostdescription\org@glspostdescription}%
927 }

```

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

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

```

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

```
929 \newcommand{\glspar}{\par}

```

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

```

930 \newcommand{\setStyleFile}[1]{%
931 \renewcommand*{\gls@istfilebase}{#1}%
  Just in case \istfilename has been modified.
932 \ifglxindy
933 \def\istfilename{\gls@istfilebase.xdy}
934 \else
935 \def\istfilename{\gls@istfilebase.ist}
936 \fi
937 }

```

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

```
938 \@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`

```

939 \ifglxindy
940 \def\istfilename{\gls@istfilebase.xdy}
941 \else
942 \def\istfilename{\gls@istfilebase.ist}
943 \fi

```

gls@istfilebase

```
944 \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  $\text{\LaTeX}$ , `\@istfilename` ignores its argument.

`\@istfilename`

```
945 \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`

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

`lsSetCompositor` Sets the compositor.

```
947 \newcommand*{\glsSetCompositor}[1]{%
948   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
949 \@onlypremakeg\glsSetCompositor
```

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

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

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

`AlphaCompositor` Sets the alpha compositor.

```
951 \ifglsxindy
952   \newcommand*\glsSetAlphaCompositor[1]{%
953     \renewcommand*\@glsAlphacompositor{#1}}
954 \else
955   \newcommand*\glsSetAlphaCompositor[1]{%
956     \glsnxindywarning\glsSetAlphaCompositor}
957 \fi
```

Can only be used before `\makeglossaries`

```
958 \@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.

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

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

```

960 \newcommand*{\glsSetSuffixF}[1]{%
961   \renewcommand*{\gls@suffixF}{#1}}

```

Only has an effect when used before `\makeglossaries`

```

962 \@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.

```

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

```

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

```

964 \newcommand*{\glsSetSuffixFF}[1]{%
965   \renewcommand*{\gls@suffixFF}{#1}%
966 }

```

`\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".

```

967 \ifcsundef{hyperlink}%
968 {%
969   \newcommand*{\glsnumberformat}[1]{#1}%
970 }%
971 {%
972   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
973 }

```

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`

```

974 \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`

```

975 \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`

```
976 \newcommand*{\glossarypreamble}{%
977   \csuse{@glossarypreamble@\currentglossary}%
978 }
```

`\glossarypreamble`

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

Code provided by Michael Pock.

```
979 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
980   \ifglossaryexists{#1}{%
981     \csgdef{@glossarypreamble@#1}{#2}%
982   }{%
983     \GlossariesWarning{%
984       Glossary ‘#1’ is not defined%
985     }%
986   }%
987 }
```

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

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

`\glossarypostamble`

```
988 \newcommand*{\glossarypostamble}{}
```

`\glossarysection`

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

```
989 \newcommand*{\glossarysection}[2][\@gls@title]{%
990   \def\@gls@title{#2}%
991   \ifcsundef{phantomsection}%
992   {%
993     \@glossarysection{#1}{#2}%
994   }%
995   {%
996     \p@glossarysection{#1}{#2}%
997   }%
```

```

998 \glsglossarymark{\glossarytoctitle}%
999 }

```

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

```

1000 \ifcsundef{glossarymark}%
1001 {%
1002   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
1003 }%
1004 {%
1005   \@ifclassloaded{memoir}
1006   {%
1007     \newcommand{\glsglossarymark}[1]{%
1008       \ifglsucmark
1009         \markboth{\memUHead{#1}}{\memUHead{#1}}%
1010       \else
1011         \markboth{#1}{#1}%
1012       \fi
1013     }
1014   }%
1015   {%
1016     \newcommand{\glsglossarymark}[1]{%
1017       \ifglsucmark
1018         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1019       \else
1020         \@mkboth{#1}{#1}%
1021       \fi
1022     }
1023   }
1024 }

```

`\glossarymark` Provided for backward compatibility:

```

1025 \providecommand{\glossarymark}[1]{%
1026   \ifglsucmark
1027     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1028   \else
1029     \@mkboth{#1}{#1}%
1030   \fi
1031 }

```

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`

```

1032 \newcommand*{\setglossarysection}[1]{%
1033 \setkeys{glossaries.sty}{section=#1}}

```

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

glossarysection

```

1034 \newcommand*{\@glossarysection}[2]{%
1035   \ifdefempty\@glossarysecstar
1036   {%
1037     \csname\@glossarysec\endcsname[#1]{#2}%
1038   }%
1039   {%
1040     \csname\@glossarysec\endcsname*{#2}%
1041     \@gls@toc{#1}{\@glossarysec}%
1042   }%

  Do automatic labelling if required
1043   \@glossaryseclabel
1044 }
```

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

```

1045 \newcommand*{\@p@glossarysection}[2]{%
1046   \glsclearpage
1047   \phantomsection
1048   \ifdefempty\@glossarysecstar
1049   {%
1050     \csname\@glossarysec\endcsname{#2}%
1051   }%
1052   {%
1053     \@gls@toc{#1}{\@glossarysec}%
1054     \csname\@glossarysec\endcsname*{#2}%
1055   }%

  Do automatic labelling if required
1056   \@glossaryseclabel
1057 }
```

\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.

```

1058 \newcommand*{\gls@doclearpage}{%
1059   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1060     {%
1061       \ifcsundef{cleardoublepage}%
1062       {%
1063         \clearpage
1064       }%
1065     }%
1066     \ifcsdef{if@openright}%
1067     {%
1068       \if@openright
```



```

1069         \cleardoublepage
1070     \else
1071         \clearpage
1072     \fi
1073 }%
1074 {%
1075     \cleardoublepage
1076 }%
1077 }%
1078 }%
1079 {}%
1080 }

```

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

```

1081 \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`

```

1082 \newcommand*{\@gls@toc}[2]{%
1083     \ifglstoc
1084         \ifglsnumberline
1085             \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1086         \else
1087             \addcontentsline{toc}{#2}{#1}%
1088         \fi
1089     \fi
1090 }

```

## 1.4 Xindy

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

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

```

1091 \newcommand*{\glsnxindywarning}[1]{%
1092     \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1093 }

```

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

```

1094 \newcommand*{\glsnomakeindexwarning}[1]{%
1095     \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1096 }

```

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

```
1097 \ifglxsindy
1098   \edef\@xdyattributes{\string"default\string"}%
1099 \fi
```

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

```
1100 \ifglxsindy
1101   \edef\@dyattributelist{}%
1102 \fi
```

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

```
1103 \ifglxsindy
1104   \def\@xdylocref{}
1105 \fi
```

`\@gls@ifinlist`

```
1106 \newcommand*{\@gls@ifinlist}[4]{%
1107   \def\@do@ifinlist##1,#1,##2\end@ifinlist{%
1108     \def\@gls@listsuffix{##2}%
1109     \ifx\@gls@listsuffix\@empty
1110       #4%
1111     \else
1112       #3%
1113     \fi
1114   }%
1115   \@do@ifinlist,#2,#1,\end@ifinlist
1116 }
```

`\@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.

```
1117 \ifglxsindy
1118   \newcommand*{\@xdycounters}{\@glscounter}
1119   \newcommand*\GlsAddXdyCounters[1]{%
1120     \@for\@gls@ctr:=#1\do{%
1121       Check if already in list before adding.
1122       \edef\@do@addcounter{%
1123         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1124         \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1125           \noexpand\@gls@ctr}%
1126       }%
1127     }%
1128     \@do@addcounter
1129   }
1130 }
```

Only has an effect before `\writeist`:

```

1131 \onlypremakeg\GlsAddXdyCounters
1132 \else
1133 \newcommand*\GlsAddXdyCounters[1]{%
1134   \glsnoxindywarning\GlsAddXdyAttribute
1135 }
1136 \fi

```

`saddxdycounters` Counters must all be identified before adding attributes.

```

1137 \newcommand*\@disabled@glssaddxdycounters{%
1138   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1139     can't be used after \string\GlsAddXdyAttribute}{Move all
1140     occurrences of \string\GlsAddXdyCounters\space before the first
1141     instance of \string\GlsAddXdyAttribute}%
1142 }

```

`AddXdyAttribute` Adds an attribute.

```

1143 \ifglxsindy

```

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

```

1144 \newcommand*\@glssaddxdyattribute[2]{%

```

Add to xindy attribute list

```

1145   \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1146     \string"#2#1\string"}%

```

Add to xindy markup location.

```

1147   \expandafter\toks@\expandafter{\@xdylocref}%
1148   \edef\@xdylocref{\the\toks@ ^^J%
1149     (markup-locref
1150     :open \string"\glstildechar n%
1151       \expandafter\string\csname glsX#2X#1\endcsname
1152       \string" ^^J
1153     :close \string"\string" ^^J
1154     :attr \string"#2#1\string")}%

```

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

```

1155   \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1156     \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1157   }%
1158 }

```

High-level command:

```

1159 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

1160   \ifx\@xdyattributelist\@empty
1161     \edef\@xdyattributelist{#1}%
1162   \else
1163     \edef\@xdyattributelist{\@xdyattributelist,#1}%
1164   \fi

```

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

```

1165 \for\@this@counter:=\@xdycounters\do{%
1166 \protected@edef\gls@do@addxdyattribute{%
1167 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1168 }
1169 \gls@do@addxdyattribute
1170 }%

```

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

```

1171 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1172 }

```

Only has an effect before `\writeist`:

```

1173 \@onlypremakeg\GlsAddXdyAttribute
1174 \else
1175 \newcommand*\GlsAddXdyAttribute[1]{%
1176 \glsnoxywarning\GlsAddXdyAttribute}
1177 \fi

```

`definedattributes` Add known attributes for all defined counters

```

1178 \ifglxindy
1179 \newcommand*\@gls@addpredefinedattributes{%
1180 \GlsAddXdyAttribute{glsnumberformat}
1181 \GlsAddXdyAttribute{textrm}
1182 \GlsAddXdyAttribute{textsf}
1183 \GlsAddXdyAttribute{texttt}
1184 \GlsAddXdyAttribute{textbf}
1185 \GlsAddXdyAttribute{textmd}
1186 \GlsAddXdyAttribute{textit}
1187 \GlsAddXdyAttribute{textup}
1188 \GlsAddXdyAttribute{textsl}
1189 \GlsAddXdyAttribute{textsc}
1190 \GlsAddXdyAttribute{emph}
1191 \GlsAddXdyAttribute{glshypernumber}
1192 \GlsAddXdyAttribute{hyperrm}
1193 \GlsAddXdyAttribute{hypersf}
1194 \GlsAddXdyAttribute{hypertt}
1195 \GlsAddXdyAttribute{hyperbf}
1196 \GlsAddXdyAttribute{hypermd}
1197 \GlsAddXdyAttribute{hyperit}
1198 \GlsAddXdyAttribute{hyperup}
1199 \GlsAddXdyAttribute{hypersl}
1200 \GlsAddXdyAttribute{hypersc}
1201 \GlsAddXdyAttribute{hyperemph}

1202 \GlsAddXdyAttribute{glsignore}
1203 }
1204 \else
1205 \let\@gls@addpredefinedattributes\relax
1206 \fi

```

dyuseralphabets List of additional alphabets

```
1207 \def\@xdyuseralphabets{}
```

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

```
1208 \ifglxindy
1209   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1210     \edef\@xdyuseralphabets{%
1211       \@xdyuseralphabets ^^J
1212       (define-alphabet "#1" (#2))}%
1213   \else
1214     \newcommand*{\GlsAddXdyAlphabet}[2]{%
1215       \glsnnoxindywarning\GlsAddXdyAlphabet}
1216 \fi
```

This code is only required for xindy:

```
1217 \ifglxindy
```

dy@locationlist List of predefined location names.

```
1218   \newcommand*{\@gls@xdy@locationlist}{%
1219     roman-page-numbers,%
1220     Roman-page-numbers,%
1221     arabic-page-numbers,%
1222     alpha-page-numbers,%
1223     Alpha-page-numbers,%
1224     Appendix-page-numbers,%
1225     arabic-section-numbers%
1226   }
```

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

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

```
1227   \protected@edef\@gls@roman{\@roman{0}\string"
1228     \string"roman-numbers-lowercase\string" :sep \string"}}%
1229   \@onelevel@sanitize\@gls@roman
1230   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1231     :sep \string"}%
1232   \@onelevel@sanitize\@tmp
1233   \ifx\@tmp\@gls@roman
1234     \expandafter
1235       \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1236         \string"roman-numbers-lowercase\string"%
1237       }%
1238   \else
1239     \expandafter
```

```

1240     \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1241         :sep \string"\@gls@roman\string"%
1242     }%
1243 \fi

an-page-numbers Upper case Roman numerals (I, II, ...).
1244 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1245     \string"roman-numbers-uppercase\string"%
1246 }%

ic-page-numbers Arabic numbers (1, 2, ...).
1247 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1248     \string"arabic-numbers\string"%
1249 }%

ha-page-numbers Lower case alphabetical (a, b, ...).
1250 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1251     \string"alpha\string"%
1252 }%

ha-page-numbers Upper case alphabetical (A, B, ...).
1253 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1254     \string"ALPHA\string"%
1255 }%

ix-page-numbers Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given by
\@glsAlphacompositor.
1256 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1257     \string"ALPHA\string"
1258     :sep \string"\@glsAlphacompositor\string"
1259     \string"arabic-numbers\string"%
1260 }

section-numbers Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by \glscompositor.
1261 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1262     \string"arabic-numbers\string"
1263     :sep \string"\glscompositor\string"
1264     \string"arabic-numbers\string"%
1265 }%

erlocationdefs List of additional location definitions (separated by ^^J)
1266 \def\@xdyuserlocationdefs{}

erlocationnames List of additional user location names
1267 \def\@xdyuserlocationnames{}

End of xindy-only block:
1268 \fi

```

`\xdycrossrefhook` Hook used after writing cross-reference class information.

```
1269 \ifglxindy
1270 \newcommand\@xdycrossrefhook{}
1271 \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.)

```
1272 \ifglxindy
1273 \newcommand*\GlsAddXdyLocation[3][{}]{%
1274 \def\@gls@tmp{#1}%
1275 \ifx\@gls@tmp\@empty
1276 \edef\@xdyuserlocationdefs{%
1277 \@xdyuserlocationdefs ^^J%
1278 (define-location-class \string"#2\string"^^J\space\space
1279 \space(:sep \string"{}\glsopenbrace\string" #3
1280 :sep \string"\glsclosebrace\string"))
1281 }%
1282 \else
1283 \edef\@xdyuserlocationdefs{%
1284 \@xdyuserlocationdefs ^^J%
1285 (define-location-class \string"#2\string"^^J\space\space
1286 \space(:sep "\glsopenbrace"
1287 #1
1288 :sep "\glsclosebrace\glsopenbrace" #3
1289 :sep "\glsclosebrace"))
1290 }%
1291 \fi

1292 \edef\@xdyuserlocationnames{%
1293 \@xdyuserlocationnames^^J\space\space\space
1294 \string"#2\string"}%
1295 }
```

Only has an effect before `\writeist`:

```
1296 \@onlypremakeg\GlsAddXdyLocation
1297 \else
1298 \newcommand*\GlsAddXdyLocation[2]{%
1299 \glsnoxindywarning\GlsAddXdyLocation}
1300 \fi
```

`\locationclassorder` Define location class order

```
1301 \ifglxindy
1302 \def\@xdylocationclassorder{^^J\space\space\space
1303 \string"roman-page-numbers\string"^^J\space\space\space
1304 \string"arabic-page-numbers\string"^^J\space\space\space
1305 \string"arabic-section-numbers\string"^^J\space\space\space
1306 \string"alpha-page-numbers\string"^^J\space\space\space
1307 \string"Roman-page-numbers\string"^^J\space\space\space
```

```

1308   \string"Alpha-page-numbers\string"^^J\space\space\space
1309   \string"Appendix-page-numbers\string"
1310   \@xdyuserlocationnames^^J\space\space\space
1311   \string"see\string"
1312   }
1313 \fi

```

Change the location order.

ationClassOrder

```

1314 \ifglxindy
1315   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1316     \def\@xdylocationclassorder{#1}}
1317 \else
1318   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1319     \glsnxindywarning\GlsSetXdyLocationClassOrder}
1320 \fi

```

\@xdysortrules Define sort rules

```

1321 \ifglxindy
1322   \def\@xdysortrules{}
1323 \fi

```

\GlsAddSortRule Add a sort rule

```

1324 \ifglxindy
1325   \newcommand*\GlsAddSortRule[2]{%
1326     \expandafter\toks@\expandafter{\@xdysortrules}%
1327     \protected@edef\@xdysortrules{\the\toks@ ^^J
1328       (sort-rule \string"#1\string" \string"#2\string")}%
1329   }
1330 \else
1331   \newcommand*\GlsAddSortRule[2]{%
1332     \glsnxindywarning\GlsAddSortRule}
1333 \fi

```

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

```

1334 \ifglxindy
1335   \def\@xdyrequiredstyles{tex}
1336 \fi

```

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

```

1337 \ifglxindy
1338   \newcommand*\GlsAddXdyStyle[1]{%
1339     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1340 \else
1341   \newcommand*\GlsAddXdyStyle[1]{%
1342     \glsnxindywarning\GlsAddXdyStyle}
1343 \fi

```



**GlsSetXdyStyles**    Reset the list of required styles

```
1344 \ifglxindy
1345   \newcommand*\GlsSetXdyStyles[1]{%
1346     \edef\@xdyrequiredstyles{#1}}
1347 \else
1348   \newcommand*\GlsSetXdyStyles[1]{%
1349     \glsnxindywarning\GlsSetXdyStyles}
1350 \fi
```

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

```
1351 \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.

```
1352 \def\@xdylanguage#1#2{}
```

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

```
1353 \ifglxindy
1354   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1355     \ifglossaryexists{#1}{%
1356       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1357     }{%
1358       \PackageError{glossaries}{Can't set language type for
1359         glossary type '#1' --- no such glossary}{%
1360         You have specified a glossary type that doesn't exist}}
1361 \else
1362   \newcommand*\GlsSetXdyLanguage[2][]{%
1363     \glsnxindywarning\GlsSetXdyLanguage}
1364 \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.

```
1365 \def\@gls@codepage#1#2{}
```

**sSetXdyCodePage**    Define command to set the code page.

```
1366 \ifglxindy
1367   \newcommand*\GlsSetXdyCodePage[1]{%
1368     \renewcommand*\@gls@codepage{#1}%
1369   }
```

    Suggested by egreg:

```
1370 \AtBeginDocument{%
```

```

1371 \ifx\gls@codepage\@empty
1372 \ifpackage{fontspec}{\def\gls@codepage{utf8}}{}%
1373 \fi
1374 }
1375 \else
1376 \newcommand*\GlsSetXdyCodePage[1]{%
1377 \glsnoxywarning\GlsSetXdyCodePage}
1378 \fi

```

**xdylettergroups** Store letter group definitions.

```

1379 \ifglxindy
1380 \ifglxindy@glnumbers
1381 \def\@xdylettergroups{(define-letter-group
1382 \string\glnumbers\string"^^J\space\space\space
1383 :prefixes (\string"0\string" \string"1\string"
1384 \string"2\string" \string"3\string" \string"4\string"
1385 \string"5\string" \string"6\string" \string"7\string"
1386 \string"8\string" \string"9\string")^^J\space\space\space
1387 :before \string"@glsfirstletter\string")}
1388 \else
1389 \def\@xdylettergroups{}
1390 \fi
1391 \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.

```

1392 \newcommand*\GlsAddLetterGroup[2]{%
1393 \expandafter\toks@\expandafter{\@xdylettergroups}%
1394 \protected@edef\@xdylettergroups{\the\toks@^^J%
1395 (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1396 }%

```

## 1.5 Loops and conditionals

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

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

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

```

1397 \newcommand*\forallglossaries[3][\@glo@types]{%
1398 \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1399 }

```

**\forallacronyms**

```
1400 \newcommand*\forallacronyms[2]{%
```

```

1401 \for#1:=\glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1402 }

```

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

```
\forlgsentries[⟨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.

```

1403 \newcommand*{\forlgsentries}[3][\glsdefaulttype]{%
1404   \edef\@glo@list{\csname glolist@#1\endcsname}%
1405   \for#2:=\@glo@list\do
1406     {%
1407       \ifdefempty{#2}{#3}%
1408     }%
1409 }

```

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

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

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

```

1410 \newcommand*{\foralllgsentries}[3][\@glo@types]{%
1411   \expandafter\foralllglossaries\expandafter[#1]{\@this@glo@}%
1412   {%
1413     \forlgsentries[\@this@glo@]{#2}{#3}%
1414   }%
1415 }

```

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

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

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

```

1416 \newcommand{\ifglossaryexists}[3]{%
1417   \ifcsundef{glo@type@#1@out}{#3}{#2}%
1418 }

```

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`

```
1419 \newcommand*{\glsdetoklabel}[1]{#1}
```

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

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

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

```
1420 \newcommand{\ifglentryexists}[3]{%
1421   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1422 }
```

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

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

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

```
1423 \newcommand*{\ifglsused}[3]{%
1424   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1425 }
```

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

`\glsdoifexists`

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

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

```
1426 \newcommand{\glsdoifexists}[2]{%
1427   \ifglentryexists{#1}{#2}{%
1428     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1429     has not been defined}{You need to define a glossary entry before you
1430     can use it.}}%
1431 }
```

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

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

```
1432 \newcommand{\glsdoifnoexists}[2]{%
1433   \ifglentryexists{#1}{%
1434     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1435     been defined}{}}{#2}%
1436 }
```

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

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

```

1437 \newcommand{\glsdoifexistsorwarn}[2]{%
1438   \ifglentryexists{#1}{#2}{%
1439     \GlossariesWarning{Glossary entry ‘\glsdetoklabel{#1}’
1440       has not been defined}%
1441   }%
1442 }
```

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

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

```

1443 \newcommand{\glsdoifexistsordo}[3]{%
1444   \ifglentryexists{#1}{#2}{%
1445     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1446       has not been defined}{You need to define a glossary entry before you
1447       can use it.}%
1448     #3%
1449   }%
1450 }
```

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

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

```

1451 \newcommand{\doifglossarynoexistsordo}[3]{%
1452   \ifglossaryexists{#1}%
1453   {%
1454     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1455     #3%
1456   }%
1457   {#2}%
1458 }
```

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

```

1459 \newcommand{\ifglshaschildren}[3]{%
1460   \glsdoifexists{#1}%
1461   {%
1462     \def\do@glshaschildren{#3}%
1463     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1464     \expandafter\forglentries\expandafter
1465     [\csname glo@\@gls@thislabel @type\endcsname]
1466     {\glo@label}%

```

```

1467    {%
1468      \letcs\glo@parent{glo@\glo@label @parent}%
1469      \ifdefequal\@gls@thislabel\glo@parent
1470      {%
1471        \def\do@glshaschildren{#2}%
1472        \@endfortrue
1473      }%
1474    }%
1475  }%
1476  \do@glshaschildren
1477 }%
1478 }

```

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

```

1479 \newcommand{\ifglshasparent}[3]{%
1480   \glsdoifexists{#1}%
1481   {%
1482     \ifcsequal{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1483   }%
1484 }

```

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

```

1485 \newcommand*{\ifglshasdesc}[3]{%
1486   \ifcsequal{glo@\glsdetoklabel{#1}@desc}%
1487   {#3}%
1488   {#2}%
1489 }

```

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

```

1490 \newcommand*{\ifglsdscsuppressed}[3]{%
1491   \ifcsequal{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1492   {#2}%
1493   {#3}%
1494 }

```

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

```

1495 \newcommand*{\ifglshassymbol}[3]{%
1496   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1497   \ifdefempty\@glo@symbol
1498   {#3}%
1499   {%
1500     \ifdefequal\@glo@symbol\@gls@default@value
1501     {#3}%
1502     {#2}%
1503   }%

```

1504 }

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

```
1505 \newcommand*{\ifglshaslong}[3]{%
1506   \letcs{\@glo@long}{glo@glstdetoklabel{#1}@long}%
1507   \ifdefempty\@glo@long
1508     {#3}%
1509     {%
1510       \ifdefequal\@glo@long\@gls@default@value
1511         {#3}%
1512         {#2}%
1513     }%
1514 }
```

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

```
1515 \newcommand*{\ifglshasshort}[3]{%
1516   \letcs{\@glo@short}{glo@glstdetoklabel{#1}@short}%
1517   \ifdefempty\@glo@short
1518     {#3}%
1519     {%
1520       \ifdefequal\@glo@short\@gls@default@value
1521         {#3}%
1522         {#2}%
1523     }%
1524 }
```

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

```
1525 \newcommand*{\ifglshasfield}[4]{%
1526   \glsdoifexists{#2}%
1527   {%
1528     \letcs{\@glo@thisvalue}{glo@glstdetoklabel{#2}@#1}%

```

First check supplied field label is defined.

```
1529   \ifdef\@glo@thisvalue
1530     {%

```

Is defined, so now check if empty.

```
1531     \ifdefempty\@glo@thisvalue
1532     {%

```

Is empty, so doesn't have field set.

```
1533       #4%
1534     }%
1535     {%

```

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

```
1536       \ifdefequal\@glo@thisvalue\@gls@default@value
1537       {%

```

Value is set to the default value.

```
1538         #4%
1539     }%
1540     {%
```

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

```
1541     \let\glscurrentfieldvalue\@glo@thisvalue
1542     #3%
1543 }%
1544 }%
1545 }%
1546 {%
```

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

```
1547     \@gls@fetchfield{\@gls@thisfield}{#1}%
```

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

```
1548     \ifdef\@gls@thisfield
1549     {%
```

Is defined, so now check if empty.

```
1550     \letcs{\@glo@thisvalue}{glo@\glsdetoklabel{#2}@\@gls@thisfield}%
1551     \ifdefempty\@glo@thisvalue
1552     {%
```

Is empty so field hasn't been set.

```
1553         #4%
1554     }%
1555     {%
```

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

```
1556     \ifdequal\@glo@thisvalue\@gls@default@value
1557     {%
```

Value is set to the default value.

```
1558         #4%
1559     }%
1560     {%
```

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

```
1561     \let\glscurrentfieldvalue\@glo@thisvalue
1562     #3%
1563 }%
1564 }%
1565 }%
1566 {%
```

Not defined.

```
1567     \GlossariesWarning{Unknown entry field '#1'}%
1568     #4%
```



```

1569     }%
1570   }%
1571 }%
1572 }

```

urrentfieldvalue

```

1573 \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`

```

1574 \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.

```

1575 \newcommand*\@gls@provide@newglossary{%
1576   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}%

```

Only need to do this once.

```

1577   \let\@gls@provide@newglossary\relax
1578 }

```

\defglsentryfmt Allow different glossaries to have different display styles.

```

1579 \newcommand*{\defglsentryfmt}[2][\glsdefaulttype]{%
1580   \csgdef{gls@#1@entryfmt}{#2}%
1581 }

```

\gls@doentryfmt

```

1582 \newcommand*{\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}

```

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

```

1583 \newcommand*{\@gls@forbidtexext}[1]{%
1584   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1585             or test {\ifdefstring{#1}{TEX}}}
1586   {%
1587     \def#1{nottex}%
1588     \PackageError{glossaries}%
1589       {Forbidden '.tex' extension replaced with '.nottex'}%
1590       {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1591         Don't use '.tex' as an extension for a temporary file.}%
1592   }%

```

```

1593 {%
1594 }%
1595 }

```

`\gls@gobbleopt` Discard optional argument.

```

1596 \newcommand*{\gls@gobbleopt}{\new@ifnextchar[{\@gls@gobbleopt}{}]}
1597 \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`

```

1598 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}

```

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

```

1599 \newcommand*{\s@newglossary}[2]{%
1600 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
1601 }

```

`\ns@newglossary` Define the unstarred version.

```

1602 \newcommand*{\ns@newglossary}[5][glg]{%
1603 \doifglossarynoexistsordo{#2}%
1604 {%

```

Check if default has been set

```

1605 \ifundef\glsdefaulttype
1606 {%
1607 \gdef\glsdefaulttype{#2}%
1608 }{}%

```

Add this to the list of glossary types:

```

1609 \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.

```

1610 \expandafter\gdef\csname glolist@#2\endcsname{,}%

```

Store the file extensions:

```

1611 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
1612 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
1613 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
1614 \expandafter\@gls@forbidtextext\csname @glotype@#2@log\endcsname
1615 \expandafter\@gls@forbidtextext\csname @glotype@#2@in\endcsname
1616 \expandafter\@gls@forbidtextext\csname @glotype@#2@out\endcsname

```

Store the title:

```

1617 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%

1618 \@gls@provide@newglossary
1619 \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.

```

1620 \ifcsundef{gls@#2@entryfmt}%
1621 {%
1622   \defglsentryfmt[#2]{\glsentryfmt}%
1623 }%
1624 {}%

```

Define sort counter if required:

```

1625 \@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.)

```

1626 \@ifnextchar[{\@gls@setcounter{#2}}%
1627   {\@gls@setcounter{#2}[\glscounter]}%
1628 }%
1629 {%
1630   \gls@gobbleopt
1631 }%
1632 }

```

`\altnewglossary`

```

1633 \newcommand*{\altnewglossary}[3]{%
1634   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1635 }

```

Only define new glossaries in the preamble:

```

1636 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1637 \@onlypremakeg\newglossary

```

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

```

\@newglossary
1638 \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):

@glsetcounter

1639 \def\@glsetcounter#1[#2]{%
1640   \expandafter\def\csname @glotype@#1@counter\endcsname{#2}%

Add counter to xindy list, if not already added:

1641   \ifglxindy
1642     \GlsAddXdyCounters{#2}%
1643   \fi
1644 }

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

@glgetcounter

1645 \newcommand*{\@glgetcounter}[1]{%
1646   \csname @glotype@#1@counter\endcsname
1647 }

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

1648 \gldefmain

Define the “acronym” glossaries if required.

1649 \@gl@do@acronymsdef

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

1650 \@gl@do@symbolsdef
1651 \@gl@do@numbersdef
1652 \@gl@do@indexdef

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

1653 \newcommand*{\newignoredglossary}[1]{%
1654   \ifdefempty\@ignored@glossaries
1655   {%
1656     \edef\@ignored@glossaries{#1}%
1657   }%
1658   {%
1659     \eappto\@ignored@glossaries{, #1}%
1660   }%
1661   \csgdef{glolist@#1}{,}%
1662   \ifcsundef{gls@#1@entryfmt}%
1663   {%

```

```

1664 \defglentryfmt[#1]{\glentryfmt}%
1665 }%
1666 {}%
1667 \ifdefempty\@gls@nohyperlist
1668 {%
1669 \renewcommand*\@gls@nohyperlist}{#1}%
1670 }%
1671 {%
1672 \eappto\@gls@nohyperlist{,#1}%
1673 }%
1674 }

```

`ignored@glossaries` List of ignored glossaries.

```

1675 \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.

```

1676 \newcommand*\ifignoredglossary}[3]{%
1677 \edef\@gls@igtype{#1}%
1678 \expandafter\DTLifinlist\expandafter
1679 {\@gls@igtype}\@ignored@glossaries}{#2}{#3}%
1680 }

```

## 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.

```

1681 \define@key{glossentry}{name}{%
1682 \def\@glo@name{#1}%
1683 }

```

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

```

1684 \define@key{glossentry}{description}{%
1685 \def\@glo@desc{#1}%
1686 }

```

descriptionplural

```
1687 \define@key{glossentry}{descriptionplural}{%
1688 \def\@glo@descplural{#1}%
1689 }
```

**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>*.

```
1690 \define@key{glossentry}{sort}{%
1691 \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.

```
1692 \define@key{glossentry}{text}{%
1693 \def\@glo@text{#1}%
1694 }
```

**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.

```
1695 \define@key{glossentry}{plural}{%
1696 \def\@glo@plural{#1}%
1697 }
```

**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.

```
1698 \define@key{glossentry}{first}{%
1699 \def\@glo@first{#1}%
1700 }
```

**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.

```
1701 \define@key{glossentry}{firstplural}{%
1702 \def\@glo@firstplural{#1}%
1703 }
```

s@default@value

```
1704 \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).

```

1705 \define@key{glossentry}{symbol}{%
1706 \def\@glo@symbol{#1}%
1707 }

```

symbolplural

```

1708 \define@key{glossentry}{symbolplural}{%
1709 \def\@glo@symbolplural{#1}%
1710 }

```

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

```

1711 \define@key{glossentry}{type}{%
1712 \def\@glo@type{#1}}

```

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

```

1713 \define@key{glossentry}{counter}{%
1714   \ifcsundef{c@#1}%
1715   {%
1716     \PackageError{glossaries}%
1717     {There is no counter called ‘#1’}%
1718     {%
1719       The counter key should have the name of a valid counter
1720       as its value%
1721     }%
1722   }%
1723   {%
1724     \def\@glo@counter{#1}%
1725   }%
1726 }

```

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

```

1727 \define@key{glossentry}{see}{%
1728   \gls@set@xr@key{see}{\@glo@see{#1}%
1729 }

```

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

Assign a cross-reference key.

```

1730 \newcommand*{\gls@set@xr@key}[3]{%
1731   \renewcommand*{\gls@xr@key}{#1}%
1732   \gls@checkseeallowed
1733   \def#2{#3}%
1734   \@glo@seeautonumberlist
1735 }

```

`\gls@xr@key`

```

1736 \newcommand*{\gls@xr@key}{see}

```

checkseeallowed

```
1737 \newcommand*{\gls@checkseeallowed}{%
1738   \@gls@see@noindex
1739 }
```

ed@preambleonly

```
1740 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1741   \GlossariesWarning{glossaries}%
1742   {'\gls@xr@key' key doesn't have any effect when used in the document
1743   environment. Move the definition to the preamble
1744   after \string\makeglossaries\space
1745   or \string\makenoidxglossaries}%
1746 }
```

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

```
1747 \define@key{glossentry}{parent}{%
1748 \def\@glo@parent{#1}}
```

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

```
1749 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1750   \ifcase\nr\relax
1751     \def\@glo@prefix{\glsnonextpages}%
1752     \@gls@savenonumberlist{true}%
1753   \else
1754     \def\@glo@prefix{\glsnextpages}%
1755     \@gls@savenonumberlist{false}%
1756   \fi
1757 }
```

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.

```
1758 \newcommand*{\@gls@savenonumberlist}[1]{}
```

initnonumberlist

```
1759 \newcommand*{\@gls@initnonumberlist}{}%
```

nitnonumberlist

```
1760 \newcommand*{\@gls@storenonumberlist}[1]{}
```

savenonumberlist Allow the nonumberlist value to be saved.

```
1761 \newcommand*{\@gls@enablesavenonumberlist}{%
1762   \renewcommand*{\@gls@initnonumberlist}{%
1763     \undef\@glo@nonumberlist
1764   }%
1765   \renewcommand*{\@gls@savenonumberlist}[1]{%
```



```

1766 \def\@glo@nonumberlist{##1}%
1767 }%
1768 \renewcommand*{\@gls@storenonumberlist}[1]{%
1769 \ifdef\@glo@nonumberlist
1770 {%
1771 \cslet{glo@glstetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
1772 }%
1773 }%
1774 }%
1775 \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1776 }

```

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

user1

```

1777 \define@key{glossentry}{user1}{%
1778 \def\@glo@useri{#1}%
1779 }

```

user2

```

1780 \define@key{glossentry}{user2}{%
1781 \def\@glo@userii{#1}%
1782 }

```

user3

```

1783 \define@key{glossentry}{user3}{%
1784 \def\@glo@useriii{#1}%
1785 }

```

user4

```

1786 \define@key{glossentry}{user4}{%
1787 \def\@glo@useriv{#1}%
1788 }

```

user5

```

1789 \define@key{glossentry}{user5}{%
1790 \def\@glo@userv{#1}%
1791 }

```

user6

```

1792 \define@key{glossentry}{user6}{%
1793 \def\@glo@uservi{#1}%
1794 }

```

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.

```

1795 \define@key{glossentry}{short}{%
1796 \def\@glo@short{#1}%
1797 }

```

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

```
1798 \define@key{glossentry}{shortplural}{%
1799   \def\@glo@shortpl{#1}%
1800 }
```

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

```
1801 \define@key{glossentry}{long}{%
1802   \def\@glo@long{#1}%
1803 }
```

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

```
1804 \define@key{glossentry}{longplural}{%
1805   \def\@glo@longpl{#1}%
1806 }
```

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

```
1807 \newcommand*{\@glsnoname}{%
1808   \PackageError{glossaries}{name key required in
1809     \string\newglossaryentry\space for entry '\@glo@label'}{You
1810     haven't specified the entry name}}
```

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

```
1811 \newcommand*{\@glsnodels}{%
1812   \PackageError{glossaries}
1813   {%
1814     description key required in \string\newglossaryentry\space
1815     for entry '\@glo@label'%
1816   }%
1817   {%
1818     You haven't specified the entry description%
1819   }%
1820 }
```

`lsdefaultplural` Now obsolete. Don't use.

```
1821 \newcommand*{\@glsdefaultplural}{}
```

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

```
1822 \newcommand*{\@gls@missingnumberlist}[1]{%
1823   ??%
1824   \ifglssavenumberlist
1825     \GlossariesWarning{Missing number list for entry '#1'.
1826       Maybe makeglossaries + rerun required}%
1827   \else
1828     \PackageError{glossaries}%
1829     {Package option 'savenumberlist=true' required}%
1830     {%
1831       You must use the 'savenumberlist' package option
1832       to reference location lists.%
1833     }%
1834 }
```

```

1833 }%
1834 \fi
1835 }

```

`@glsdefaultsort` Define command to set default sort.

```
1836 \newcommand*{\@glsdefaultsort}{\@glo@name}
```

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

```
1837 \newcount\gls@level
```

`@noexpand@field`

```

1838 \newcommand{\@@gls@noexpand@field}[3]{%
1839 \expandafter\global\expandafter
1840 \let\csname glo@#1@#2\endcsname#3%
1841 }

```

`noexpand@fields`

```

1842 \newcommand{\@gls@noexpand@fields}[4]{%
1843 \ifcsdef{gls@assign@#3@field}
1844 {%
1845 \ifdefequal{#4}{\@gls@default@value}%
1846 {%
1847 \edef\@gls@value{\expandonce{#1}}%
1848 \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1849 }%
1850 {%
1851 \csuse{gls@assign@#3@field}{#2}{#4}%
1852 }%
1853 }%
1854 {%
1855 \ifdefequal{#4}{\@gls@default@value}%
1856 {%
1857 \edef\@gls@value{\expandonce{#1}}%
1858 \@@gls@noexpand@field{#2}{#3}{\@gls@value}%
1859 }%
1860 {%
1861 \@@gls@noexpand@field{#2}{#3}{#4}%
1862 }%
1863 }%
1864 }

```

`ls@expand@field`

```

1865 \newcommand{\@@gls@expand@field}[3]{%
1866 \expandafter
1867 \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1868 }

```

s@expand@fields

```

1869 \newcommand{\@gls@expand@fields}[4]{%
1870   \ifcsdef{gls@assign@#3@field}
1871   {%
1872     \ifdefequal{#4}{\@gls@default@value}%
1873     {%
1874       \edef\@gls@value{\expandonce{#1}}%
1875       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1876     }%
1877   {%
1878     \expandafter\@gls@startswithexpandonce#4\relax\relax\gls@endcheck
1879     {%
1880       \@gls@expand@field{#2}{#3}{#4}%
1881     }%
1882     {%
1883       \csuse{gls@assign@#3@field}{#2}{#4}%
1884     }%
1885   }%
1886 }%
1887 {%
1888   \ifdefequal{#4}{\@gls@default@value}%
1889   {%
1890     \@gls@expand@field{#2}{#3}{#1}%
1891   }%
1892   {%
1893     \@gls@expand@field{#2}{#3}{#4}%
1894   }%
1895 }%
1896 }

```

swithexpandonce

```

1897 \def\@gls@expandonce{\expandonce}
1898 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1899   \def\@gls@tmp{#1}%
1900   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1901 }

```

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.

```
1902 \let\gls@assign@field\@gls@expand@fields
```

glsexpandfields

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

```
1903 \newcommand*{\glsexpandfields}{%
```

```

1904 \let\gls@assign@field\@gls@expand@fields
1905 }

snoexpandfields Don't expand values when assigning fields (except for specific fields that are overridden by
\glssetexpandfield).
1906 \newcommand*{\glsnoexpandfields}{%
1907 \let\gls@assign@field\@gls@noexpand@fields
1908 }

ewglossaryentry Define \newglossaryentry {<label>} {<key-val list>}. There are two required fields in
<key-val list>: name (or parent) and description. (See above.)
1909 \newrobustcmd{\newglossaryentry}[2]{%
    Check to see if this glossary entry has already been defined:
1910 \glsdoifnoexists{#1}%
1911 {%
1912 \gls@defglossaryentry{#1}{#2}%
1913 }%
1914 }

ewglossaryentry 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.
1915 \newcommand*{\gls@defdocnewglossaryentry}{%
1916 \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1917 \let\newglossaryentry\new@glossaryentry
1918 }

deglossaryentry Like \newglossaryentry but does nothing if the entry has already been defined.
1919 \newrobustcmd{\provideglossaryentry}[2]{%
1920 \ifglstryexists{#1}%
1921 {}%
1922 {%
1923 \gls@defglossaryentry{#1}{#2}%
1924 }%
1925 }
1926 \@onlypreamble{\provideglossaryentry}

w@glossaryentry For use in document environment.
1927 \newrobustcmd{\new@glossaryentry}[2]{%
1928 \ifundef\@gls@deffile
1929 {%
1930 \global\newwrite\@gls@deffile
1931 \immediate\openout\@gls@deffile=\jobname.glsdefs
1932 }%
1933 {}%
1934 \ifglstryexists{#1}{}%
1935 {%
1936 \gls@defglossaryentry{#1}{#2}%
1937 }%

```

```

1938 \@gls@writedef{#1}%
1939 }
1940 \AtBeginDocument
1941 {
1942   \@gls@enablesavenonumberlist
1943   \makeatletter
1944   \InputIfFileExists{\jobname.glsdefs}{-}{-}%
1945   \makeatother
1946   \gls@defdocnewglossaryentry
1947 }
1948 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{-}}

```

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

```

1949 \newcommand*{\@gls@writedef}[1]{%
1950   \immediate\write\@gls@deffile
1951   {%
1952     \string\ifglsentryexists{#1}{-}\glspercentchar^^J%
1953     \expandafter\@gobble\string\{\glspercentchar^^J%
1954     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1955     \expandafter\@gobble\string\{\glspercentchar%
1956   }%

   Write key value information:
1957   \@for\@gls@map:=\@gls@keymap\do
1958   {%
1959     \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
1960     \ifdef\glo@value
1961     {%
1962       \@onelevel@sanitize\glo@value
1963       \immediate\write\@gls@deffile
1964       {%
1965         \expandafter\@firstoftwo\@gls@map
1966         =\expandafter\@gobble\string\{\glo@value\expandafter\@gobble\string\},%
1967         \glspercentchar
1968       }%
1969     }%
1970   }%
1971 }%

```

Provide hook:

```

1972 \gls.writedefhook
1973 \immediate\write\@gls@deffile
1974 {%
1975   \glspercentchar^^J%
1976   \expandafter\@gobble\string\}\glspercentchar^^J%
1977   \expandafter\@gobble\string\}\glspercentchar%
1978 }%
1979 }

```

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

```

1980 \newcommand*{\@gls@keymap}{%
1981   {name}{name},%
1982   {sort}{sortvalue},% unescaped sort value
1983   {type}{type},%
1984   {first}{first},%
1985   {firstplural}{firstpl},%
1986   {text}{text},%
1987   {plural}{plural},%
1988   {description}{desc},%
1989   {descriptionplural}{descplural},%
1990   {symbol}{symbol},%
1991   {symbolplural}{symbolplural},%
1992   {user1}{useri},%
1993   {user2}{userii},%
1994   {user3}{useriii},%
1995   {user4}{useriv},%
1996   {user5}{userv},%
1997   {user6}{uservi},%
1998   {long}{long},%
1999   {longplural}{longpl},%
2000   {short}{short},%
2001   {shortplural}{shortpl},%
2002   {counter}{counter},%
2003   {parent}{parent}%
2004 }
```

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

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

```

2005 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```

2006   \edef\@gls@thisval{#2}%
```

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

```

2007   \@for\@gls@map:=\@gls@keymap\do{%
2008     \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
2009     \ifdefequal{\@this@key}{\@gls@thisval}%
2010     {%
```

Found it.

```

2011       \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```

2012       \@endfortrue
2013     }%
2014   }%
```

2015 }%  
2016 }

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.

2017 `\newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}`

Starred version switches on expansion for this key.

2018 `\newcommand*{\@sglsaddstoragekey}[1]{%`  
2019 `\key@ifundefined{glossentry}{#1}%`  
2020 `{%`  
2021 `\expandafter\newcommand\expandafter*\expandafter`  
2022 `{\csname gls@assign@#1@field\endcsname}[2]{%`  
2023 `\@gls@expand@field{##1}{#1}{##2}%`  
2024 `}%`  
2025 `}%`  
2026 `{}%`  
2027 `\@glsaddstoragekey{#1}%`  
2028 `}`

Unstarred version doesn't override default expansion.

2029 `\newcommand*{\@glsaddstoragekey}[3]{%`

Check the specified key doesn't already exist.

2030 `\key@ifundefined{glossentry}{#1}%`  
2031 `{%`

Set up the key.

2032 `\define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%`  
2033 `\appto\@gls@keymap{, #1}{#1}}%`

Set the default value.

2034 `\appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%`

Assignment code.

2035 `\appto\@newglossaryentryposthook{%`  
2036 `\letcs{\@glo@tmp}{@glo@#1}%`  
2037 `\gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%`  
2038 `}%`

Define the no-link commands.

2039 `\newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%`  
2040 `}%`  
2041 `{%`  
2042 `\PackageError{glossaries}{Key '#1' already exists}{}%`  
2043 `}%`  
2044 `}`



```
\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.

```
2045 \newcommand*{\glsaddkey}{\@ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
2046 \newcommand*{\@sglsaddkey}[1]{%
2047   \key@ifundefined{glossentry}{#1}%
2048   {%
2049     \expandafter\newcommand\expandafter*\expandafter
2050     {\csname gls@assign@#1@field@endcsname}[2]{%
2051       \@gls@expand@field{##1}{#1}{##2}%
2052     }%
2053   }%
2054   {%
2055     \@glsaddkey{#1}%
2056 }
```

Unstarred version doesn't override default expansion.

```
2057 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2058   \key@ifundefined{glossentry}{#1}%
2059   {%
```

Set up the key.

```
2060   \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2061   \appto\@gls@keymap{,{#1}{#1}}%
```

Set the default value.

```
2062   \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2063   \appto\@newglossaryentryposthook{%
2064     \letcs{\@glo@tmp}{@glo@#1}%
2065     \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2066   }%
```

Define the no-link commands.

```
2067   \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2068   \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

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

```
2069   \ifcsdef{@gls@user@#1@}%
2070   {%
2071     \PackageError{glossaries}%
2072     {Can't define '\string#5' as helper command
2073     '\expandafter\string\csname @gls@user@#1@endcsname' already exists}%
2074     {%
2075   }%
2076   {%
```

```

2077 \expandafter\newcommand\expandafter*\expandafter
2078 {\csname @gls@user@#1\endcsname}[2][\%
2079 \new@ifnextchar[%
2080 {\csuse{@gls@user@#1@}{##1}{##2}}}%
2081 {\csuse{@gls@user@#1@}{##1}{##2}[]}}}%
2082 \csdef{@gls@user@#1@}##1##2[##3]{%
2083 \@gls@field@link{##1}{##2}{#3{##2}##3}%
2084 }%
2085 \newrobustcmd*{#5}{%
2086 \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2087 }%

```

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

```

2088 \ifcsdef{@Gls@user@#1@}%
2089 {%
2090 \PackageError{glossaries}%
2091 {Can't define '\string#6' as helper command
2092 '\expandafter\string\csname @Gls@user@#1@\endcsname' already exists}%
2093 }%
2094 }%
2095 {%
2096 \expandafter\newcommand\expandafter*\expandafter
2097 {\csname @Gls@user@#1\endcsname}[2][\%
2098 \new@ifnextchar[%
2099 {\csuse{@Gls@user@#1@}{##1}{##2}}}%
2100 {\csuse{@Gls@user@#1@}{##1}{##2}[]}}}%
2101 \csdef{@Gls@user@#1@}##1##2[##3]{%
2102 \@gls@field@link{##1}{##2}{#4{##2}##3}%
2103 }%
2104 \newrobustcmd*{#6}{%
2105 \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2106 }%

```

Finally the all caps version:

```

2107 \ifcsdef{@GLS@user@#1@}%
2108 {%
2109 \PackageError{glossaries}%
2110 {Can't define '\string#7' as helper command
2111 '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2112 }%
2113 }%
2114 {%
2115 \expandafter\newcommand\expandafter*\expandafter
2116 {\csname @GLS@user@#1\endcsname}[2][\%
2117 \new@ifnextchar[%
2118 {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2119 {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
2120 \csdef{@GLS@user@#1@}##1##2[##3]{%

```

```

2121      \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2122    }%
2123    \newrobustcmd*{#7}{%
2124      \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2125    }%
2126  }%
2127  {%
2128    \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2129  }%
2130 }

```

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

```

2131 \newcommand{\glsfieldxdef}[3]{%
2132   \glsdoifexists{#1}%
2133   {%
2134     \edef\@glo@label{\glsdetoklabel{#1}}}%
2135     \ifcsdef{glo@\@glo@label @#2}%
2136     {%
2137       \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2138     }%
2139     {%
2140       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2141     }%
2142   }%
2143 }

```

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

```

2144 \newcommand{\glsfielddedef}[3]{%
2145   \glsdoifexists{#1}%
2146   {%
2147     \edef\@glo@label{\glsdetoklabel{#1}}}%
2148     \ifcsdef{glo@\@glo@label @#2}%
2149     {%
2150       \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2151     }%
2152     {%
2153       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2154     }%
2155   }%
2156 }

```

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

```

2157 \newcommand{\glsfieldgdef}[3]{%
2158   \glsdoifexists{#1}%
2159   {%
2160     \edef\@glo@label{\glsdetoklabel{#1}}%
2161     \ifcsdef{glo@\@glo@label @#2}%
2162     {%
2163       \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2164     }%
2165     {%
2166       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2167     }%
2168   }%
2169 }

```

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

```

2170 \newcommand{\glsfielddef}[3]{%
2171   \glsdoifexists{#1}%
2172   {%
2173     \edef\@glo@label{\glsdetoklabel{#1}}%
2174     \ifcsdef{glo@\@glo@label @#2}%
2175     {%
2176       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2177     }%
2178     {%
2179       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2180     }%
2181   }%
2182 }

```

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

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

```

2183 \newcommand{\glsfieldfetch}[3]{%
2184   \glsdoifexists{#1}%
2185   {%
2186     \edef\@glo@label{\glsdetoklabel{#1}}%
2187     \ifcsdef{glo@\@glo@label @#2}%
2188     {%
2189       \letcs#3{glo@\@glo@label @#2}%
2190     }%
2191     {%
2192       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2193     }%

```

```
2194 }%
2195 }
```

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

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

```
2196 \newcommand{\ifglsfieldeq}[5]{%
2197   \glsdoifexists{#1}%
2198   {%
2199     \edef\glo@label{\glsdetoklabel{#1}}%
2200     \ifcsdef{glo@\glo@label @#2}%
2201     {%
2202       \ifcsstring{glo@\glo@label @#2}{#3}{#4}{#5}%
2203     }%
2204     {%
2205       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2206     }%
2207   }%
2208 }
```

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

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

```
2209 \newcommand{\ifglsfieldddefeq}[5]{%
2210   \glsdoifexists{#1}%
2211   {%
2212     \edef\glo@label{\glsdetoklabel{#1}}%
2213     \ifcsdef{glo@\glo@label @#2}%
2214     {%
2215       \expandafter\ifdefstrequal
2216       \csname glo@\glo@label @#2\endcsname{#3}{#4}{#5}%
2217     }%
2218     {%
2219       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2220     }%
2221   }%
2222 }
```

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

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

```
2223 \newcommand{\ifglsfielddcseq}[5]{%
2224   \glsdoifexists{#1}%
2225   {%
2226     \edef\glo@label{\glsdetoklabel{#1}}%
```

```

2227 \ifcsdef{glo@\@glo@label @#2}%
2228 {%
2229 \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2230 }%
2231 {%
2232 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2233 }%
2234 }%
2235 }

```

gls@writedefhook

```

2236 \newcommand*{\glswritedefhook}{}

```

gls@assign@desc

```

2237 \newcommand*{\glsgl@assign@desc}[1]{%
2238 \glsgl@assign@field{#1}{desc}{\@glo@desc}%
2239 \glsgl@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2240 }

```

ewglossaryentry

```

2241 \newcommand{\longnewglossaryentry}[3]{%
2242 \glsgl@doifnoexists{#1}%
2243 {%
2244 \bgroup
2245 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2246 \long\def\@newglossaryentryprehook{%
2247 \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2248 \@org@newglossaryentryprehook
2249 }%
2250 \renewcommand*{\glsgl@assign@desc}[1]{%
2251 \global\cslet{glo@\glsgl@detoklabel{#1}@desc}{\@glo@desc}%
2252 \global\cslet{glo@\glsgl@detoklabel{#1}@descplural}{\@glo@desc}%
2253 }
2254 \glsgl@defglossaryentry{#1}{#2}%
2255 \egroup
2256 }
2257 }

```

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

```

2258 \@onlypreamble{\longnewglossaryentry}

```

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

```

2259 \newcommand{\longprovideglossaryentry}[3]{%
2260 \ifglsgl@entryexists{#1}{}%
2261 {\longnewglossaryentry{#1}{#2}{#3}}%
2262 }
2263 \@onlypreamble{\longprovideglossaryentry}

```

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

Defines a new entry without checking if it already exists.

```
2264 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of \GlsSetQuote:

```
2265 \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2266 \edef\@glo@label{\glsdetoklabel{#1}}%
```

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

```
2267 \let\glslabel\@glo@label
```

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

```
2268 \let\@glo@name\@gls@name
```

```
2269 \let\@glo@desc\@gls@desc
```

```
2270 \let\@glo@descplural\@gls@default@value
```

```
2271 \let\@glo@type\@gls@default@value
```

```
2272 \let\@glo@symbol\@gls@default@value
```

```
2273 \let\@glo@symbolplural\@gls@default@value
```

```
2274 \let\@glo@text\@gls@default@value
```

```
2275 \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.)

```
2276 \let\@glo@first\@gls@default@value
```

```
2277 \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2278 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2279 \let\@glo@counter\@gls@default@value
```

```
2280 \def\@glo@see{}%
```

```
2281 \def\@glo@parent{}%
```

```
2282 \def\@glo@prefix{}%
```

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

```
2283 \@gls@initnonnumberlist
```

```
2284 \def\@glo@useri{}%
```

```
2285 \def\@glo@userii{}%
```

```
2286 \def\@glo@useriii{}%
```

```
2287 \def\@glo@useriv{}%
```

```
2288 \def\@glo@userv{}%
```

```
2289 \def\@glo@uservi{}%
```

```

2290 \def\@glo@short{}%
2291 \def\@glo@shortpl{}%
2292 \def\@glo@long{}%
2293 \def\@glo@longpl{}%

```

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

```
2294 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2295 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```

2296 \ifundef\glsdefaulttype
2297 {%
2298   \PackageError{glossaries}%
2299     {No default glossary type (have you used ‘nomain’ by mistake?)}%
2300     {If you use package option ‘nomain’ you must define
2301      a new glossary before you can define entries}%
2302 }%
2303 {}%

```

Assign type. This must be fully expandable

```

2304 \gls@assign@field{\glsdefaulttype}{\@glo@label}{type}{\@glo@type}%
2305 \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.

```

2306 \ifcsundef{glolist@\@glo@type}%
2307 {%
2308   \PackageError{glossaries}%
2309     {Glossary type ‘\@glo@type’ has not been defined}%
2310     {You need to define a new glossary type, before making entries
2311      in it}%
2312 }%
2313 {%

```

Check if it's an ignored glossary

```

2314 \ifignoredglossary\@glo@type
2315 {%

```

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

```

2316   \ifx\@glo@desc\@glsnodel
2317     \let\@glo@desc\@empty
2318   \fi
2319 }%
2320 {%
2321 }%
2322 \protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
2323 \expandafter\xdef\csname glolist@\@glo@type\endcsname{%
2324   \@glolist{\@glo@label},}%
2325 }%

```



Initialise level to 0.

```

2326 \gls@level=0\relax

```

Has this entry been assigned a parent?

```

2327 \ifx\@glo@parent\@empty

```

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

```

2328 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2329 \else

```

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

```

2330 \ifdefequal\@glo@label\@glo@parent%
2331 {%
2332 \PackageError{glossaries}{Entry '@glo@label' can't be its own parent}{}%
2333 \def\@glo@parent{}%
2334 \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2335 }%
2336 {%

```

Check the parent exists:

```

2337 \ifglentryexists{\@glo@parent}%
2338 {%

```

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

```

2339 \expandafter\xdef\csname glo@\@glo@label @parent\endcsname{%
2340 \@glo@parent}%

```

Determine level.

```

2341 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2342 \advance\gls@level by 1\relax

```

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

```

2343 \ifx\@glo@name\@gls@name
2344 \expandafter\let\expandafter\@glo@name
2345 \csname glo@\@glo@parent @name\endcsname

```

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

```

2346 \ifx\@glo@plural\@gls@default@value
2347 \expandafter\let\expandafter\@glo@plural
2348 \csname glo@\@glo@parent @plural\endcsname
2349 \fi
2350 \fi
2351 }%
2352 {%

```

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

```

2353 \PackageError{glossaries}%
2354 {%
2355 Invalid parent '@glo@parent'
2356 for entry '@glo@label' - parent doesn't exist%
2357 }%
2358 {%
2359 Parent entries must be defined before their children%

```

```

2360     }%
2361     \def\@glo@parent{}%
2362     \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2363     }%
2364     }%
2365 \fi

Set the level for this entry
2366 \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%

Define commands associated with this entry:
2367 \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2368 \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2369 \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2370 \expandafter\gls@assign@field\expandafter
2371     {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2372     {\@glo@label}{plural}{\@glo@plural}%
2373 \expandafter\gls@assign@field\expandafter
2374     {\csname glo@\@glo@label @text\endcsname}%
2375     {\@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.
2376 \ifx\@glo@first\@gls@default@value
2377     \expandafter\gls@assign@field\expandafter
2378         {\csname glo@\@glo@label @plural\endcsname}%
2379         {\@glo@label}{firstpl}{\@glo@firstplural}%
2380 \else
2381     \expandafter\gls@assign@field\expandafter
2382         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2383         {\@glo@label}{firstpl}{\@glo@firstplural}%
2384 \fi

2385 \ifcsundef{@glotype@\@glo@type @counter}%
2386 {%
2387     \def\@glo@defaultcounter{\glscounter}%
2388 }%
2389 {%
2390     \letcs\@glo@defaultcounter{@glotype@\@glo@type @counter}%
2391 }%
2392 \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2393 \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2394 \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2395 \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2396 \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2397 \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2398 \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2399 \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2400 \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2401 \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2402 \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%

```

```

2403 \ifx\@glo@name\@glsnoname
2404   \@glsnoname
2405   \let\@glo@name\@gls@default@value
2406 \fi
2407 \gls@assign@field{}\@glo@label{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2408 \ifcsundef{glo@\@glo@label @numberlist}%
2409 {%
2410   \csxdef{glo@\@glo@label @numberlist}{%
2411     \noexpand\@gls@missingnumberlist{\@glo@label}}%
2412   }%
2413   {}%

```

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

```

2414 \@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.

```

2415 \def\@glo@@desc{\@glo@first}%
2416 \ifx\@glo@desc\@glo@@desc
2417   \let\@glo@desc\@glo@first
2418 \fi
2419 \ifx\@glo@desc\@gls@nodesc
2420   \@gls@nodesc
2421   \let\@glo@desc\@gls@default@value
2422 \fi
2423 \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```

2424 \@gls@defsort{\@glo@type}{\@glo@label}%

2425 \def\@glo@@symbol{\@glo@text}%
2426 \ifx\@glo@symbol\@glo@@symbol
2427   \let\@glo@symbol\@glo@text
2428 \fi
2429 \gls@assign@field{relax}{\@glo@label}{symbol}{\@glo@symbol}%
2430 \expandafter
2431   \gls@assign@field\expandafter
2432   {\csname glo@\@glo@label @symbol\endcsname}
2433   {\@glo@label}{symbolplural}{\@glo@symbolplural}%

```

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

```

2434 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2435   \noexpand\global
2436   \noexpand\let\expandafter\noexpand
2437   \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2438   }%
2439 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2440   \noexpand\global

```

```

2441      \noexpand\let\expandafter\noexpand
2442      \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2443  }%
2444  \csname glo@\@glo@label @flagfalse\endcsname

```

Sort out any cross-referencing if required.

```
2445  \@glo@autosee
```

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

```

2446  \ifignoredglossary\@glo@type
2447  {%
2448    \csdef{glo@\@glo@label @index}{}%
2449  }
2450  {%
2451    \do@glo@storeentry{\@glo@label}%
2452  }%

```

Define entry counters if enabled:

```
2453  \@newglossaryentry@defcounters
```

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

```

2454  \@newglossaryentryposthook
2455 }

```

\@glo@autosee Automatically implement \glssee.

```

2456 \newcommand*{\@glo@autosee}{%
2457   \ifdefvoid\@glo@see{}%
2458   {%
2459     \protected@edef\@do@glssee{%
2460       \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see\noexpand\@nil
2461       \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{\@glo@label}}%
2462     \@do@glssee
2463   }%
2464   \@glo@autoseehook
2465 }%

```

glo@autoseehook

```
2466 \newcommand*{\@glo@autoseehook}{}
```

aryentryprehook Allow extra information to be added to glossary entries:

```
2467 \newcommand*{\@newglossaryentryprehook}{}
```

ryentryposthook Allow extra information to be added to glossary entries:

```
2468 \newcommand*{\@newglossaryentryposthook}{}
```

try@defcounters

```
2469 \newcommand*{\@newglossaryentry@defcounters}{}
```

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

```

2470 \newcommand*{\glsmoveentry}[2]{%
2471   \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2472   \edef\glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2473   \def\glo@list{,%}
2474   \forglsentries[\glo@type]{\glo@label}%
2475   {%

2476     \ifdefequal\@glo@thislabel\glo@label
2477       {\eappto\glo@list{\glo@label,%}}%
2478     }%
2479     \cslet\glo@list@\glo@type{\glo@list}%
2480     \csdef{glo@\@glo@thislabel @type}{#2}%
2481 }

```

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

```

2482 \ifglxindy
2483   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2484 \else
2485   \newcommand*{\@glossaryentryfield}{\string\glossentry}
2486 \fi

```

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

```

2487 \ifglxindy
2488   \newcommand*{\@glossarysubentryfield}{%
2489     \string\subglossentry}
2490 \else
2491   \newcommand*{\@glossarysubentryfield}{%
2492     \string\subglossentry}
2493 \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.)

```

2494 \newcommand{\@glo@storeentry}[1]{%

```

Escape makeindex/xindy special characters in the label:

```

2495   \edef\@glo@esclabel{#1}%
2496   \@gls@checkmkidxchars\@glo@esclabel

```

Get the sort string and escape any special characters

```

2497 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
2498 \@gls@checkmkidxchars\@glo@sort
    Same again for the name string. Escape any special characters in the prefix
2499 \@gls@checkmkidxchars\@glo@prefix
    Get the parent, if one exists
2500 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
    Write the information to the glossary file.
2501 \ifglxsindy
    Store using xindy syntax.
2502 \ifx\@glo@parent\@empty
    Entry doesn't have a parent
2503 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2504 (\string"\@glo@sort\string" %
2505 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
2506 }%
2507 \else
    Entry has a parent
2508 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2509 \csname glo@\@glo@parent @index\endcsname
2510 (\string"\@glo@sort\string" %
2511 \string"\@glo@prefix\@glossarysubentryfield
2512 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
2513 }%
2514 \fi
2515 \else
    Store using makeindex syntax.
2516 \ifx\@glo@parent\@empty
    Sanitize \@glo@prefix
2517 \@onelevel@sanitize\@glo@prefix
    Entry doesn't have a parent
2518 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2519 \@glo@sort\@gls@actualchar\@glo@prefix
2520 \@glossaryentryfield{\@glo@esclabel}%
2521 }%
2522 \else
    Entry has a parent
2523 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
2524 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2525 \@glo@sort\@gls@actualchar\@glo@prefix
2526 \@glossarysubentryfield
2527 {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2528 }%
2529 \fi
2530 \fi
2531 }

```

## 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`

```
2532 \AtBeginDocument{%
2533   \ifpackageloaded{amsmath}%
2534   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2535   }%
2536 }
2537 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2538   \ifmeasuring@
2539   \else
2540     #1%
2541   \fi
2542 }
2543 \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.

```
2544 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2545   \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2546 }
2547 \newcommand*\glspatchtabularx{%
2548   \ifdef\TX@trial
2549   {%
2550     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2551     \let\glspatchtabularx\relax
2552   }%
2553   }%
2554 }
```

`\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.

```
2555 \newcommand*{\glsreset}[1]{%
2556   \gls@ifnotmeasuring
2557   {%
2558     \glsdoifexists{#1}%
2559     {%
2560       \@glsreset{#1}%
2561     }%
2562   }%
2563 }
```

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

```

2564 \newcommand*{\glslocalreset}[1]{%
2565   \gls@ifnotmeasuring
2566   {%
2567     \glsdoifexists{#1}%
2568     {%
2569       \@glslocalreset{#1}%
2570     }%
2571   }%
2572 }

```

`\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.

```

2573 \newcommand*{\glsunset}[1]{%
2574   \gls@ifnotmeasuring
2575   {%
2576     \glsdoifexists{#1}%
2577     {%
2578       \@glsunset{#1}%
2579     }%
2580   }%
2581 }

```

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

```

2582 \newcommand*{\glslocalunset}[1]{%
2583   \gls@ifnotmeasuring
2584   {%
2585     \glsdoifexists{#1}%
2586     {%
2587       \@glslocalunset{#1}%
2588     }%
2589   }%
2590 }

```

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

```

2591 \newcommand*{\@glslocalunset}{\@glslocalunset}

```

`@@glslocalunset` Local unset without checks.

```

2592 \newcommand*{\@glslocalunset}[1]{%
2593   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2594 }

```

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

```

2595 \newcommand*{\@glsunset}{\@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2596 \newcommand*{\@@glsunset}[1]{%
2597   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2598 }

```



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

```
2599 \newcommand*{\@glslocalreset}{\@@glslocalreset}
```

`\@@glslocalreset` Local reset without checks.

```
2600 \newcommand*{\@@glslocalreset}[1]{%
2601   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2602 }
```

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

```
2603 \newcommand*{\@glsreset}{\@@glsreset}
```

`\@@glsreset` Global reset without checks.

```
2604 \newcommand*{\@@glsreset}[1]{%
2605   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2606 }
```

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

`\glsresetall`

```
2607 \newcommand*{\glsresetall}[1][\@glo@types]{%
2608   \forallglsentries[#1]{\@glsentry}%
2609   {%
2610     \glsreset{\@glsentry}%
2611   }%
2612 }
```

As above, but with only a local effect:

`\glslocalresetall`

```
2613 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2614   \forallglsentries[#1]{\@glsentry}%
2615   {%
2616     \glslocalreset{\@glsentry}%
2617   }%
2618 }
```

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

`\glsunsetall`

```
2619 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2620   \forallglsentries[#1]{\@glsentry}%
2621   {%
2622     \glsunset{\@glsentry}%
2623   }%
2624 }
```

As above, but with only a local effect:

lslocalunsetall

```

2625 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2626   \forallglsentries[#1]{\@glsentry}%
2627   {%
2628     \glslocalunset{\@glsentry}%
2629   }%
2630 }

```

## 1.9 Keeping Track of How Many Times an Entry Has Been Unset

Version 4.14 introduced `\glsenableentrycount` that keeps track of how many times an entry is marked as used. The counter is reset back to zero when the first use flag is reset. Note that although the word “counter” is used here, it’s not an actual  $\TeX$  counter or even an explicit  $\TeX$  count register but is just a macro. Any of the commands that use `\glsunset` or `\glslocalunset`, such as `\gls`, will automatically increment this value. Commands that don’t modify the first use flag (such as `\glstext` or `\glsentrytext`) don’t modify this value.

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

```

2631 \newcommand*{\@newglossaryentry@defcounters}{%
2632   \csdef{glo@\@glo@label @currcount}{0}%
2633   \csdef{glo@\@glo@label @prevcount}{0}%
2634 }

```

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

```

2635 \newcommand*{\glsenableentrycount}{%

```

Enable new entry fields.

```

2636   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters

```

Disable `\newglossaryentry` in the document environment.

```

2637   \renewcommand*{\gls@defdocnewglossaryentry}{%
2638     \renewcommand*{\newglossaryentry}[2]{%
2639       \PackageError{glossaries}{\string\newglossaryentry\space
2640       may only be used in the preamble when entry counting has
2641       been activated}{If you use \string\glsenableentrycount\space
2642       you must place all entry definitions in the preamble not in
2643       the document environment}%
2644     }%
2645   }%

```

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

```

2646   \newcommand*{\glsentrycurrcount}[1]{%
2647     \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2648     {0}{\@gls@entry@field{##1}{currcount}}%
2649   }%
2650   \newcommand*{\glsentryprevcount}[1]{%

```

```

2651 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2652 {0}{\@gls@entry@field{##1}{prevcount}}}%
2653 }%

```

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

```

2654 \renewcommand*{\@glsunset}[1]{%
2655   \@glsunset{##1}%
2656   \@gls@increment@currcount{##1}%
2657 }%
2658 \renewcommand*{\@glslocalunset}[1]{%
2659   \@glslocalunset{##1}%
2660   \@gls@local@increment@currcount{##1}%
2661 }%
2662 \renewcommand*{\@glsreset}[1]{%
2663   \@glsreset{##1}%
2664   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2665 }%
2666 \renewcommand*{\@glslocalreset}[1]{%
2667   \@glslocalreset{##1}%
2668   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2669 }%

```

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.)

```

2670 \def\@cgl's@##1##2[##3]{%
2671   \ifnum\glsentryprevcount{##2}=1\relax
2672     \cgl'sformat{##2}{##3}%
2673     \glsunset{##2}%
2674   \else
2675     \@gls@{##1}{##2}[##3]%
2676   \fi
2677 }%

```

Similarly for the analogous commands. No case change plural:

```

2678 \def\@cgl'spl@##1##2[##3]{%
2679   \ifnum\glsentryprevcount{##2}=1\relax
2680     \cgl'splformat{##2}{##3}%
2681     \glsunset{##2}%
2682   \else
2683     \@cgl'spl@{##1}{##2}[##3]%
2684   \fi
2685 }%

```

First letter uppercase singular:

```

2686 \def\@cGls@##1##2[##3]{%
2687   \ifnum\glsentryprevcount{##2}=1\relax
2688     \cGlsformat{##2}{##3}%
2689     \glsunset{##2}%
2690   \else
2691     \@cGls@{##1}{##2}[##3]%
2692   \fi

```

2693 }%

First letter uppercase plural:

```
2694 \def\@cGlsp1@##1##2[##3]{%
2695   \ifnum\glentryprevcount{##2}=1\relax
2696     \cGlsp1format{##2}{##3}%
2697     \glunset{##2}%
2698   \else
2699     \@Glsp1@{##1}{##2}[##3]%
2700   \fi
2701 }%
```

Write information to aux file at the end of the document

```
2702 \AtEndDocument{\@gls@write@entrycounts}%
```

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

```
2703 \renewcommand*{\@gls@entry@count}[2]{%
2704   \csxdef{glo@\glsetoklabel{##1}@prevcount}{##2}%
2705 }%
```

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

```
2706 \let\glsenableentrycount\relax
2707 }
2708 \@onlypreamble\glsenableentrycount
```

ement@currcount

```
2709 \newcommand*{\@gls@increment@currcount}[1]{%
2710   \csxdef{glo@\glsetoklabel{##1}@currcount}{%
2711     \number\numexpr\glentrycurrcount{##1}+1}%
2712 }
```

ement@currcount

```
2713 \newcommand*{\@gls@local@increment@currcount}[1]{%
2714   \csxdef{glo@\glsetoklabel{##1}@currcount}{%
2715     \number\numexpr\glentrycurrcount{##1}+1}%
2716 }
```

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.)

```
2717 \newcommand*{\@gls@write@entrycounts}{%
2718   \immediate\write\@auxout
2719     {\string\providecommand*\string\@gls@entry@count}[2]{}}%
2720 \forallglsentries{\@glentry}{%
2721   \ifglused{\@glentry}%
2722     {\immediate\write\@auxout
2723       {\string\@gls@entry@count{\@glentry}{\glentrycurrcount{\@glentry}}}}%
2724   {}}%
```

```
2725 }%
2726 }
```

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

```
2727 \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.)

```
2728 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}
```

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

```
2729 \newcommand*{\@cgl}[2][{}]{%
2730   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[]}%
2731 }
```

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

```
2732 \def\@cgl@#1#2[#3]{%
2733   \GlossariesWarning{\string\cgl\space is defaulting to
2734     \string\gls\space since you haven't enabled entry counting}%
2735   \@gls@{#1}{#2}[#3]%
2736 }
```

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

```
2737 \newcommand*{\cglformat}[2]{%
2738   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2739 }
```

`\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.)

```
2740 \newrobustcmd*{\cGl}{\@gls@hyp@opt\@cGl}
```

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

```
2741 \newcommand*{\@cGl}[2][{}]{%
2742   \new@ifnextchar[{\@cGl@{#1}{#2}}{\@cGl@{#1}{#2}[]}%
2743 }
```

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

```
2744 \def\@cGl@#1#2[#3]{%
2745   \GlossariesWarning{\string\cGl\space is defaulting to
2746     \string\Gls\space since you haven't enabled entry counting}%
2747   \@Gls@{#1}{#2}[#3]%
2748 }
```

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

```
2749 \newcommand*{\cGlsformat}[2]{%
2750   \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2751 }
```

`\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.)

```
2752 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}
```

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

```
2753 \newcommand*{\@cglsp1}[2][{}]{%
2754   \new@ifnextchar[\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2}[{}]}%
2755 }
```

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

```
2756 \def\@cglsp1@#1#2[#3]{%
2757   \GlossariesWarning{\string\cglsp1\space is defaulting to
2758     \string\glsp1\space since you haven't enabled entry counting}%
2759   \@glsp1@{#1}{#2}[#3]%
2760 }
```

`\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.

```
2761 \newcommand*{\cglsp1format}[2]{%
2762   \ifglshaslong{#1}{\glsp1long{#1}}{\glsp1firstplural{#1}}#2%
2763 }
```

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

```
2764 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}
```

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

```
2765 \newcommand*{\@cGlspl}[2][{}]{%
2766   \new@ifnextchar[\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2}[{}]}%
2767 }
```

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

```
2768 \def\@cGlspl@#1#2[#3]{%
2769   \GlossariesWarning{\string\cGlspl\space is defaulting to
2770     \string\Glspl\space since you haven't enabled entry counting}%
2771   \@Glspl@{#1}{#2}[#3]%
2772 }
```

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

```
2773 \newcommand*{\cGlsplformat}[2]{%
2774   \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2775 }
```

## 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>

`\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`

```
2776 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2777   \let\@gls@default\glsdefaulttype
2778   \def\glsdefaulttype{#1}\input{#2}%
2779   \let\glsdefaulttype\@gls@default
2780 }
```

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

```
2781 \@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`

```
2782 \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

---

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

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

```
2783 \newcommand*{\glsentryfmt}{%
2784   \@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2785 }
```

Format that provides backwards compatibility:

```
2786 \newcommand*{\@gls@default@entryfmt}[2]{%
2787   \ifdefempty\glscustomtext
2788   {%
2789     \glsifplural
2790     {%
```

Plural form

```
2791     \glscapscase
2792     {%
```

Don't adjust case

```
2793     \ifglsused\glslabel
2794     {%
```

Subsequent use

```
2795         #2{\glsentryplural{\glslabel}}%
2796         {\glsentrydescplural{\glslabel}}%
2797         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2798     }%
2799     {%
```

First use

```
2800         #1{\glsentryfirstplural{\glslabel}}%
2801         {\glsentrydescplural{\glslabel}}%
2802         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2803     }%
2804     }%
2805     {%
```

Make first letter upper case

```
2806     \ifglsused\glslabel
2807     {%
```

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.)

```
2808     \ifbool{glscompatible-3.07}%
2809     {%
2810       \protected@edef\@glo@etext{%
2811         #2{\glsentryplural{\glslabel}}%
2812         {\glsentrydescplural{\glslabel}}%
2813         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2814       \xmakefirstuc\@glo@etext
2815     }%
```



```

2816      {%
2817      #2{\Glsentryplural{\glslabel}}%
2818      {\glsentrydescplural{\glslabel}}%
2819      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2820      }%
2821      }%
2822      {%

```

First use

```

2823      \ifbool{glscompatible-3.07}%
2824      {%
2825      \protected@edef\@glo@etext{%
2826      #1{\Glsentryfirstplural{\glslabel}}%
2827      {\glsentrydescplural{\glslabel}}%
2828      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2829      \xmakefirstuc\@glo@etext
2830      }%
2831      {%
2832      #1{\Glsentryfirstplural{\glslabel}}%
2833      {\glsentrydescplural{\glslabel}}%
2834      {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2835      }%
2836      }%
2837      }%
2838      {%

```

#### Make all upper case

```

2839      \ifglsused\glslabel
2840      {%

```

#### Subsequent use

```

2841      \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2842      {\glsentrydescplural{\glslabel}}%
2843      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2844      }%
2845      {%

```

#### First use

```

2846      \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2847      {\glsentrydescplural{\glslabel}}%
2848      {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2849      }%
2850      }%
2851      }%
2852      {%

```

#### Singular form

```

2853      \glscapscase
2854      {%

```

#### Don't adjust case

```

2855      \ifglsused\glslabel

```

```

2856      {%
Subsequent use
2857      #2{\glsentrytext{\glslabel}}%
2858      {\glsentrydesc{\glslabel}}%
2859      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2860      }%
2861      {%

First use
2862      #1{\glsentryfirst{\glslabel}}%
2863      {\glsentrydesc{\glslabel}}%
2864      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2865      }%
2866      }%
2867      {%

Make first letter upper case
2868      \ifglused\glslabel
2869      {%

Subsequent use
2870      \ifbool{glscompatible-3.07}%
2871      {%
2872      \protected@edef\@glo@etext{%
2873      #2{\glsentrytext{\glslabel}}%
2874      {\glsentrydesc{\glslabel}}%
2875      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2876      \xmakefirstuc\@glo@etext
2877      }%
2878      {%
2879      #2{\Glsentrytext{\glslabel}}%
2880      {\glsentrydesc{\glslabel}}%
2881      {\glsentrysymbol{\glslabel}}{\glsinsert}%
2882      }%
2883      }%
2884      {%

First use
2885      \ifbool{glscompatible-3.07}%
2886      {%
2887      \protected@edef\@glo@etext{%
2888      #1{\glsentryfirst{\glslabel}}%
2889      {\glsentrydesc{\glslabel}}%
2890      {\glsentrysymbol{\glslabel}}{\glsinsert}}%
2891      \xmakefirstuc\@glo@etext
2892      }%
2893      {%
2894      #1{\Glsentryfirst{\glslabel}}%
2895      {\glsentrydesc{\glslabel}}%
2896      {\glsentrysymbol{\glslabel}}{\glsinsert}%

```

```

2897         }%
2898     }%
2899 }%
2900 {%

```

Make all upper case

```

2901     \ifglsused\glslabel
2902     {%

```

Subsequent use

```

2903         \mfirstucMakeUppercase{#2{\glsentrytext{\glslabel}}}%
2904         {\glsentrydesc{\glslabel}}}%
2905         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2906     }%
2907     {%

```

First use

```

2908         \mfirstucMakeUppercase{#1{\glsentryfirst{\glslabel}}}%
2909         {\glsentrydesc{\glslabel}}}%
2910         {\glsentrysymbol{\glslabel}}{\glsinsert}}}%
2911     }%
2912 }%
2913 }%
2914 }%
2915 {%

```

Custom text provided in \glsdisp

```

2916     \ifglsused{\glslabel}%
2917     {%

```

Subsequent use

```

2918         #2{\glscustomtext}%
2919         {\glsentrydesc{\glslabel}}}%
2920         {\glsentrysymbol{\glslabel}}{}%
2921     }%
2922     {%

```

First use

```

2923         #1{\glscustomtext}%
2924         {\glsentrydesc{\glslabel}}}%
2925         {\glsentrysymbol{\glslabel}}{}%
2926     }%
2927 }%
2928 }

```

`\glsenentryfmt` Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

```

2929 \newcommand*{\glsenentryfmt}{%
2930     \ifdefempty\glscustomtext
2931     {%
2932         \glsifplural
2933         {%

```

### Plural form

2934        \glscapscase  
2935        {%

### Don't adjust case

2936        \ifglused\glslabel  
2937        {%

### Subsequent use

2938        \glentryplural{\glslabel}\glinsert  
2939        }%  
2940        {%

### First use

2941        \glentryfirstplural{\glslabel}\glinsert  
2942        }%  
2943        }%  
2944        {%

### Make first letter upper case

2945        \ifglused\glslabel  
2946        {%

### Subsequent use.

2947        \Glsentryplural{\glslabel}\glinsert  
2948        }%  
2949        {%

### First use

2950        \Glsentryfirstplural{\glslabel}\glinsert  
2951        }%  
2952        }%  
2953        {%

### Make all upper case

2954        \ifglused\glslabel  
2955        {%

### Subsequent use

2956        \mfirstucMakeUppercase  
2957        {\glentryplural{\glslabel}\glinsert}%  
2958        }%  
2959        {%

### First use

2960        \mfirstucMakeUppercase  
2961        {\glentryfirstplural{\glslabel}\glinsert}%  
2962        }%  
2963        }%  
2964        }%  
2965        {%

Singular form

```
2966      \glscapscase
2967      {%
```

Don't adjust case

```
2968      \ifglused\glslabel
2969      {%
```

Subsequent use

```
2970      \glentrytext{\glslabel}\glsinsert
2971      }%
2972      {%
```

First use

```
2973      \glentryfirst{\glslabel}\glsinsert
2974      }%
2975      }%
2976      {%
```

Make first letter upper case

```
2977      \ifglused\glslabel
2978      {%
```

Subsequent use

```
2979      \Glentrytext{\glslabel}\glsinsert
2980      }%
2981      {%
```

First use

```
2982      \Glentryfirst{\glslabel}\glsinsert
2983      }%
2984      }%
2985      {%
```

Make all upper case

```
2986      \ifglused\glslabel
2987      {%
```

Subsequent use

```
2988      \mfirstucMakeUppercase{\glentrytext{\glslabel}\glsinsert}%
2989      }%
2990      {%
```

First use

```
2991      \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glsinsert}%
2992      }%
2993      }%
2994      }%
2995      }%
2996      {%
```

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

```
2997      \glscustomtext\glsinsert
```

```

2998 }%
2999 }

```

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

```

3000 \newcommand*{\glsgenacfmt}{%
3001   \ifdefempty\glscustomtext
3002   {%
3003     \ifglused\glslabel
3004     {%

```

Subsequent use:

```

3005     \glsifplural
3006     {%

```

Subsequent plural form:

```

3007     \glscapscase
3008     {%

```

Subsequent plural form, don't adjust case:

```

3009     \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert
3010     }%
3011     {%

```

Subsequent plural form, make first letter upper case:

```

3012     \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert
3013     }%
3014     {%

```

Subsequent plural form, all caps:

```

3015     \mfirstucMakeUppercase
3016     {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%
3017     }%
3018     }%
3019     {%

```

Subsequent singular form

```

3020     \glscapscase
3021     {%

```

Subsequent singular form, don't adjust case:

```

3022     \acronymfont{\glsentryshort{\glslabel}}\glsinsert
3023     }%
3024     {%

```

Subsequent singular form, make first letter upper case:

```

3025     \acronymfont{\Glsentryshort{\glslabel}}\glsinsert
3026     }%
3027     {%

```

Subsequent singular form, all caps:

```

3028     \mfirstucMakeUppercase
3029     {\acronymfont{\glsentryshort{\glslabel}}\glsinsert}%

```

```

3030      }%
3031      }%
3032      }%
3033      {%

```

First use:

```

3034      \glsifplural
3035      {%

```

First use plural form:

```

3036      \glscapscase
3037      {%

```

First use plural form, don't adjust case:

```

3038      \genplacrformat{\glslabel}{\glsinsert}%
3039      }%
3040      {%

```

First use plural form, make first letter upper case:

```

3041      \Genplacrformat{\glslabel}{\glsinsert}%
3042      }%
3043      {%

```

First use plural form, all caps:

```

3044      \mfirstucMakeUppercase
3045      {\genplacrformat{\glslabel}{\glsinsert}}%
3046      }%
3047      }%
3048      {%

```

First use singular form

```

3049      \glscapscase
3050      {%

```

First use singular form, don't adjust case:

```

3051      \genacrformat{\glslabel}{\glsinsert}%
3052      }%
3053      {%

```

First use singular form, make first letter upper case:

```

3054      \Genacrformat{\glslabel}{\glsinsert}%
3055      }%
3056      {%

```

First use singular form, all caps:

```

3057      \mfirstucMakeUppercase
3058      {\genacrformat{\glslabel}{\glsinsert}}%
3059      }%
3060      }%
3061      }%
3062      }%
3063      {%

```

User supplied text.

```
3064 \glscustomtext
3065 }%
3066 }
```

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

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

```
3067 \newcommand*{\genacrfullformat}[2]{%
3068   \glentrylong{#1}#2\space
3069   (\protect\firstacronymfont{\glentryshort{#1}})%
3070 }
```

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

As above but makes the first letter upper case.

```
3071 \newcommand*{\Genacrfullformat}[2]{%
3072   \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%
3073   \xmakefirstuc\gls@text
3074 }
```

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

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

```
3075 \newcommand*{\genplacrfullformat}[2]{%
3076   \glentrylongpl{#1}#2\space
3077   (\protect\firstacronymfont{\glentryshortpl{#1}})%
3078 }
```

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

As above but makes the first letter upper case.

```
3079 \newcommand*{\Genplacrfullformat}[2]{%
3080   \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%
3081   \xmakefirstuc\gls@text
3082 }
```

glsdisplayfirst Deprecated. Kept for backward compatibility.

```
3083 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3084 \newcommand*{\glsdisplay}[4]{#1#4}
```



`\defglsdisplay`   Deprecated. Kept for backward compatibility.

```
3085 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3086   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3087   Use \string\defglsentryfmt\space instead}%
3088   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3089   \edef\@gls@doentrydef{%
3090     \noexpand\defglsentryfmt[#1]{%
3091       \noexpand\ifcsdef{gls@#1@displayfirst}%
3092       {%
3093         \noexpand\@@gls@default@entryfmt
3094         {\noexpand\csuse{gls@#1@displayfirst}}}%
3095         {\noexpand\csuse{gls@#1@display}}}%
3096       }%
3097       {%
3098         \noexpand\@@gls@default@entryfmt
3099         {\noexpand\glsdisplayfirst}%
3100         {\noexpand\csuse{gls@#1@display}}}%
3101       }%
3102     }%
3103   }%
3104   \@gls@doentrydef
3105 }
```

`glsdisplayfirst`   Deprecated. Kept for backward compatibility.

```
3106 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3107   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3108   Use \string\defglsentryfmt\space instead}%
3109   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3110   \edef\@gls@doentrydef{%
3111     \noexpand\defglsentryfmt[#1]{%
3112       \noexpand\ifcsdef{gls@#1@display}%
3113       {%
3114         \noexpand\@@gls@default@entryfmt
3115         {\noexpand\csuse{gls@#1@displayfirst}}}%
3116         {\noexpand\csuse{gls@#1@display}}}%
3117       }%
3118       {%
3119         \noexpand\@@gls@default@entryfmt
3120         {\noexpand\csuse{gls@#1@displayfirst}}}%
3121         {\noexpand\glsdisplay}%
3122       }%
3123     }%
3124   }%
3125   \@gls@doentrydef
3126 }
```

## 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.

```
3127 \define@key{glslink}{counter}{%
3128   \ifcsundef{c@#1}%
3129   {%
3130     \PackageError{glossaries}%
3131     {There is no counter called ‘#1’}%
3132     {%
3133       The counter key should have the name of a valid counter
3134       as its value%
3135     }%
3136   }%
3137   {%
3138     \def\@gls@counter{#1}%
3139   }%
3140 }
```

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.

```
3141 \define@key{glslink}{format}{%
3142   \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.

```
3143 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3144 \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.

```
3145 \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.

```
3146 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3147 \newcommand*{\glsifhyper}[2]{%
3148 \glslinkvar{#1}{#2}{#1}%
3149 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3150 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3151 }
```

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

```
3152 \newcommand*{\@gls@hyp@opt}[1]{%
3153 \let\glslinkvar\@firstofthree
3154 \let\@gls@hyp@opt@cs#1\relax
3155 \@ifstar{\s@gls@hyp@opt}%
3156 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3157 }
```

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

```
3158 \newcommand*{\s@gls@hyp@opt}[1] []{%
3159 \let\glslinkvar\@secondofthree
3160 \@gls@hyp@opt@cs[hyper=false,#1]}
```

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

```
3161 \newcommand*{\p@gls@hyp@opt}[1] []{%
3162 \let\glslinkvar\@thirdofthree
3163 \@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

```
3164 \newrobustcmd*{\glslink}{%  
3165 \@gls@hyp@opt\@gls@@link  
3166 }
```

\@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.

```
3167 \newcommand*{\@gls@@link}[3][\@gls@link]{%  
3168 \glsdoifexistsordo{#2}%  
3169 {%  
3170 \let\do@gls@link@checkfirsthyper\relax  
3171 \@gls@link[#1]{#2}{#3}%  
3172 }%
```

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

```
3173 \glstextformat{#3}%  
3174 }%  
  
3175 \glspostlinkhook  
3176 }
```

glspostlinkhook

```
3177 \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.

```
3178 \newcommand*{\@gls@link@checkfirsthyper}{%  
3179 \ifglsused{\glslabel}%  
3180 {%  
3181 }%  
3182 {%  
3183 \gls@checkisacronymlist\glstype  
3184 \ifglshyperfirst  
3185 \if@glsisacronymlist  
3186 \ifglsacrfootnote  
3187 \KV@glslink@hyperfalse  
3188 \fi  
3189 \fi  
3190 \else  
3191 \KV@glslink@hyperfalse  
3192 \fi  
3193 }%
```

Allow user to hook into this

```
3194 \glslinkcheckfirsthyperhook  
3195 }
```

`checkfirsthyperhook` Allow used to hook into the `\@gls@link@checkfirsthyper` macro  
3196 `\newcommand*{\glslinkcheckfirsthyperhook}{}`

`linkpostsetkeys`  
3197 `\newcommand*{\glslinkpostsetkeys}{}`

`\glsifhyperon` Check the value of the hyper key:  
3198 `\newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}`

`disablehyperinlist` Disable hyperlink if in the “nohyper” list.  
3199 `\newcommand*{\do@glssdisablehyperinlist}{%`  
3200 `\expandafter\DTLifinlist\expandafter{\gls@type}{\@gls@nohyperlist}%`  
3201 `{\KV@glslink@hyperfalse}{}}%`  
3202 `}`

`let@glslink@opts` Hook to set default options for `\@glslink`.  
3203 `\newcommand*{\@gls@setdefault@glslink@opts}{}`

`\@gls@link`  
3204 `\def\@gls@link[#1]#2#3{%`  
Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with tabularx).  
3205 `\leavevmode`  
3206 `\edef\glslabel{\glsdetoklabel{#2}}%`  
Save options in `\@gls@link@opts` and label in `\@gls@link@label`  
3207 `\def\@gls@link@opts{#1}%`  
3208 `\let\@gls@link@label\glslabel`  
3209 `\def\@glsnumberformat{glsnumberformat}%`  
3210 `\edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%`  
If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default  
3211 `\edef\gls@type{\csname glo@\glslabel @type\endcsname}%`  
Save original setting  
3212 `\let\org@ifKV@glslink@hyper\ifKV@glslink@hyper`  
Set defaults:  
3213 `\@gls@setdefault@glslink@opts`  
Switch off hyper setting if the glossary type has been identified in nohyperlist.  
3214 `\do@glssdisablehyperinlist`  
Macros must set this before calling `\@gls@link`. The commands that check the first use flag should set this to `\@gls@link@checkfirsthyper` otherwise it should be set to `\relax`.  
3215 `\do@gls@link@checkfirsthyper`  
3216 `\setkeys{glslink}{#1}%`  
Add a hook for the user to customise things after the keys have been set.  
3217 `\glslinkpostsetkeys`

```

    Store the entry's counter in \theglentrycounter
3218   \@gls@saveentrycounter
    Define sort key if necessary:
3219   \@gls@setsort{\glslabel}%
    (De-tok'ing done by \@do@wrglossary)
3220   \@do@wrglossary{#2}%
3221   \ifKV@glslink@hyper
3222     \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3223   \else
3224     \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3225   \fi
    Restore original setting
3226   \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3227 }

\glolinkprefix
3228 \newcommand*{\glolinkprefix}{glo:}

glsentrycounter  Set default value of entry counter
3229 \def\glsentrycounter{\glscounter}%

saveentrycounter  Need to check if using equation counter in align environment:
3230 \newcommand*{\@gls@saveentrycounter}{%
3231   \def\@gls@Hcounter{}}%
    Are we using equation counter?
3232   \ifthenelse{\equal{\@gls@counter}{equation}}{%
3233     {
    If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentvir as
    may be inside an inner environment.)
3234     \ifcsundef{xatlevel@}%
3235     {%
3236       \edef\theglentrycounter{\expandafter\noexpand
3237         \csname the\@gls@counter\endcsname}%
3238     }%
3239     {%
3240       \ifx\xatlevel@\@empty
3241         \edef\theglentrycounter{\expandafter\noexpand
3242           \csname the\@gls@counter\endcsname}%
3243       \else
3244         \savecounters@
3245         \advance\c@equation by 1\relax
3246         \edef\theglentrycounter{\csname the\@gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3247 \ifcsundef{theH\@gls@counter}%
3248 {%
3249 \def\@gls@Hcounter{\theglsentrycounter}%
3250 }%
3251 {%
3252 \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3253 }%
3254 \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3255 \restorecounters@
3256 \fi
3257 }%
3258 }%
3259 {%

```

Not using equation counter so no special measures:

```

3260 \edef\theglsentrycounter{\expandafter\noexpand
3261 \csname the\@gls@counter\endcsname}%
3262 }%

```

Check if hyperref version of this counter

```

3263 \ifx\@gls@Hcounter\@empty
3264 \ifcsundef{theH\@gls@counter}%
3265 {%
3266 \def\theHglentrycounter{\theglsentrycounter}%
3267 }%
3268 {%
3269 \protected@edef\theHglentrycounter{\expandafter\noexpand
3270 \csname theH\@gls@counter\endcsname}%
3271 }%
3272 \fi
3273 }

```

`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.

```

3274 \def\@set@glo@numformat#1#2#3#4{%
3275 \expandafter\@glo@check@mkidxrangechar#3\@nil
3276 \protected@edef#1{%
3277 \@glo@prefix setentrycounter[#4]{#2}%
3278 \expandafter\string\csname\@glo@suffix\endcsname
3279 }%
3280 \@gls@checkmkidxchars#1%
3281 }

```

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.

```

3282 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3283 \if#1(\relax
3284   \def\@glo@prefix{()%
3285   \if\relax#2\relax
3286     \def\@glo@suffix{glsnumberformat}%
3287   \else
3288     \def\@glo@suffix{#2}%
3289   \fi
3290 \else
3291   \if#1)\relax
3292     \def\@glo@prefix{}}}%
3293   \if\relax#2\relax
3294     \def\@glo@suffix{glsnumberformat}%
3295   \else
3296     \def\@glo@suffix{#2}%
3297   \fi
3298 \else
3299   \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3300 \fi
3301 \fi}

```

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

```

3302 \newcommand*{\@gls@escbsdq}[1]{%
3303   \def\@gls@checkedmkidx{%
3304     \let\gls@xdystring=#1\relax
3305     \@onelevel@sanitize\gls@xdystring
3306     \edef\do@gls@xdycheckbackslash{%
3307       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3308       \@backslashchar\@backslashchar\noexpand\null}%
3309     \do@gls@xdycheckbackslash
3310     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3311     \def\@gls@checkedmkidx{%
3312       \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3313       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

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

```

3314   \@for\@gls@tmp:=\gls@protected@pagefmts\do
3315   {%
3316     \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3317     \@onelevel@sanitize\@gls@sanitized@tmp
3318     \edef\gls@dosubst{%
3319       \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3320       {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3321     }%
3322     \gls@dosubst
3323   }%

```

Assign to required control sequence

```

3324   \let#1=\gls@xdystring

```



3325 }

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

checkmkidxchars

```
3326 \newcommand{\@gls@checkmkidxchars}[1]{%
3327   \ifglxsindy
3328     \@gls@escbsdq{#1}%
3329   \else
3330     \def\@gls@checkedmkidx{%
3331       \expandafter\@gls@checkquote#1\@nil""\null
3332       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3333       \def\@gls@checkedmkidx{%
3334         \expandafter\@gls@checkescquote#1\@nil\""\null
3335         \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3336         \def\@gls@checkedmkidx{%
3337           \expandafter\@gls@checkescactual#1\@nil"??\null
3338           \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3339           \def\@gls@checkedmkidx{%
3340             \expandafter\@gls@checkactual#1\@nil??\null
3341             \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3342             \def\@gls@checkedmkidx{%
3343               \expandafter\@gls@checkbar#1\@nil||\null
3344               \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3345               \def\@gls@checkedmkidx{%
3346                 \expandafter\@gls@checkescbar#1\@nil\\|\null
3347                 \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3348                 \def\@gls@checkedmkidx{%
3349                   \expandafter\@gls@checklevel#1\@nil!!\null
3350                   \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3351                 \fi
3352 }
```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```
3353 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
3354 \newtoks\@gls@tmpb
```

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

```
3355 \def\@gls@checkquote#1"#2"#3\null{%
3356   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3357   \toks@={#1}%
3358   \ifx\@nil#2\null
3359   \ifx\@nil#3\null
3360     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3361     \def\@gls@checkquote{\relax}%
3362   \else
```

```

3363 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3364 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3365 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
3366 \fi
3367 \else
3368 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3369 \@gls@quotechar\@gls@quotechar}%
3370 \ifx\null#3\null
3371 \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
3372 \else
3373 \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
3374 \fi
3375 \fi
3376 \@@gls@checkquote
3377 }

```

s@checkescquote Do the same for \":

```

3378 \def\@gls@checkescquote#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@checkescquote{\relax}%
3385 \else
3386 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3387 \@gls@quotechar\string\"@gls@quotechar
3388 \@gls@quotechar\string\"@gls@quotechar}%
3389 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3390 \fi
3391 \else
3392 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3393 \@gls@quotechar\string\"@gls@quotechar}%
3394 \ifx\null#3\null
3395 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3396 \else
3397 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3398 \fi
3399 \fi
3400 \@@gls@checkescquote
3401 }

```

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

```

3402 \def\@gls@checkescactual#1\?#2\?#3\null{%
3403 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3404 \toks@={#1}%
3405 \ifx\null#2\null
3406 \ifx\null#3\null
3407 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```

```

3408 \def\@gls@checkescactual{\relax}%
3409 \else
3410 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3411 \gls@quotechar\string"\@gls@actualchar
3412 \gls@quotechar\string"\@gls@actualchar}%
3413 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3414 \fi
3415 \else
3416 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3417 \gls@quotechar\string"\@gls@actualchar}%
3418 \ifx\null#3\null
3419 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3420 \else
3421 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3422 \fi
3423 \fi
3424 \@gls@checkescactual
3425 }

```

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

```

3426 \def\@gls@checkescbar#1\|#2\|#3\null{%
3427 \gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3428 \toks@={#1}%
3429 \ifx\null#2\null
3430 \ifx\null#3\null
3431 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3432 \def\@gls@checkescbar{\relax}%
3433 \else
3434 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3435 \gls@quotechar\string"\@gls@encapchar
3436 \gls@quotechar\string"\@gls@encapchar}%
3437 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3438 \fi
3439 \else
3440 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3441 \gls@quotechar\string"\@gls@encapchar}%
3442 \ifx\null#3\null
3443 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3444 \else
3445 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3446 \fi
3447 \fi
3448 \@gls@checkescbar
3449 }

```

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

```

3450 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3451 \gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3452 \toks@={#1}%

```

```

3453 \ifx\null#2\null
3454 \ifx\null#3\null
3455 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3456 \def\@gls@checkesclevel{\relax}%
3457 \else
3458 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3459 \@gls@quotechar\string"\@gls@levelchar
3460 \@gls@quotechar\string"\@gls@levelchar}%
3461 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3462 \fi
3463 \else
3464 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3465 \@gls@quotechar\string"\@gls@levelchar}%
3466 \ifx\null#3\null
3467 \def\@gls@checkesclevel{\@gls@checkesclevel#2!!\null}%
3468 \else
3469 \def\@gls@checkesclevel{\@gls@checkesclevel#2!#3\null}%
3470 \fi
3471 \fi
3472 \@gls@checkesclevel
3473 }

```

\@gls@checkbar and for |:

```

3474 \def\@gls@checkbar#1|#2|#3\null{%
3475 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3476 \toks@={#1}%
3477 \ifx\null#2\null
3478 \ifx\null#3\null
3479 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3480 \def\@gls@checkbar{\relax}%
3481 \else
3482 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3483 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
3484 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3485 \fi
3486 \else
3487 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3488 \@gls@quotechar\@gls@encapchar}%
3489 \ifx\null#3\null
3490 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3491 \else
3492 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3493 \fi
3494 \fi
3495 \@gls@checkbar
3496 }

```

@gls@checklevel and for !:

```

3497 \def\@gls@checklevel#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@checklevel{\relax}%
3504   \else
3505     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3506       \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3507     \def\@gls@checklevel{\@gls@checklevel#3\null}%
3508   \fi
3509 \else
3510   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3511     \@gls@quotechar\@gls@levelchar}%
3512   \ifx\null#3\null
3513     \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3514   \else
3515     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3516   \fi
3517 \fi
3518 \@gls@checklevel
3519 }

```

gls@checkactual and for ?:

```

3520 \def\@gls@checkactual#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@checkactual{\relax}%
3527     \else
3528       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3529         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3530       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3531     \fi
3532   \else
3533     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3534       \@gls@quotechar\@gls@actualchar}%
3535     \ifx\null#3\null
3536       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3537     \else
3538       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3539     \fi
3540   \fi
3541   \@gls@checkactual
3542 }

```

s@xdycheckquote As before but for use with xindy

```

3543 \def\@gls@xdycheckquote#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@xdycheckquote{\relax}%
3550     \else
3551       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3552         \string"\string"}%
3553       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3554     \fi
3555   \else
3556     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3557       \string"}%
3558     \ifx\null#3\null
3559       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3560     \else
3561       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3562     \fi
3563   \fi
3564   \@gls@xdycheckquote
3565 }

```

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

```

3566 \edef\def\@gls@xdycheckbackslash{%
3567   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3568     ##2\@backslashchar##3\noexpand\null{%
3569     \noexpand\@gls@tmpb=\noexpand\expandafter
3570       {\noexpand\@gls@checkedmkidx}%
3571     \noexpand\toks@={##1}%
3572     \noexpand\ifx\noexpand\null##2\noexpand\null
3573       \noexpand\ifx\noexpand\null##3\noexpand\null
3574         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3575           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3576         \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3577       \noexpand\else
3578         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3579           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3580           \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3581       \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3582         \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3583       \noexpand\fi
3584     \noexpand\else
3585       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3586         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3587         \@backslashchar\@backslashchar}%
3588     \noexpand\ifx\noexpand\null##3\noexpand\null
3589       \noexpand\def\noexpand\@gls@xdycheckbackslash{%

```

```

3590     \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3591     \@backslashchar\noexpand\null}%
3592 \noexpand\else
3593     \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3594     \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3595     ##3\noexpand\null}%
3596 \noexpand\fi
3597 \noexpand\fi
3598 \noexpand\@gls@xdycheckbackslash
3599 }%
3600 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3601 \def@gls@xdycheckbackslash

```

glsdohypertarget

```

3602 \newlength\gls@tmplen
3603 \newcommand*\glsdohypertarget}[2]{%
3604   \settoheight{\gls@tmplen}{#2}%
3605   \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3606 }

```

glsdohyperlink

```

3607 \newcommand*\glsdohyperlink}[2]{%
3608   \hyperlink{#1}{#2}%
3609 }

```

glsdonohyperlink

```

3610 \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.

```

3611 \ifcsundef{hyperlink}%
3612 {%
3613   \let\@glslink\glsdonohyperlink
3614 }%
3615 {%
3616   \let\@glslink\glsdohyperlink
3617 }

```

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

```

3618 \ifcsundef{hypertarget}%
3619 {%
3620   \let\@glstarget\@secondoftwo
3621 }%
3622 {%
3623   \let\@glstarget\glsdohypertarget
3624 }

```

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

`\glsdisablehyper`

```
3625 \newcommand{\glsdisablehyper}{%
3626   \KV@glslink@hyperfalse
3627   \let\@glslink\glsdonohyperlink
3628   \let\@glstarget\@secondoftwo
3629 }
```

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

`\glsenablehyper`

```
3630 \newcommand{\glsenablehyper}{%
3631   \KV@glslink@hypertrue
3632   \let\@glslink\glsdohyperlink
3633   \let\@glstarget\glsdohypertarget
3634 }
```

Provide some convenience commands if not already defined:

```
3635 \providecommand{\@firstofthree}[3]{#1}
3636 \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`

```
3637 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

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

`\@gls`

```
3638 \newcommand*{\@gls}[2][ ]{%
3639   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[ ]}%
3640 }
```

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

```
3641 \def\@gls@#1#2[#3]{%
3642   \glsdoifexists{#2}%
3643   {%
3644     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```



```

3645 \let\glsifplural\@secondoftwo
3646 \let\glscapscase\@firstofthree
3647 \let\glscustomtext\@empty
3648 \def\glsinsert{#3}%

```

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

```

3649 \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.

```

3650 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3651 \ifKV@glslink@local
3652 \glslocalunset{#2}%
3653 \else
3654 \glsunset{#2}%
3655 \fi
3656 }%

```

```

3657 \glspostlinkhook
3658 }

```

\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

```

3659 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}

```

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

```

3660 \newcommand*{\@Gls}[2][{}]{%
3661 \new@ifnextchar[\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[{}]}%
3662 }

```

\@Gls@ Read in the final optional argument:

```

3663 \def\@Gls@#1#2[#3]{%
3664 \glsdoifexists{#2}%
3665 {%
3666 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3667 \let\glsifplural\@secondoftwo
3668 \let\glscapscase\@secondofthree
3669 \let\glscustomtext\@empty
3670 \def\glsinsert{#3}%

```

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

```

3671 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call `\@gls@link` If footnote package option has been used and the glossary type is `\acronymstype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3672 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3673 \ifKV@glslink@local
3674 \glslocalunset{#2}%
3675 \else
3676 \glsunset{#2}%
3677 \fi
3678 }%
```

```
3679 \glspostlinkhook
3680 }
```

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

`\GLS`

```
3681 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

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

```
3682 \newcommand*{\@GLS}[2][\@GLS@{#1}{#2}]{\@GLS@{#1}{#2}[\@GLS@{#1}{#2}[]]}%
3683 \new@ifnextchar[\@GLS@{#1}{#2}]{\@GLS@{#1}{#2}[\@GLS@{#1}{#2}[]]}%
3684 }
```

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

```
3685 \def\@GLS@#1#2[#3]{%
3686 \glsdoifexists{#2}%
3687 {%
3688 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3689 \let\glsifplural\@secondoftwo
3690 \let\glscapscase\@thirdofthree
3691 \let\glscustomtext\@empty
3692 \def\glsinsert{#3}%
```

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

```
3693 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@gls@link` If footnote package option has been used and the glossary type is `\acronymstype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3694 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3695 \ifKV@glslink@local
3696 \glslocalunset{#2}%
3697 \else
3698 \glsunset{#2}%
3699 \fi
3700 }%
```

```

3701 \glspostlinkhook
3702 }

```

\glspl behaves in the same way as \gls except it uses the plural form.

\glspl

```

3703 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}

```

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

```

3704 \newcommand*{\@glspl}[2][\@gls@hyp@opt\@glspl]{%
3705   \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}[]}%
3706 }

```

\@glspl@ Read in the final optional argument:

```

3707 \def\@glspl@#1#2[#3]{%
3708   \glsdoifexists{#2}%
3709   {%
3710     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3711     \let\glsifplural\@firstoftwo
3712     \let\glsapscase\@firstofthree
3713     \let\glscustomtext\@empty
3714     \def\glsinsert{#3}%

```

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

```

3715   \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.

```

3716   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3717   \ifKV@glslink@local
3718     \glslocalunset{#2}%
3719   \else
3720     \glsunset{#2}%
3721   \fi
3722 }%

```

```

3723 \glspostlinkhook
3724 }

```

\Glspl behaves in the same way as \glspl, 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

```

3725 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}

```

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

```
3726 \newcommand*{\@Glspl}[2] [] {%
3727   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2} []}%
3728 }
```

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

```
3729 \def\@Glspl@#1#2[#3] {%
3730   \glsdoifexists{#2}%
3731   {%
3732     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3733     \let\glsifplural\@firstoftwo
3734     \let\glsupcase\@secondofthree
3735     \let\glscustomtext\@empty
3736     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`). This needs to be expanded so that the `\@glo@text` can be passed to `\xmakefirstuc`. Note that `\@gls@link` sets `\glstyle`.

```
3737   \def\@glo@text{\csname gls@\glstyle @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.

```
3738   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3739   \ifKV@glslink@local
3740     \glslocalunset{#2}%
3741   \else
3742     \glsunset{#2}%
3743   \fi
3744 }%
3745 \glspostlinkhook
3746 }
```

`\GLSp1` behaves like `\glsp1` except that all the link text is converted to uppercase.

`\GLSp1`

```
3747 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

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

```
3748 \newcommand*{\@GLSp1}[2] [] {%
3749   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
3750 }
```

`\@GLSp1` Read in the final optional argument:

```
3751 \def\@GLSp1@#1#2[#3] {%
3752   \glsdoifexists{#2}%
3753   {%
3754     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3755 \let\glsifplural\@firstoftwo
3756 \let\glsifscapscase\@thirdofthree
3757 \let\glsifcustomtext\@empty
3758 \def\glsinsert{#3}%

```

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

```

3759 \def\@glo@text{\csname gls@\glstyp @entryfmt\endcsname}%

```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3760 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3761 \ifKV@gls@link@local
3762 \glslocalunset{#2}%
3763 \else
3764 \glsunset{#2}%
3765 \fi
3766 }%

```

```

3767 \glspostlinkhook
3768 }

```

`\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:

```

3769 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3770 \newcommand*{\@glsdisp}[3] [] {%
3771 \glsdoifexists{#2}{%
3772 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3773 \let\glsifplural\@secondoftwo
3774 \let\glsifscapscase\@firstofthree
3775 \def\glsifcustomtext{#3}%
3776 \def\glsinsert{}%

```

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

```

3777 \def\@glo@text{\csname gls@\glstyp @entryfmt\endcsname}%

```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3778 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```
3779 \ifKV@glslink@local
3780 \glsllocalunset{#2}%
3781 \else
3782 \glunset{#2}%
3783 \fi
3784 }%

3785 \glspostlinkhook
3786 }
```

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.

```
3787 \newcommand*{\@gl@link@nocheckfirsthyper}{}
```

@gl@field@link

```
3788 \newcommand{\@gl@field@link}[3]{%
3789 \glsoifexists{#2}%
3790 {%
3791 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3792 \@gl@link[#1]{#2}{#3}%
3793 }%

3794 \glspostlinkhook
3795 }
```

\glstext behaves like \gls except it always uses the value given by the text key and it doesn't mark the entry as used.

\glstext

```
3796 \newrobustcmd*{\glstext}{\@gl@hyp@opt\@glstext}
```

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

```
3797 \newcommand*{\@glstext}[2][{}]{%
3798 \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3799 \def\@glstext@#1#2[#3]{%
3800 \@gl@field@link{#1}{#2}{\glstentrytext{#2}{#3}%
3801 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

\GLStext

```
3802 \newrobustcmd*{\GLStext}{\@gl@hyp@opt\@GLStext}
```

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

```
3803 \newcommand*{\@GLStext}[2][{}]{%
3804 \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3805 \def\@GLStext@#1#2[#3]{%
3806   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}#3}}%
3807 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
3808 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@GLstext}
```

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

```
3809 \newcommand*{\@GLstext}[2] [] {%
3810   \new@ifnextchar[{\@GLstext@{#1}{#2}}{\@GLstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3811 \def\@GLstext@#1#2[#3]{%
3812   \@gls@field@link{#1}{#2}{\Glstentrytext{#2}#3}%
3813 }
```

`\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`

```
3814 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

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

```
3815 \newcommand*{\@glsfirst}[2] [] {%
3816   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3817 \def\@glsfirst@#1#2[#3]{%
3818   \@gls@field@link{#1}{#2}{\glstentryfirst{#2}#3}%
3819 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
3820 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

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

```
3821 \newcommand*{\@Glsfirst}[2] [] {%
3822   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3823 \def\@Glsfirst@#1#2[#3]{%
3824   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3825 }
```

`\GLSfirst` behaves like `\Glsfirst` except it displays the text in uppercase.

`\GLSfirst`

```
3826 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

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

```
3827 \newcommand*{\@GLSfirst}[2] [] {%  
3828   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3829 \def\@GLSfirst@#1#2[#3] {%  
3830   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%  
3831 }
```

`\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`

```
3832 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

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

```
3833 \newcommand*{\@glsplural}[2] [] {%  
3834   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3835 \def\@glsplural@#1#2[#3] {%  
3836   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}}%  
3837 }
```

`\Glsplural` behaves like `\glsplural` except that the first letter is converted to uppercase.

`\Glsplural`

```
3838 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

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

```
3839 \newcommand*{\@Glsplural}[2] [] {%  
3840   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3841 \def\@Glsplural@#1#2[#3] {%  
3842   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}}%  
3843 }
```

`\GLSplural` behaves like `\glsplural` except that the text is converted to uppercase.

`\GLSplural`

```
3844 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

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

```
3845 \newcommand*{\@GLSplural}[2] [] {%  
3846   \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3847 \def\@GLSplural@#1#2[#3] {%  
3848   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%  
3849 }
```

`\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`

```
3850 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

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

```
3851 \newcommand*{\@glsfirstplural}[2] [] {%
```

```
3852   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3853 \def\@glsfirstplural@#1#2[#3] {%
```

```
3854   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
```

```
3855 }
```

`\Glsfirstplural` behaves like `\glsfirstplural` except that the first letter is converted to uppercase.

`\Glsfirstplural`

```
3856 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

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

```
3857 \newcommand*{\@Glsfirstplural}[2] [] {%
```

```
3858   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3859 \def\@Glsfirstplural@#1#2[#3] {%
```

```
3860   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
```

```
3861 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
3862 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

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

```
3863 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3864   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3865 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3866   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
```

```
3867 }
```

`\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`

```
3868 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

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

```
3869 \newcommand*{\@glsname}[2] [] {%
```

```
3870   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3871 \def\@glsname@#1#2[#3]{%
3872   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
3873 }
```

\Glsname behaves like \glsname except that the first letter is converted to uppercase.

\Glsname

```
3874 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

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

```
3875 \newcommand*{\@Glsname}[2][\%]
3876   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3877 \def\@Glsname@#1#2[#3]{%
3878   \@gls@field@link{#1}{#2}{\Glsentryname{#2}#3}%
3879 }
```

\GLSname behaves like \glsname except that the link text is converted to uppercase.

\GLSname

```
3880 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

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

```
3881 \newcommand*{\@GLSname}[2][\%]
3882   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3883 \def\@GLSname@#1#2[#3]{%
3884   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3885 }
```

\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

```
3886 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

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

```
3887 \newcommand*{\@glsdesc}[2][\%]
3888   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3889 \def\@glsdesc@#1#2[#3]{%
3890   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3891 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3892 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```

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

```
3893 \newcommand*{\@GLSdesc}[2] [] {%
3894   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3895 \def\@GLSdesc@#1#2[#3] {%
3896   \@gls@field@link{#1}{#2}{\Glsentrydesc{#2}#3}%
3897 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3898 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

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

```
3899 \newcommand*{\@GLSdesc}[2] [] {%
3900   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3901 \def\@GLSdesc@#1#2[#3] {%
3902   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentrydesc{#2}#3}}%
3903 }
```

\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

```
3904 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

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

```
3905 \newcommand*{\@glsdescplural}[2] [] {%
3906   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3907 \def\@glsdescplural@#1#2[#3] {%
3908   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3909 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
3910 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

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

```
3911 \newcommand*{\@Glsdescplural}[2] [] {%
3912   \new@ifnextchar [{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3913 \def\@Glsdescplural@#1#2[#3] {%
3914   \@gls@field@link{#1}{#2}{\Glsentrydescplural{#2}#3}%
3915 }
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
3916 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

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

```
3917 \newcommand*{\@GLSdescplural}[2] [] {%
```

```
3918   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3919 \def\@GLSdescplural@#1#2[#3] {%
```

```
3920   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
```

```
3921 }
```

`\glsymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glsymbol`

```
3922 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

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

```
3923 \newcommand*{\@glsymbol}[2] [] {%
```

```
3924   \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3925 \def\@glsymbol@#1#2[#3] {%
```

```
3926   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}}%
```

```
3927 }
```

`\Glsymbol` behaves like `\glsymbol` except that the first letter is converted to uppercase.

`\Glsymbol`

```
3928 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

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

```
3929 \newcommand*{\@Glsymbol}[2] [] {%
```

```
3930   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3931 \def\@Glsymbol@#1#2[#3] {%
```

```
3932   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}}%
```

```
3933 }
```

`\GLSsymbol` behaves like `\glsymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3934 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

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

```
3935 \newcommand*{\@GLSsymbol}[2] [] {%
```

```
3936   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3937 \def\@GLSsymbol@#1#2[#3]{%
3938   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
3939 }
```

`\glsymbolplural` behaves like `\gls` except it always uses the value given by the symbolplural key and it doesn't mark the entry as used.

`glsymbolplural`

```
3940 \newrobustcmd*{\glsymbolplural}{\@gls@hyp@opt\@glsymbolplural}
```

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

```
3941 \newcommand*{\@glsymbolplural}[2][\%]
3942   \new@ifnextchar[{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3943 \def\@glsymbolplural@#1#2[#3]{%
3944   \@gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}%
3945 }
```

`\Glsymbolplural` behaves like `\glsymbolplural` except that the first letter is converted to uppercase.

`Glsymbolplural`

```
3946 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

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

```
3947 \newcommand*{\@Glsymbolplural}[2][\%]
3948   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3949 \def\@Glsymbolplural@#1#2[#3]{%
3950   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3951 }
```

`\GLSsymbolplural` behaves like `\glsymbolplural` except that the link text is converted to uppercase.

`GLSsymbolplural`

```
3952 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

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

```
3953 \newcommand*{\@GLSsymbolplural}[2][\%]
3954   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3955 \def\@GLSsymbolplural@#1#2[#3]{%
3956   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
3957 }
```

`\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`

```
3958 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

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

```
3959 \newcommand*{\@glsuseri}[2] [] {%
```

```
3960   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3961 \def\@glsuseri@#1#2[#3]{%
```

```
3962   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
```

```
3963 }
```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```
3964 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

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

```
3965 \newcommand*{\@Glsuseri}[2] [] {%
```

```
3966   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3967 \def\@Glsuseri@#1#2[#3]{%
```

```
3968   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
```

```
3969 }
```

`\GLSuseri` behaves like `\glsuseri` except that the link text is converted to uppercase.

`\GLSuseri`

```
3970 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

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

```
3971 \newcommand*{\@GLSuseri}[2] [] {%
```

```
3972   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3973 \def\@GLSuseri@#1#2[#3]{%
```

```
3974   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
```

```
3975 }
```

`\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`

```
3976 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

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

```
3977 \newcommand*{\@glsuserii}[2] [] {%
```

```
3978   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3979 \def\@glsuserii@#1#2[#3]{%
```

```
3980   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
```

```
3981 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3982 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

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

```
3983 \newcommand*{\@Glsuserii}[2] [] {%
```

```
3984   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3985 \def\@Glsuserii@#1#2[#3] {%
```

```
3986   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
3987 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
3988 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

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

```
3989 \newcommand*{\@GLSuserii}[2] [] {%
```

```
3990   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3991 \def\@GLSuserii@#1#2[#3] {%
```

```
3992   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuserii{#2}#3}}%
```

```
3993 }
```

\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

```
3994 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

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

```
3995 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3996   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3997 \def\@glsuseriii@#1#2[#3] {%
```

```
3998   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
```

```
3999 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
4000 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

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

```
4001 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
4002   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4003 \def\@Glsuseriii@#1#2[#3]{%
4004   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
4005 }
```

\GLSuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
4006 \newrobustcmd*{\GLSuseriii}{\@gls@hyp@opt\@GLSuseriii}
```

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

```
4007 \newcommand*{\@GLSuseriii}[2][\@GLSuseriii@#1]{%
4008   \new@ifnextchar[\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4009 \def\@GLSuseriii@#1#2[#3]{%
4010   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
4011 }
```

\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

```
4012 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

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

```
4013 \newcommand*{\@glsuseriv}[2][\@glsuseriv@#1]{%
4014   \new@ifnextchar[\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4015 \def\@glsuseriv@#1#2[#3]{%
4016   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
4017 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
4018 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

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

```
4019 \newcommand*{\@Glsuseriv}[2][\@Glsuseriv@#1]{%
4020   \new@ifnextchar[\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
4021 \def\@Glsuseriv@#1#2[#3]{%
4022   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
4023 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
4024 \newrobustcmd*{\GLSuseriv}{\@gls@hyp@opt\@GLSuseriv}
```



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

```
4025 \newcommand*{\@GLSuseriv}[2] [] {%
4026   \new@ifnextchar [{\@GLSuseriv@{#1}{#2}}]{\@GLSuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4027 \def\@GLSuseriv@#1#2[#3] {%
4028   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%
4029 }
```

\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

```
4030 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

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

```
4031 \newcommand*{\@glsuserv}[2] [] {%
4032   \new@ifnextchar [{\@glsuserv@{#1}{#2}}]{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4033 \def\@glsuserv@#1#2[#3] {%
4034   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}}%
4035 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
4036 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

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

```
4037 \newcommand*{\@Glsuserv}[2] [] {%
4038   \new@ifnextchar [{\@Glsuserv@{#1}{#2}}]{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4039 \def\@Glsuserv@#1#2[#3] {%
4040   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}}%
4041 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4042 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

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

```
4043 \newcommand*{\@GLSuserv}[2] [] {%
4044   \new@ifnextchar [{\@GLSuserv@{#1}{#2}}]{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4045 \def\@GLSuserv@#1#2[#3] {%
4046   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%
4047 }
```

\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

```
4048 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

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

```
4049 \newcommand*{\@glsuservi}[2] [] {%
```

```
4050   \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
4051 \def\@glsuservi@#1#2[#3] {%
```

```
4052   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
```

```
4053 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4054 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

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

```
4055 \newcommand*{\@Glsuservi}[2] [] {%
```

```
4056   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
4057 \def\@Glsuservi@#1#2[#3] {%
```

```
4058   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
4059 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4060 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

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

```
4061 \newcommand*{\@GLSuservi}[2] [] {%
```

```
4062   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]
```

Read in the final optional argument:

```
4063 \def\@GLSuservi@#1#2[#3] {%
```

```
4064   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
```

```
4065 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4066 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

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

```
4067 \newcommand*{\@ns@acrshort}[2] [] {%
```

```
4068   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]
```

```
4069 }
```

Read in the final optional argument:

```
4070 \def\@acrshort#1#2[#3] {%
```

```
4071   \glsdoifexists{#2}%
```

```
4072   {%
```

```

4073 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

4074 \let\gl@sifplural\@secondoftwo
4075 \let\gl@scapscase\@firstofthree
4076 \let\gl@insert\@empty
4077 \def\glscustomtext{%
4078 \acronymfont{\gl@sentryshort{#2}}#3%
4079 }%

```

Call \@gl@link Note that \@gl@link sets \glstype.

```

4080 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4081 }%

4082 \glspostlinkhook
4083 }

```

## \Acrshort

```

4084 \newrobustcmd*{\Acrshort}{\@gl@hyp@opt\@ns@Acrshort}

```

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

```

4085 \newcommand*{\ns@Acrshort}[2][{}]{%
4086 \new@ifnextchar[\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2}[{}]}%
4087 }

```

Read in the final optional argument:

```

4088 \def\@Acrshort#1#2[#3]{%
4089 \gl@sdoifexists{#2}%
4090 {%
4091 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper

4092 \def\glslabel{#2}%
4093 \let\gl@sifplural\@secondoftwo
4094 \let\gl@scapscase\@secondofthree
4095 \let\gl@insert\@empty
4096 \def\glscustomtext{%
4097 \acronymfont{\gl@sentryshort{#2}}#3%
4098 }%

```

Call \@gl@link Note that \@gl@link sets \glstype.

```

4099 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
4100 }%

4101 \glspostlinkhook
4102 }

```

## \ACRshort

```

4103 \newrobustcmd*{\ACRshort}{\@gl@hyp@opt\@ns@ACRshort}

```

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

```
4104 \newcommand*{\ns@ACRshort}[2][\%
4105   \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
4106 }
```

Read in the final optional argument:

```
4107 \def\@ACRshort#1#2[#3]{%
4108   \glsdoifexists{#2}%
4109   {%
4110     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4111     \def\glslabel{#2}%
4112     \let\glsifplural\@secondoftwo
4113     \let\glscapscase\@thirdofthree
4114     \let\glsinsert\@empty
4115     \def\glscustomtext{%
4116       \mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}%
4117     }%
```

Call \@gl@link Note that \@gl@link sets \gls type.

```
4118   \@gl@link[#1]{#2}{\csname gls@\gls type @entryfmt\endcsname}%
4119   }%
4120   \glspostlinkhook
4121 }
```

Short plural:

\acrshortpl

```
4122 \newrobustcmd*{\acrshortpl}{\@gl@hyp@opt\ns@acrshortpl}
```

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

```
4123 \newcommand*{\ns@acrshortpl}[2][\%
4124   \new@ifnextchar[\@acrshortpl{#1}{#2}]{\@acrshortpl{#1}{#2}[]}%
4125 }
```

Read in the final optional argument:

```
4126 \def\@acrshortpl#1#2[#3]{%
4127   \glsdoifexists{#2}%
4128   {%
4129     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4130     \def\glslabel{#2}%
4131     \let\glsifplural\@firstoftwo
4132     \let\glscapscase\@firstofthree
4133     \let\glsinsert\@empty
4134     \def\glscustomtext{%
4135       \acronymfont{\glsentryshortpl{#2}}#3%
4136     }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4137   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4138   }%

4139   \glspostlinkhook
4140 }
```

\Acrshortpl

```
4141 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}
```

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

```
4142 \newcommand*{\ns@Acrshortpl}[2] [] {%
4143   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%
4144 }
```

Read in the final optional argument:

```
4145 \def\@Acrshortpl#1#2[#3]{%
4146   \glsdoifexists{#2}%
4147   {%

4148     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4149     \def\glslabel{#2}%
4150     \let\glsifplural\@firstoftwo
4151     \let\gls caps case\@secondofthree
4152     \let\glsinsert\@empty
4153     \def\gls custom text{%
4154       \acronymfont{\Glsentryshortpl{#2}}#3%
4155     }%
```

Call \@gls@link Note that \@gls@link sets \glstype.

```
4156   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4157   }%

4158   \glspostlinkhook
4159 }
```

\ACRshortpl

```
4160 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}
```

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

```
4161 \newcommand*{\ns@ACRshortpl}[2] [] {%
4162   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%
4163 }
```

Read in the final optional argument:

```
4164 \def\@ACRshortpl#1#2[#3]{%
4165   \glsdoifexists{#2}%
4166   {%

4167     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```

4168 \def\glslabel{#2}%
4169 \let\glsifplural\@firstoftwo
4170 \let\glsupcase\@thirdofthree
4171 \let\glsinsert\@empty
4172 \def\glscustomtext{%
4173     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
4174 }%

Call \@gls@link Note that \@gls@link sets \glstype.
4175 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4176 }%

4177 \glspostlinkhook
4178 }

```

\acrlong

```

4179 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\ns@acrlong}

```

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

```

4180 \newcommand*{\ns@acrlong}[2] [] {%
4181     \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}%
4182 }

```

Read in the final optional argument:

```

4183 \def\@acrlong#1#2[#3] {%
4184     \glsdoifexists{#2}%
4185     {%
4186         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4187         \def\glslabel{#2}%
4188         \let\glsifplural\@secondoftwo
4189         \let\glsupcase\@firstofthree
4190         \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4191     \def\glscustomtext{%
4192         \glsentrylong{#2}#3%
4193     }%

```

Call \@gls@link Note that \@gls@link sets \glstype.

```

4194     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4195     }%

4196     \glspostlinkhook
4197 }

```

\Acrlong

```

4198 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}

```

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

```
4199 \newcommand*{\ns@Acrlong}[2][{}]{%
4200   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%
4201 }
```

Read in the final optional argument:

```
4202 \def\@Acrlong#1#2[#3]{%
4203   \glsdoifexists{#2}%
4204   {%
4205     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4206     \def\glslabel{#2}%
4207     \let\glsifplural\@secondoftwo
4208     \let\glscapscase\@secondofthree
4209     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4210   \def\glscustomtext{%
4211     \Glsentrylong{#2}#3%
4212   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4213   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4214   }%
4215   \glspostlinkhook
4216 }
```

\ACRlong

```
4217 \newrobustcmd*{\ACRlong}{\@gl@hyp@opt\ns@ACRlong}
```

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

```
4218 \newcommand*{\ns@ACRlong}[2][{}]{%
4219   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4220 }
```

Read in the final optional argument:

```
4221 \def\@ACRlong#1#2[#3]{%
4222   \glsdoifexists{#2}%
4223   {%
4224     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4225     \def\glslabel{#2}%
4226     \let\glsifplural\@secondoftwo
4227     \let\glscapscase\@thirdofthree
4228     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4229 \def\glscustomtext{%
4230 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4231 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4232 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4233 }%

4234 \glspostlinkhook
4235 }
```

Short plural:

\acrlongpl

```
4236 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\ns@acrlongpl}
```

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

```
4237 \newcommand*{\ns@acrlongpl}[2][\%
4238 \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2}[]}%
4239 }
```

Read in the final optional argument:

```
4240 \def\@acrlongpl#1#2[#3]{%
4241 \glsdoidexists{#2}%
4242 {%
4243 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4244 \def\glslabel{#2}%
4245 \let\glsifplural\@firstoftwo
4246 \let\glscapscase\@firstofthree
4247 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4248 \def\glscustomtext{%
4249 \glsentrylongpl{#2}#3%
4250 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4251 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4252 }%

4253 \glspostlinkhook
4254 }
```

\Acrlongpl

```
4255 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\ns@Acrlongpl}
```



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

```
4256 \newcommand*{\ns@Acrlongpl}[2][\%
4257   \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%
4258 }
```

Read in the final optional argument:

```
4259 \def\@Acrlongpl#1#2[#3]{%
4260   \glsdoifexists{#2}%
4261   {%
4262     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4263     \def\glslabel{#2}%
4264     \let\glsifplural\@firstoftwo
4265     \let\glscapscase\@secondofthree
4266     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4267   \def\glscustomtext{%
4268     \Glsentrylongpl{#2}#3%
4269   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4270   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4271   }%
4272   \glspostlinkhook
4273 }
```

\ACRlongpl

```
4274 \newrobustcmd*{\ACRlongpl}{\@gl@hyp@opt\ns@ACRlongpl}
```

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

```
4275 \newcommand*{\ns@ACRlongpl}[2][\%
4276   \new@ifnextchar[\@ACRlongpl{#1}{#2}]{\@ACRlongpl{#1}{#2}[]}%
4277 }
```

Read in the final optional argument:

```
4278 \def\@ACRlongpl#1#2[#3]{%
4279   \glsdoifexists{#2}%
4280   {%
4281     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4282     \def\glslabel{#2}%
4283     \let\glsifplural\@firstoftwo
4284     \let\glscapscase\@thirdofthree
4285     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4286 \def\glscustomtext{%
4287 \mfirstucMakeUppercase{\glsentrylongpl{#2}#3}%
4288 }%

Call \@gls@link. Note that \@gls@link sets \glstype.
4289 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4290 }%

4291 \glspostlinkhook
4292 }

```

## 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>}`

```

4293 \newcommand*{\@gls@entry@field}[2]{%
4294 \csname glo@\glsdetoklabel{#1}@#2\endcsname
4295 }

```

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

```

4296 \newcommand*{\glsletentryfield}[3]{%
4297 \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
4298 }

```

`\Gls@entry@field` Generic first letter uppercase version.

`\@Gls@entry@field{<label>}{<field>}`

```

4299 \newcommand*{\@Gls@entry@field}[2]{%
4300 \glsdoifexistsordo{#1}%
4301 {%
4302 \letcs{\@glo@text}{glo@\glsdetoklabel{#1}@#2}%
4303 \ifdef\@glo@text
4304 {%
4305 \xmakefirstuc{\@glo@text}%
4306 }%
4307 }%
4308 ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4309 entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry

```

```

4310      label and the field name}%
4311    }%
4312  }%
4313  {%
4314    ??%
4315  }%
4316 }

```

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`

```

4317 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4318 \newrobustcmd*{\Glsentryname}[1]{%
4319   \@Gls@entryname{#1}%
4320 }

```

`\@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:

```

4321 \newcommand*{\@Gls@entryname}[1]{%
4322   \@Gls@entry@field{#1}{name}%
4323 }

```

`\ls@acentryname` Now the behaviour when `\setacronymstyle` is used:

```

4324 \newcommand*{\@Gls@acentryname}[1]{%
4325   \ifglshaslong{#1}%
4326   {%
4327     \letcs\@glo@text{glo\@glsdetoklabel{#1}@name}%
4328     \expandafter\@gls@getbody\@glo@text{}\@nil
4329     \expandafter\ifx\@gls@body\glsentrylong\relax
4330     \expandafter\Glsentrylong\@gls@rest
4331   \else
4332     \expandafter\ifx\@gls@body\glsentryshort\relax
4333     \expandafter\Glsentryshort\@gls@rest
4334   \else
4335     \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make `\glsentryshort` behave like `\Glsentryshort`. (This is on the assumption that the argument of `\acronymfont` is `\glsentryshort{<label>}`, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

4336     {%
4337       \let\glsentryshort\Glsentryshort
4338       \@glo@text
4339     }%
4340   \else

```

```

4341      \xmakefirstuc{\@glo@text}%
4342      \fi
4343      \fi
4344      \fi
4345  }%
4346  {%

```

Not an acronym

```

4347      \@Gls@entry@field{#1}{name}%
4348  }%
4349 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

`\glentrydesc`

```

4350 \newcommand*{\glentrydesc}[1]{\@Gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4351 \newrobustcmd*{\Glsentrydesc}[1]{%
4352   \@Gls@entry@field{#1}{desc}%
4353 }

```

Plural form:

`entrydescplural`

```

4354 \newcommand*{\glentrydescplural}[1]{%
4355   \@Gls@entry@field{#1}{descplural}%
4356 }

```

`entrydescplural`

```

4357 \newrobustcmd*{\Glsentrydescplural}[1]{%
4358   \@Gls@entry@field{#1}{descplural}%
4359 }

```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

`\glentrytext`

```

4360 \newcommand*{\glentrytext}[1]{\@Gls@entry@field{#1}{text}}

```

`\Glsentrytext`

```

4361 \newrobustcmd*{\Glsentrytext}[1]{%
4362   \@Gls@entry@field{#1}{text}%
4363 }

```

Get the plural form:

\glentryplural

```
4364 \newcommand*{\glentryplural}[1]{%
4365   \@gls@entry@field{#1}{plural}%
4366 }
```

\Glsentryplural

```
4367 \newrobustcmd*{\Glsentryplural}[1]{%
4368   \@Gls@entry@field{#1}{plural}%
4369 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

\glentrysymbol

```
4370 \newcommand*{\glentrysymbol}[1]{%
4371   \@gls@entry@field{#1}{symbol}%
4372 }
```

\Glsentrysymbol

```
4373 \newrobustcmd*{\Glsentrysymbol}[1]{%
4374   \@Gls@entry@field{#1}{symbol}%
4375 }
```

Plural form:

trysymbolplural

```
4376 \newcommand*{\glentrysymbolplural}[1]{%
4377   \@gls@entry@field{#1}{symbolplural}%
4378 }
```

trysymbolplural

```
4379 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4380   \@Gls@entry@field{#1}{symbolplural}%
4381 }
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

\glentryfirst

```
4382 \newcommand*{\glentryfirst}[1]{%
4383   \@gls@entry@field{#1}{first}%
4384 }
```

\Glsentryfirst

```
4385 \newrobustcmd*{\Glsentryfirst}[1]{%
4386   \@Gls@entry@field{#1}{first}%
4387 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

entryfirstplural

```
4388 \newcommand*{\glentryfirstplural}[1]{%
4389   \@gls@entry@field{#1}{firstpl}%
4390 }
```

entryfirstplural

```
4391 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4392   \@Gls@entry@field{#1}{firstpl}%
4393 }
```

entrytitlecase

```
4394 \newrobustcmd*{\@glentrytitlecase}[2]{%
4395   \glsfieldfetch{#1}{#2}{\@gls@value}%
4396   \xcapitalisewords{\@gls@value}%
4397 }
4398 \ifdef\texorpdfstring
4399 {
4400   \newcommand*{\glentrytitlecase}[2]{%
4401     \texorpdfstring
4402       {\@glentrytitlecase{#1}{#2}}%
4403       {\@gls@entry@field{#1}{#2}}%
4404   }
4405 }
4406 {
4407   \newcommand*{\glentrytitlecase}[2]{\@glentrytitlecase{#1}{#2}}
4408 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glentrytype

```
4409 \newcommand*{\glentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glentrysort

```
4410 \newcommand*{\glentrysort}[1]{%
4411   \@gls@entry@field{#1}{sort}%
4412 }
```

\glentryuseri Get the first user key (as specified by the user1 when the entry was defined). The argument is the label associated with the entry.

```
4413 \newcommand*{\glentryuseri}[1]{%
4414   \@gls@entry@field{#1}{useri}%
4415 }
```

\Glsentryuseri

```
4416 \newrobustcmd*{\Glsentryuseri}[1]{%
```

```

4417 \@Gls@entry@field{#1}{useri}%
4418 }

```

`\glentryuserii` Get the second user key (as specified by the user2 when the entry was defined). The argument is the label associated with the entry.

```

4419 \newcommand*{\glentryuserii}[1]{%
4420 \@Gls@entry@field{#1}{userii}%
4421 }

```

`\Glsentryuserii`

```

4422 \newrobustcmd*{\Glsentryuserii}[1]{%
4423 \@Gls@entry@field{#1}{userii}%
4424 }

```

`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.

```

4425 \newcommand*{\glentryuseriii}[1]{%
4426 \@Gls@entry@field{#1}{useriii}%
4427 }

```

`Glsentryuseriii`

```

4428 \newrobustcmd*{\Glsentryuseriii}[1]{%
4429 \@Gls@entry@field{#1}{useriii}%
4430 }

```

`\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.

```

4431 \newcommand*{\glentryuseriv}[1]{%
4432 \@Gls@entry@field{#1}{useriv}%
4433 }

```

`\Glsentryuseriv`

```

4434 \newrobustcmd*{\Glsentryuseriv}[1]{%
4435 \@Gls@entry@field{#1}{useriv}%
4436 }

```

`\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.

```

4437 \newcommand*{\glentryuserv}[1]{%
4438 \@Gls@entry@field{#1}{userv}%
4439 }

```

`\Glsentryuserv`

```

4440 \newrobustcmd*{\Glsentryuserv}[1]{%
4441 \@Gls@entry@field{#1}{userv}%
4442 }

```

`\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.

```
4443 \newcommand*{\glentryuservi}[1]{%
4444   \@gls@entry@field{#1}{uservi}%
4445 }
```

`\Glsentryuservi`

```
4446 \newrobustcmd*{\Glsentryuservi}[1]{%
4447   \@Gls@entry@field{#1}{uservi}%
4448 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4449 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4450 \newrobustcmd*{\Glsentryshort}[1]{%
4451   \@Gls@entry@field{#1}{short}%
4452 }
```

`glentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4453 \newcommand*{\glentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`Glsentryshortpl`

```
4454 \newrobustcmd*{\Glsentryshortpl}[1]{%
4455   \@Gls@entry@field{#1}{shortpl}%
4456 }
```

`\glentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4457 \newcommand*{\glentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4458 \newrobustcmd*{\Glsentrylong}[1]{%
4459   \@Gls@entry@field{#1}{long}%
4460 }
```

`\glentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4461 \newcommand*{\glentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4462 \newrobustcmd*{\Glsentrylongpl}[1]{%
4463   \@Gls@entry@field{#1}{longpl}%
4464 }
```



Short cut macros to access full form:

`\glsentryfull`

```
4465 \newcommand*{\glsentryfull}[1]{%
4466   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4467 }
```

`\Glsentryfull`

```
4468 \newrobustcmd*{\Glsentryfull}[1]{%
4469   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4470 }
```

`\glsentryfullpl`

```
4471 \newcommand*{\glsentryfullpl}[1]{%
4472   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4473 }
```

`\Glsentryfullpl`

```
4474 \newrobustcmd*{\Glsentryfullpl}[1]{%
4475   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4476 }
```

`entrynumberlist` Displays the number list as is.

```
4477 \newcommand*{\glsentrynumberlist}[1]{%
4478   \glsdoifexists{#1}%
4479   {%
4480     \@gls@entry@field{#1}{numberlist}%
4481   }%
4482 }
```

`splaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4483 \@ifpackageloaded{hyperref} {%
4484   \newcommand*{\glsdisplaynumberlist}[1]{%
4485     \GlossariesWarning
4486     {%
4487       \string\glsdisplaynumberlist\space
4488       doesn't work with hyperref.^^JUsing
4489       \string\glsentrynumberlist\space instead%
4490     }%
4491     \glsentrynumberlist{#1}%
4492   }%
4493 }%
4494 {%
4495   \newcommand*{\glsdisplaynumberlist}[1]{%
4496     \glsdoifexists{#1}%
4497     {%
4498       \bgroup
```



```

4534 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4535 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

This key is only used by `\glsaddall`:

```

4536 \define@key{glossadd}{types}{\def\@glo@type{#1}}

```

**`\glsadd[options]{label}`**

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *options* only has two keys: counter and format (the types key will be ignored).

`\glsadd`

```

4537 \newrobustcmd*{\glsadd}[2][]{%

```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4538   \@gls@adjustmode
4539   \glsdoifexists{#2}%
4540   {%
4541     \def\@glsnumberformat{glsnumberformat}%
4542     \edef\@gls@counter{\csname glo@%glsdetoklabel{#2}@counter\endcsname}%
4543     \setkeys{glossadd}{#1}%

```

Store the entry's counter in `\theglsentrycounter`

```

4544   \@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.

```

4545     \@do@wrglossary{#2}%
4546   }%
4547 }

```

`@gls@adjustmode`

```

4548 \newcommand*{\@gls@adjustmode}{}
4549 \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`

```

4550 \newrobustcmd*{\glsaddall}[1][]{%
4551   \edef\@glo@type{\@glo@types}%
4552   \setkeys{glossadd}{#1}%
4553   \forallglsentries[\@glo@type]{\@glo@entry}{%

```

```

4554 \glsadd[#1]{\@glo@entry}%
4555 }%
4556 }

```

```

\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.

```

4557 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4558 \forallglsentries[#1]{\@glo@entry}%
4559 {%
4560 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4561 }%
4562 }

```

```

\glsignore

```

```

4563 \newcommand*{\glsignore}[1]{}

```

## 1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The makeindex actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about makeindex special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgroupname` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

```

\glsopenbrace Define \glsopenbrace to make it easier to write an opening brace to a file.

```

```

4564 \edef\glsopenbrace{\expandafter\@gobble\string\{ }

```

```

\glsclosebrace Define \glsclosebrace to make it easier to write an opening brace to a file.

```

```

4565 \edef\glsclosebrace{\expandafter\@gobble\string\} }

```

```

\glsbackslash Define \glsbackslash to make it easier to write a backslash to a file.

```

```

4566 \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.

```
4567 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4568 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4569 \edef\glstildechar{\string~}
```

`@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4570 \ifglsxindy
4571   \newcommand*{\@glsfirstletter}{A}
4572 \fi
```

`letterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4573 \ifglsxindy
4574   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4575     \renewcommand*{\@glsfirstletter}{#1}}
4576 \else
4577   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4578     \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4579 \fi
```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```
4580 \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.

```
4581 \ifglsxindy
4582   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4583     \renewcommand*{\@glsminrange}{#1}}
4584 \else
4585   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4586     \glsnoxywarning\GlsSetXdyMinRangeLength}
4587 \fi
```

`\writeist`

```
4588 \ifglsxindy
  Code to use if xindy is required.
4589   \def\writeist{%
    Define write register if not already defined
4590     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4591     \@gls@addpredefinedattributes
```

Open the file.

```
4592 \openout\glswrite=\istfilename
```

Write header comment at the start of the file

```
4593 \write\glswrite{;; xindy style file created by the glossaries
4594   package}%
4595 \write\glswrite{;; for document '\jobname' on
4596   \the\year-\the\month-\the\day}%
```

Specify the required styles

```
4597 \write\glswrite{^^J; required styles^^J}
4598 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4599   \ifx\@xdystyle\@empty
4600   \else
4601     \protected@write\glswrite{}{(require
4602       \string"\@xdystyle.xdy\string")}%
4603   \fi
4604 }%
```

List the allowed attributes (possible values used by the format key)

```
4605 \write\glswrite{^^J%
4606   ; list of allowed attributes (number formats)^^J}%
4607 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4608 \write\glswrite{^^J; user defined alphabets^^J}%
4609 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4610 \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.

```
4611 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case where  $\langle Hprefix \rangle$  is empty:

```
4612 \protected@write\glswrite{}{(define-location-class
4613   \string"\@gls@classI\string"^^J\space\space\space
4614   (
4615     :sep "{}{"
4616     \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4617     :sep "}"
4618   )
4619   ^^J\space\space\space
4620   :min-range-length \@glsminrange^^J%
4621   )
4622 }%
```

Nested iteration over all classes:

```
4623 {%
4624   \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4625     \protected@write\glswrite{}{(define-location-class
```

```

4626         \string"\@gls@classII-\@gls@classI\string"
4627         ^^J\space\space\space
4628     (
4629         :sep "{"
4630         \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4631         :sep "{"
4632         \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4633         :sep "}"
4634     )
4635     ^^J\space\space\space
4636     :min-range-length \@glsminrange^^J%
4637 )
4638 }%
4639 }%
4640 }%
4641 }%

```

User defined location classes (needs checking for new location format).

```

4642 \write\glswrite{^^J; user defined location classes}%
4643 \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.)

```

4644 \write\glswrite{^^J; define cross-reference class^^J}%
4645 \write\glswrite{(define-crossref-class \string"see\string"
4646     :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.)

```

4647 \write\glswrite{(markup-crossref-list
4648     :class \string"see\string"^^J\space\space\space
4649     :open \string"\string\glsseeformat\string"
4650     :close \string"{}\string")}%

```

Provide hook to write extra material here (used by `glossaries-extra` to define a `seealso` class).

```

4651 \@xdycrossrefhook

```

List the order to sort the classes.

```

4652 \write\glswrite{^^J; define the order of the location classes}%
4653 \write\glswrite{(define-location-class-order
4654     (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4655 \write\glswrite{^^J; define the glossary markup^^J}%

4656 \write\glswrite{(markup-index^^J\space\space\space
4657     :open \string"\string
4658     \glossarysection[\string\glossarytoctitle]{\string
4659     \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)

```

4660 \@for\@this@ctr:=\@xdycounters\do{%
4661   {%
4662     \@for\@this@attr:=\@xdyattributelist\do{%
4663       \protected@write\glswrite{}\string\providecommand*%
4664         \expandafter\string
4665         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4666         {%
4667           \string\setentrycounter
4668             [\expandafter@gobble\string\#1]{\@this@ctr}%
4669           \expandafter\string
4670           \csname\@this@attr\endcsname
4671             {\expandafter@gobble\string\#2}%
4672         }%
4673     }%
4674 }%
4675 }%
4676 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4677 \write\glswrite{%
4678   \string\begin
4679     {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4680     \space\space:close \string"\glspercentchar\glstildechar n\string
4681     \end{theglossary}\string\glossarypostamble
4682     \glstildechar n\string" ^^J\space\space\space
4683     :tree)}}%

```

Specify what to put between letter groups

```

4684 \write\glswrite{(markup-letter-group-list
4685   :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4686 \write\glswrite{(markup-indexentry
4687   :open \string"\string\relax \string\glsresetentrylist
4688   \glstildechar n\string")}%

```

Specify how to format entries

```

4689 \write\glswrite{(markup-locclass-list :open
4690   \string"\glsopenbrace\string\glossaryentrynumbers
4691   \glsopenbrace\string\relax\space \string"^^J\space\space\space
4692   :sep \string", \string"
4693   :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

4694 \write\glswrite{(markup-locref-list
4695   :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

4696 \write\glswrite{(markup-range
4697   :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.

```

4698 \onelevel@sanitize\gls@suffixF
4699 \onelevel@sanitize\gls@suffixFF
4700 \ifx\gls@suffixF\@empty
4701 \else
4702 \write\glswrite{(markup-range
4703 :close "\gls@suffixF" :length 1 :ignore-end)}}%
4704 \fi
4705 \ifx\gls@suffixFF\@empty
4706 \else
4707 \write\glswrite{(markup-range
4708 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4709 \fi

```

Specify how to format locations.

```

4710 \write\glswrite{^^J; define format to use for locations^^J}%
4711 \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4712 \write\glswrite{^^J; define letter group list format^^J}%
4713 \write\glswrite{(markup-letter-group-list
4714 :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Define letter group headings.

```

4715 \write\glswrite{^^J; letter group headings^^J}%
4716 \write\glswrite{(markup-letter-group
4717 :open-head \string"\string\glsgroupheading
4718 \glsopenbrace\string"^^J\space\space\space
4719 :close-head \string"\glsclosebrace\string")}%

```

Define additional letter groups.

```

4720 \write\glswrite{^^J; additional letter groups^^J}%
4721 \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

4722 \write\glswrite{^^J; additional sort rules^^J}
4723 \write\glswrite{\@xdysortrules}%

```

Hook for any additional information:

```

4724 \@gls@writeisthook

```

Close the style file

```

4725 \closeout\glswrite

```

Suppress any further calls.

```

4726 \let\writeist\relax
4727 }
4728 \else

```

Code to use if makeindex is required.

```

4729 \edef\@gls@actualchar{\string?}
4730 \edef\@gls@encapchar{\string|}
4731 \edef\@gls@levelchar{\string!}
4732 \edef\@gls@quotechar{\string"}%
4733 \let\GlsSetQuote\gls@nosetquote
4734 \def\writeist{\relax
4735   \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4736   \openout\glswrite=\istfilename
4737   \write\glswrite{\glspersentchar\space makeindex style file
4738     created by the glossaries package}
4739   \write\glswrite{\glspersentchar\space for document
4740     '\jobname' on \the\year-\the\month-\the\day}
4741   \write\glswrite{actual '\@gls@actualchar'}
4742   \write\glswrite{encap '\@gls@encapchar'}
4743   \write\glswrite{level '\@gls@levelchar'}
4744   \write\glswrite{quote '\@gls@quotechar'}
4745   \write\glswrite{keyword \string\string\glossaryentry\string}
4746   \write\glswrite{preamble \string\string\glossarysection[\string
4747     \glossarytoctitle]{\string\glossarytitle}\string
4748     \glossarypreamble\string\n\string\begin{theglossary}\string
4749     \glossaryheader\string\n\string}
4750   \write\glswrite{postamble \string\string%\string\n\string
4751     \end{theglossary}\string\glossarypostamble\string\n
4752     \string}
4753   \write\glswrite{group_skip \string\string\glsgroupskip\string\n
4754     \string}
4755   \write\glswrite{item_0 \string\string%\string\n\string}
4756   \write\glswrite{item_1 \string\string%\string\n\string}
4757   \write\glswrite{item_2 \string\string%\string\n\string}
4758   \write\glswrite{item_01 \string\string%\string\n\string}
4759   \write\glswrite{item_x1
4760     \string\string\relax \string\glsresetentrylist\string\n
4761     \string}
4762   \write\glswrite{item_12 \string\string%\string\n\string}
4763   \write\glswrite{item_x2
4764     \string\string\relax \string\glsresetentrylist\string\n
4765     \string}

4766   \write\glswrite{delim_0 \string\string\{\string
4767     \glossaryentrynumbers\string\{\string\relax \string}
4768   \write\glswrite{delim_1 \string\string\{\string
4769     \glossaryentrynumbers\string\{\string\relax \string}
4770   \write\glswrite{delim_2 \string\string\{\string
4771     \glossaryentrynumbers\string\{\string\relax \string}
4772   \write\glswrite{delim_t \string\string\}\string\}\string}
4773   \write\glswrite{delim_n \string\string\delimN \string}
4774   \write\glswrite{delim_r \string\string\delimR \string}
4775   \write\glswrite{headings_flag 1}
4776   \write\glswrite{heading_prefix

```

```

4777     \string"\string\glsgroupheading\string\{\string"}
4778 \write\glswrite{heading_suffix
4779     \string"\string\}\string\relax
4780     \string\glsgroupheading \string"}
4781 \write\glswrite{symhead_positive \string"glssymbols\string"}
4782 \write\glswrite{numhead_positive \string"glslnumbers\string"}
4783 \write\glswrite{page_compositor \string"glscpositor\string"}
4784 \@gls@escbsdq\gls@suffixF
4785 \@gls@escbsdq\gls@suffixFF
4786 \ifx\gls@suffixF\@empty
4787 \else
4788     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4789 \fi
4790 \ifx\gls@suffixFF\@empty
4791 \else
4792     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4793 \fi

```

Hook for any additional information:

```

4794 \@gls@writeisthook

```

Close the file and disable \writeist.

```

4795 \closeout\glswrite
4796 \let\writeist\relax
4797 }
4798 \fi

```

**SetWriteIstHook** Allow user to append information to the style file.

```

4799 \newcommand*\GlsSetWriteIstHook[1]{\renewcommand*\@gls@writeisthook{#1}}
4800 \@onlypremake\GlsSetWriteIstHook

```

**ls@writeisthook**

```

4801 \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.

```

4802 \ifglxindy
4803 \newcommand*\GlsSetQuote[1]{\glsnomakeindexwarning\GlsSetQuote}
4804 \newcommand*\gls@nosetquote[1]{\glsnomakeindexwarning\GlsSetQuote}
4805 \else
4806 \newcommand*\GlsSetQuote[1]{\edef\@gls@quotechar{\string#1}%

```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```

4807 \@ifpackageloaded{tracklang}%
4808 {%
4809     \IfTrackedLanguage{german}%
4810     {%
4811         \def\@gls@extramakeindexopts{-g}%
4812     }%
4813 }%

```

```

4814 }%
4815 {}%

Need to redefine \@gls@checkquote
4816 \edef\@gls@docheckquotedef{%
4817   \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
4818     \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4819     \noexpand\toks@={####1}%
4820     \noexpand\ifx\noexpand\null####2\noexpand\null
4821     \noexpand\ifx\noexpand\null####3\noexpand\null
4822     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4823       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4824     \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
4825     \noexpand\else
4826     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4827       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4828       \noexpand\@gls@quotechar\noexpand\@gls@quotechar
4829       \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4830     \noexpand\def\noexpand\@gls@checkquote{%
4831       \noexpand\@gls@checkquote####3\noexpand\null}%
4832     \noexpand\fi
4833   \noexpand\else
4834     \noexpand\edef\noexpand\@gls@checkedmkidx{%
4835       \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4836       \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4837     \noexpand\ifx\noexpand\null####3\noexpand\null
4838     \noexpand\def\noexpand\@gls@checkquote{%
4839       \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
4840     \noexpand\else
4841     \noexpand\def\noexpand\@gls@checkquote{%
4842       \noexpand\@gls@checkquote####2#1####3\noexpand\null}%
4843     \noexpand\fi
4844     \noexpand\fi
4845     \noexpand\@gls@checkquote
4846   }%
4847 }%

\@gls@docheckquotedef
4848 \edef\@gls@docheckquotedef{%
4849   \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
4850     \noexpand\def\noexpand\@gls@checkedmkidx{%
4851       \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
4852       #1#1\noexpand\null
4853       \noexpand\expandafter\noexpand\@gls@updatechecked
4854       \noexpand\@gls@checkedmkidx{####1}%
4855       \noexpand\def\noexpand\@gls@checkedmkidx{%
4856         \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
4857         \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
4858         \noexpand\null
4859       \noexpand\expandafter\noexpand\@gls@updatechecked
4860       \noexpand\@gls@checkedmkidx{####1}%

```

```

4862 \noexpand\def\noexpand\@gls@checkedmkidx{%
4863 \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
4864 \noexpand\?\noexpand\?\noexpand\null
4865 \noexpand\expandafter\noexpand\@gls@updatechecked
4866 \noexpand\@gls@checkedmkidx{####1}%
4867 \noexpand\def\noexpand\@gls@checkedmkidx{%
4868 \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
4869 \noexpand?\noexpand?\noexpand\null
4870 \noexpand\expandafter\noexpand\@gls@updatechecked
4871 \noexpand\@gls@checkedmkidx{####1}%
4872 \noexpand\def\noexpand\@gls@checkedmkidx{%
4873 \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
4874 \noexpand|\noexpand|\noexpand\null
4875 \noexpand\expandafter\noexpand\@gls@updatechecked
4876 \noexpand\@gls@checkedmkidx{####1}%
4877 \noexpand\def\noexpand\@gls@checkedmkidx{%
4878 \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
4879 \noexpand||\noexpand||\noexpand\null
4880 \noexpand\expandafter\noexpand\@gls@updatechecked
4881 \noexpand\@gls@checkedmkidx{####1}%
4882 \noexpand\def\noexpand\@gls@checkedmkidx{%
4883 \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
4884 \noexpand!\noexpand!\noexpand\null
4885 \noexpand\expandafter\noexpand\@gls@updatechecked
4886 \noexpand\@gls@checkedmkidx{####1}%
4887 }%
4888 }%
4889 \@gls@docheckquotedef
4890 \edef\@gls@docheckquotedef{%
4891 \noexpand\def\noexpand\@gls@checkescquote####1%
4892 \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
4893 ####3\noexpand\null{%
4894 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4895 \noexpand\toks@={####1}%
4896 \noexpand\ifx\noexpand\null####2\noexpand\null
4897 \noexpand\ifx\noexpand\null####3\noexpand\null
4898 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4899 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4900 \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
4901 \noexpand\else
4902 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4903 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4904 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4905 \csname#1\endcsname}\noexpand\@gls@quotechar
4906 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4907 \csname#1\endcsname}\noexpand\@gls@quotechar}%
4908 \noexpand\def\noexpand\@gls@checkescquote{%
4909 \noexpand\@gls@checkescquote####3\noexpand\null}%
4910 \noexpand\fi

```

```

4911 \noexpand\else
4912 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4913 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4914 \noexpand\@gls@quotearch\noexpand\string
4915 \expandonce{\csname#1\endcsname}\noexpand\@gls@quotearch}%
4916 \noexpand\ifx\noexpand\null####3\noexpand\null
4917 \noexpand\def\noexpand\@gls@checkescquote{%
4918 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4919 \expandonce{\csname#1\endcsname}\noexpand\null}%
4920 \noexpand\else
4921 \noexpand\def\noexpand\@gls@checkescquote{%
4922 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4923 ####3\noexpand\null}%
4924 \noexpand\fi
4925 \noexpand\fi
4926 \noexpand\@gls@checkescquote
4927 }%
4928 }%
4929 \@gls@docheckquotedef
4930 }
4931 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
4932 {\string\GlsSetQuote\space not permitted here}%
4933 {Move \string\GlsSetQuote\space earlier in the preamble, as
4934 soon as possible after glossaries.sty has been loaded}}
4935 \fi

```

ramakeindexopts

```

4936 \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`

```

4937 \newcommand{\noist}{%
  Update attributes list
4938 \@gls@addpredefinedattributes
4939 \let\writeist\relax
4940 }

```

`\@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

```
4941 \newcommand*{\@makeglossary}[1]{%
4942   \ifglossaryexists{#1}%
4943   {%
```

Only create a new write if savewrites=false otherwise create a token to collect the information.

```
4944   \ifglssavewrites
4945     \expandafter\newtoks\csname glo@#1@filetok\endcsname
4946   \else
4947     \expandafter\newwrite\csname glo@#1@file\endcsname
4948     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4949   \fi
4950   \@gls@renewglossary
4951   \writeist
4952 }%
4953 {%
4954   \PackageError{glossaries}%
4955   {Glossary type ‘#1’ not defined}%
4956   {New glossaries must be defined before using \string\makeglossary}%
4957 }%
4958 }
```

\@glsopenfile Open write file associated with the given glossary.

```
4959 \newcommand*{\@glsopenfile}[2]{%
4960   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4961   \PackageInfo{glossaries}{Writing glossary file
4962     \jobname.\csname @glotype@#2@out\endcsname}%
4963 }
```

\@closegls

```
4964 \newcommand*{\@closegls}[1]{%
4965   \closeout\csname glo@#1@file\endcsname
4966 }
```

\@gls@automake

```
4967 \ifglxindy
4968 \newcommand*{\@gls@automake}[1]{%
4969   \ifglossaryexists{#1}
4970   {%
4971     \@closegls{#1}%
4972     \ifdefstring{\glsorder}{letter}%
4973     {\def\@gls@order{-M ord/letorder }}%
4974     {\let\@gls@order\@empty}%
4975     \ifcsundef{@xdy@#1@language}%
4976     {\let\@gls@langmod\@xdy@main@language}%
4977     {\letcs\@gls@langmod{@xdy@#1@language}}%
4978     \edef\@gls@dothiswrite{\noexpand\write18{xindy
4979       -I xindy
```

```

4980      \@gls@order
4981      -L \@gls@langmod\space
4982      -M \@gls@istfilebase\space
4983      -C \@gls@codepage\space
4984      -t \jobname.\csuse{@glotype@#1@log}
4985      -o \jobname.\csuse{@glotype@#1@in}
4986      \jobname.\csuse{@glotype@#1@out}}}%
4987  }%
4988  \@gls@dothiswrite
4989  }%
4990  {%
4991  \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4992  }%
4993 }
4994 \else
4995 \newcommand*{\@gls@automake}[1]{%
4996 \ifglossaryexists{#1}
4997 {%
4998   \@closegls{#1}%
4999   \ifdefstring{\glsorder}{letter}%
5000   {\def\@gls@order{-l }}%
5001   {\let\@gls@order\@empty}%
5002   \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
5003     -s \istfilename\space
5004     -t \jobname.\csuse{@glotype@#1@log}
5005     -o \jobname.\csuse{@glotype@#1@in}
5006     \jobname.\csuse{@glotype@#1@out}}}%
5007   }%
5008   \@gls@dothiswrite
5009   }%
5010   {%
5011   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
5012   }%
5013 }
5014 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```
5015 \newcommand*{\@warn@nomakeglossaries}{}
```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```
5016 \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`

```
5017 \newcommand*{\makeglossaries}{%
```

Define the write used for style file also used for all other output files if `savewrites=true`.



```
5018 \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.

```
5019 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}%
5020 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}%}
```

If \@@gls@extramakeindexopts has been defined, write it:

```
5021 \ifundef\@@gls@extramakeindexopts
5022 {}%
5023 {%
5024 \protected@write\@auxout{}{\string\providecommand
5025 \string\@gls@extramakeindexopts[1]{}%
5026 \protected@write\@auxout{}{\string\@gls@extramakeindexopts
5027 {\@gls@extramakeindexopts}}%
5028 }%
```

Write the name of the style file to the aux file (needed by makeglossaries)

```
5029 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
5030 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}%
```

Iterate through each glossary type and activate it.

```
5031 \@for\@glo@type:=\@glo@types\do{%
5032 \ifthenelse{equal{\@glo@type}{}}{ }{%
5033 \makeglossary{\@glo@type}}%
5034 }%
```

New glossaries must be created before \makeglossaries so disable \newglossary.

```
5035 \renewcommand*\newglossary[4][]{%
5036 \PackageError{glossaries}{New glossaries
5037 must be created before \makeglossaries}{You need
5038 to move \string\makeglossaries\space after all your
5039 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
5040 \let\@makeglossary\relax
5041 \let\makeglossary\relax
5042 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
5043 \@disable@onlypremakeg
```

Allow see key:

```
5044 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
5045 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
5046 \def\warn@noprntglossary{%
5047 \ifdefstring{\@glo@types}{,}%
5048 {%
5049 \GlossariesWarningNoLine{No glossaries have been defined}}%
```

```

5050 }%
5051 {%
5052     \GlossariesWarningNoLine{No \string\printglossary\space
5053     or \string\printglossaries\space
5054     found. ^^J(Remove \string\makeglossaries\space if you
5055     don't want any glossaries.) ^^JThis document will not
5056     have a glossary}%
5057 }%
5058 }%

```

Declare list parser for \glsdisplaynumberlist

```

5059 \ifglssavenumberlist
5060     \edef\@gls@dodolistparser{\noexpand\DeclareListParser
5061     {\noexpand\glsnumlistparser}{\delimN}}}%
5062 \@gls@dodolistparser
5063 \fi

```

Prevent user from also using \makenoidxglossaries

```

5064 \let\makenoidxglossaries\@no@makeglossaries

```

Prohibit sort key in printgloss family:

```

5065 \renewcommand*{\@printgloss@setsort}{%
5066     \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5067 }%

```

Check the automake setting:

```

5068 \ifglsautomake
5069     \renewcommand*{\@gls@doautomake}{%
5070         \@for\@gls@type:=\@glo@types\do{%
5071             \ifdefempty{\@gls@type}{}%
5072             {\@gls@automake{\@gls@type}}}%
5073         }%
5074     }%
5075 \fi

```

Check the sort setting:

```

5076 \@glo@check@sortallowed\makeglossaries
5077 }

```

Must occur in the preamble:

```

5078 \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`

```

5079 \let\makeglossary\makeglossaries

```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
5080 \AtEndDocument{%
5081   \warn@nomakeglossaries
5082   \warn@noprintglossary
5083 }
```

`noidxglossaries` Analogous to `\makeglossaries` this activates the commands needed for `\printnoidxglossary`

```
5084 \newcommand*{\makenoidxglossaries}{%
```

Redefine empty glossary warning:

```
5085   \renewcommand{\@gls@noref@warn}[1]{%
5086     \GlossariesWarning{Empty glossary for
5087       \string\printnoidxglossary[type={##1}].
5088     Rerun may be required (or you may have forgotten to use
5089     commands like \string\gls)}%
5090   }%
```

Don't escape makeindex/xindy characters

```
5091   \let\@gls@checkmkidxchars\@gobble
```

Write glossary information to aux instead of glossary files

```
5092   \let\@do@wrglossary\gls@noidxglossary
```

Switch on group headings that use the character code:

```
5093   \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle
```

Allow see key:

```
5094   \let\gls@checkseeallowed\relax
```

Redefine cross-referencing macro:

```
5095   \renewcommand{\@do@seeglossary}[2]{%
5096     \edef\@gls@label{\glsdetoklabel{##1}}%
5097     \protected@write\@auxout{}{%
5098       \string\@gls@reference
5099       {\csname glo@\@gls@label @type\endcsname}%
5100       {\@gls@label}%
5101       {%
5102         \string\glsseeformat##2}%
5103       }%
5104     }%
5105   }%
```

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.

```
5106   \AtBeginDocument
5107   {%
5108     \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5109   }%
```

Change warning about no glossaries

```
5110 \def\warn@noprintglossary{%
5111   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5112     or \string\printnoidxglossaries ^^J
5113     found. (Remove \string\makenoidxglossaries\space if you
5114     don't want any glossaries.)^^JThis document will not have a glossary}%
5115 }%
```

Suppress warning about no \makeglossaries

```
5116 \let\warn@nomakeglossaries\relax
```

Prevent user from also using \makeglossaries

```
5117 \let\makeglossaries\@no@makeglossaries
```

Allow sort key in printgloss family:

```
5118 \renewcommand*{\@printgloss@setsort}{%
5119   \let\@glo@assign@sortkey\@glo@assign@sortkey
```

Initialise default sort order:

```
5120   \def\@glo@sorttype{\@glo@default@sorttype}%
5121 }%
```

All entries must be defined in the preamble:

```
5122 \renewcommand*\new@glossaryentry[2]{%
5123   \PackageError{glossaries}{Glossary entries must be
5124     defined in the preamble^^Jwhen you use
5125     \string\makenoidxglossaries}%
5126   {Either move your definitions to the preamble or use
5127     \string\makeglossaries}%
5128 }%
```

Redefine \glsentrynumberlist

```
5129 \renewcommand*{\glsentrynumberlist}[1]{%
5130   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5131   \ifdef\@gls@loclist
5132     {%
5133       \glsnoidxloclist{\@gls@loclist}%
5134     }%
5135     {%
5136       ??\glsdoifexists{##1}%
5137       {%
5138         \GlossariesWarning{Missing location list for '##1'. Either
5139           a rerun is required or you haven't referenced the entry}%
5140       }%
5141     }%
5142 }%
```

Redefine \glsdisplaynumberlist

```
5143 \renewcommand*{\glsdisplaynumberlist}[1]{%
5144   \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5145   \ifdef\@gls@loclist
5146     {%
```

```

5147 \def\@gls@noidxloclist@sep{%
5148 \def\@gls@noidxloclist@sep{%
5149 \def\@gls@noidxloclist@sep{%
5150 \glsnumlistsep
5151 }%
5152 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5153 }%
5154 }%
5155 \def\@gls@noidxloclist@finalsep{}%
5156 \def\@gls@noidxloclist@prev{}%
5157 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5158 \@gls@noidxloclist@finalsep
5159 \@gls@noidxloclist@prev
5160 }%
5161 {%
5162 ??\glsdoifexists{##1}%
5163 {%
5164 \GlossariesWarning{Missing location list for ‘##1’. Either
5165 a rerun is required or you haven’t referenced the entry}%
5166 }%
5167 }%
5168 }%

```

Provide a generic way of iterating through the number list:

```

5169 \renewcommand*\glsnumberlistloop}[3]{%
5170 \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5171 \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
5172 \let\@gls@org@glsseeformat\glsseeformat
5173 \let\glsnoidxdisplayloc##2\relax
5174 \let\glsseeformat##3\relax
5175 \ifdef\@gls@loclist
5176 {%
5177 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5178 }%
5179 {%
5180 ??\glsdoifexists{##1}%
5181 {%
5182 \GlossariesWarning{Missing location list for ‘##1’. Either
5183 a rerun is required or you haven’t referenced the entry}%
5184 }%
5185 }%
5186 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
5187 \let\glsseeformat\@gls@org@glsseeformat
5188 }%

```

Modify sanitize sort function

```

5189 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5190 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5191 \@gls@noidx@setsanitizesort

```

Check sort option allowed.

```

5192 \@glo@check@sortallowed\makenoidxglossaries
5193 }

```

Preamble-only command:

```

5194 \@onlypreamble{\makenoidxglossaries}

```

```

\lsnumberlistloop \glsnumberlistloop{<label>}{<handler>}

```

```

5195 \newcommand*{\glsnumberlistloop}[2]{%
5196   \PackageError{glossaries}{\string\glsnumberlistloop\space
5197     only works with \string\makenoidxglossaries}{}%
5198 }

```

`\listloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc {<prefix>}{<counter>}{<format>}{<n>}`)

```

5199 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5200   #1%
5201 }

```

`@makeglossaries` Can't use both `\makeglossaries` and `\makenoidxglossaries`

```

5202 \newcommand*{\@no@makeglossaries}{%
5203   \PackageError{glossaries}{You can't use both
5204     \string\makeglossaries\space and \string\makenoidxglossaries}%
5205   {Either use one or other (or none) of those commands but not both
5206     together.}%
5207 }

```

`@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```

5208 \newcommand{\@gls@noref@warn}[1]{%
5209   \GlossariesWarning{\string\makenoidxglossaries\space
5210     is required to make \string\printnoidxglossary[type={#1}] work}%
5211 }

```

`s@noidxglossary` Write the glossary information to the aux file:

```

5212 \newcommand*{\gls@noidxglossary}{%
5213   \protected@write\@auxout{}{%
5214     \string\@gls@reference
5215       {\csname glo@\@gls@label @type\endcsname}%
5216       {\@gls@label}%
5217       {\string\glsnoidxdisplayloc
5218         {\@glo@counterprefix}%
5219         {\@gls@counter}%
5220         {\@glsnumberformat}%
5221         {\@glslocref}%
5222       }%
5223   }%
5224 }

```

## 1.14 Writing information to associated files

`\istfile`   Deprecated.

```
5225 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if `savewrites=true`.

```
5226 \AtEndDocument{%
```

```
5227   \glswritefiles
```

```
5228 }
```

`\@glswritefiles`   Only write the files if `savewrites=true`

```
5229 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
5230   \forallglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
5231     \ifcsundef{glo@\@glo@type @filetok}%
```

```
5232     {%
```

```
5233       \def\gls@tmp{}%
```

```
5234     }%
```

```
5235     {%
```

```
5236       \edef\gls@tmp{\expandafter\the
```

```
5237         \csname glo@\@glo@type @filetok\endcsname}%
```

```
5238     }%
```

```
5239     \ifx\gls@tmp\@empty
```

```
5240       \ifx\@glo@type\glsdefaulttype
```

```
5241         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
5242           entries.^^JRemember to use package option ‘nomain’ if
```

```
5243 you
```

```
5244           don’t want to^^Juse the main glossary}%
```

```
5245       \else
```

```
5246         \GlossariesWarningNoLine{Glossary ‘\@glo@type’ has no
```

```
5247           entries}%
```

```
5248       \fi
```

```
5249     \else
```

```
5250       \@glsopenfile{\glswrite}{\@glo@type}%
```

```
5251       \immediate\write\glswrite{%
```

```
5252         \expandafter\the
```

```
5253         \csname glo@\@glo@type @filetok\endcsname}%
```

```
5254       \immediate\closeout\glswrite
```

```
5255     \fi
```

```
5256   }%
```

```
5257 }
```

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`

```
5258 \if@gls@docloaded
5259 \else
5260   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
5261 \fi
```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
5262 \newcommand*{\gls@glossary}[1]{%
5263   \@gls@glossary{#1}%
5264 }
```

`\@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.

```
5265 \newcommand*{\@gls@glossary}[2]{%
5266   \if@gls@debug
5267     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5268   \fi
5269   \index{#2}%
5270 }
```

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`

```
5271 \newcommand{\@gls@renewglossary}{%
5272   \gdef\@gls@glossary##1{\@bsphack\beginingroup\gls@wrglossary{##1}}%
5273   \let\@gls@renewglossary\empty
5274 }
```

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`

```
5275 \newcommand*{\gls@wrglossary}[2]{%
5276   \ifglssavewrites
5277     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5278     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
```



```

5279     \expandafter{\@gls@tmp^^J}%
5280 \else

5281     \ifcsdef{glo@#1@file}%
5282     {%
5283         \expandafter\protected@write\csname glo@#1@file\endcsname{%
5284             \gls@disablepagerefexpansion}{#2}%
5285     }%
5286     {%
5287         \ifignoredglossary{#1}{}%
5288         {%
5289             \GlossariesWarning{No file defined for glossary ‘#1’}%
5290         }%
5291     }%
5292 \fi
5293 \endgroup\@esphack
5294 }

```

\do@wrglossary

```

5295 \newcommand*{\do@wrglossary}[1]{%
5296     \glswriteentry{#1}{\do@wrglossary{#1}}%
5297 }

```

\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.

```

5298 \newcommand*{\glswriteentry}[2]{%
5299     \ifglsindexonlyfirst
5300     \ifglsused{#1}{#2}%
5301 \else
5302     #2%
5303 \fi
5304 }

```

protected@pagefmts List of page formats to be protected against expansion.

```

5305 \newcommand{\gls@protected@pagefmts}{%
5306     \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5307 }

```

pagerefexpansion

```

5308 \newcommand*{\gls@disablepagerefexpansion}{%
5309     \@for\@gls@this:=\gls@protected@pagefmts\do
5310     {%
5311         \expandafter\let\@gls@this\relax
5312     }%
5313 }

```

\gls@alphpage

```

5314 \newcommand*{\gls@alphpage}{\@alph\c@page}

```

```

\gls@Alphpage
5315 \newcommand*{\gls@Alphpage}{\@Alph\c@page}

\gls@numberpage
5316 \newcommand*{\gls@numberpage}{\number\c@page}

\gls@arabicpage
5317 \newcommand*{\gls@arabicpage}{\@arabic\c@page}

\gls@romanpage
5318 \newcommand*{\gls@romanpage}{\romannumeral\c@page}

\gls@Romanpage
5319 \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).

5320 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5321   \eappto\gls@protected@pagefmts{\expandonce{\csname gls#1page\endcsname}}%
5322   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5323   \eappto\@wrglossarynumberhook{%
5324     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5325     \expandonce{\csname#1\endcsname}%
5326     \noexpand\def\expandonce{\csname#1\endcsname}{%
5327       \noexpand\@wrglossary@pageformat
5328       \expandonce{\csname gls#1page\endcsname}%
5329       \expandonce{\csname org@gls#1\endcsname}%
5330     }%
5331   }%
5332 }

ssarynumberhook Hook used by \@@do@wrglossary
5333 \newcommand*\@wrglossarynumberhook{}

sary@pageformat
5334 \newcommand{\@wrglossary@pageformat}[3]{%
5335   \ifx#3\c@page #1\else #2#3\fi
5336 }

owprimitivemods Conditional to determine whether or not \@@do@wrglossary should be allowed to temporar-
ily redefine \the and \number.
5337 \newif\ifglswrallowprimitivemods
5338 \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.

```
5339 \newcommand*{\@@do@wrglossary}[1]{%
5340   \begingroup
```

First a bit of hackery to prevent premature expansion of \c@page. Store original definitions:

```
5341   \let\orgthe\the
5342   \let\orgnumber\number

5343   \let\orgarabic\@arabic
5344   \let\orgromannumeral\romannumeral
5345   \let\orgalph\@alph
5346   \let\orgAlph\@Alph
5347   \let\orgRoman\@Roman
```

Redefine:

```
5348   \ifglswrallowprimitivemods
5349     \def\the##1{%
5350       \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5351     \def\number##1{%
5352       \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5353     \fi
5354     \def\@arabic##1{%
5355       \ifx##1\c@page \gls@arabicpage\else\orgarabic##1\fi}%
5356     \def\romannumeral##1{%
5357       \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5358     \def\@Roman##1{%
5359       \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5360     \def\@alph##1{%
5361       \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5362     \def\@Alph##1{%
5363       \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```
5364   \@wrglossarynumberhook
```

Prevent expansion:

```
5365   \gls@disablepagerefexpansion
```

Now store location in \@glslocref:

```
5366   \protected@xdef\@glslocref{\theHglentrycounter}%
5367   \endgroup
```

Escape any special characters

```
5368   \@gls@checkmkidxchars\@glslocref
```

Check if the hyper-location is the same as the location and set the hyper prefix.

```
5369   \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5370   \def\@glo@counterprefix{}%
5371   \else
5372     \protected@edef\@glsHlocref{\theHglentrycounter}%
5373     \@gls@checkmkidxchars\@glsHlocref
```

```

5374 \edef\@do@glS@getcounterprefix{\noexpand\@glS@getcounterprefix
5375 {\@glS@locref}{\@glS@Hlocref}%
5376 }%
5377 \@do@glS@getcounterprefix
5378 \fi

```

De-tok label if required

```

5379 \edef\@glS@label{\glSdetoklabel{#1}}%

```

Write the information to file:

```

5380 \@do@wrglossary
5381 }

```

@do@wrglossary

```

5382 \newcommand*{\@do@wrglossary}{%

```

Determine whether to use xindy or makeindex syntax

```

5383 \ifglSxindy

```

Need to determine if the formatting information starts with a ( or ) indicating a range.

```

5384 \expandafter\@glo@check@mkidxrangechar\@glS@numberformat\@nil
5385 \def\@glo@range{}%
5386 \expandafter\if\@glo@prefix(\relax
5387 \def\@glo@range{:open-range}%
5388 \else
5389 \expandafter\if\@glo@prefix)\relax
5390 \def\@glo@range{:close-range}%
5391 \fi
5392 \fi

```

Write to the glossary file using xindy syntax.

```

5393 \glS@glossary{\csname glo@\@glS@label @type\endcsname}{%
5394 (indexentry :tkey (\csname glo@\@glS@label @index\endcsname)

5395 :locref \string"{\@glo@counterprefix}{\@glS@locref}\string" %
5396 :attr \string"\@glS@counter\@glo@suffix\string"
5397 \@glo@range
5398 )
5399 }%
5400 \else

```

Convert the format information into the format required for makeindex

```

5401 \@set@glo@numformat{\@glo@numfmt}{\@glS@counter}{\@glS@numberformat}%
5402 {\@glo@counterprefix}%

```

Write to the glossary file using makeindex syntax.

```

5403 \glS@glossary{\csname glo@\@glS@label @type\endcsname}{%
5404 \string\glossaryentry{\csname glo@\@glS@label @index\endcsname
5405 \@glS@encapchar\@glo@numfmt}{\@glS@locref}}%
5406 \fi
5407 }

```

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.)

```

5408 \newcommand*\@gls@getcounterprefix[2]{%
5409   \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5410   \ifx\@gls@thisloc\@gls@thisHloc
5411     \def\@glo@counterprefix{}%
5412   \else
5413     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5414       \def\@glo@tmp{##2}%
5415       \ifx\@glo@tmp\@empty
5416         \def\@glo@counterprefix{}%
5417       \else
5418         \def\@glo@counterprefix{##1}%
5419       \fi
5420     }%
5421     \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed.

```

5422   \ifx\@glo@counterprefix\@empty
5423     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5424       prefixing^^Jlocation ‘#1’. You need to modify the
5425       definition of \string\theH\@gls@counter^^Jotherwise you
5426       will get the warning: “name{\@gls@counter.#1}’ has been^^J
5427       referenced but does not exist"%
5428   \fi
5429 \fi
5430 }

```

## 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.

```

5431 \newcommand{\@do@seeglossary}[2]{%
5432 \def\@gls@xref{#2}%
5433 \@onelevel@sanitize\@gls@xref
5434 \@gls@checkmkidxchars\@gls@xref
5435 \ifglxindy
5436   \gls@glossary{\csname glo@#1@type\endcsname}{%
5437     (indexentry
5438       :tkey (\csname glo@#1@index\endcsname)
5439       :xref (\string"\@gls@xref\string")
5440       :attr \string"see\string"
5441     )
5442   }%

```

```

5443 \else
5444   \gls@glossary{\csname glo@#1@type\endcsname}{%
5445     \string\glossaryentry{\csname glo@#1@index\endcsname
5446       \gls@encapchar glsseeformat\@gls@xref}{Z}}}%
5447 \fi
5448 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

5449 \def\@gls@fixbraces#1#2#3\@nil{%
5450   \ifx#2[\relax
5451     \@gls@fixbraces#1#2#3\@end@fixbraces
5452   \else
5453     \def#1{{#2#3}}%
5454   \fi
5455 }

```

`@@gls@fixbraces`

```

5456 \def@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5457   \def#1{[#2]{#3}}%
5458 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

5459 \DeclareRobustCommand*\glssee[3][\seename]{%
5460   \@do@seeglossary{#2}{#1}{#3}}
5461 \newcommand*\@glssee[3][\seename]{%
5462   \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.

```

5463 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5464   \emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

5465 \DeclareRobustCommand*\glsseelist[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

5466   \let\@gls@dolast\relax

```

Don’t display separator on the first iteration of the loop

```

5467   \let\@gls@donext\relax

```

Iterate through the labels

```

5468   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

5469     \ifx\@xfor@nextelement\@nnil
5470       \@gls@dolast
5471     \else
5472       \@gls@donext
5473     \fi

```

Display the entry for this label. (Expanding label as it's a temporary control sequence that's used elsewhere.)

```
5474 \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5475 \let\@gls@dolast\glsseelastsep
```

```
5476 \let\@gls@donext\glsseesep
```

```
5477 }%
```

```
5478 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5479 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
5480 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5481 \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.)

```
5482 \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.

```
5483 \newcommand*{\gls@save@numberlist}[1]{%
```

```
5484 \ifglssavenumberlist
```

```
5485 \toks@{#1}%
```

```
5486 \edef\@do@writeaux@info{%
```

```
5487 \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
```

```
5488 }%
```

```
5489 \@onelevel@sanitize\@do@writeaux@info
```

```
5490 \protected@write\@auxout{}\@do@writeaux@info%
```

```
5491 \fi
```

```
5492 }
```

`\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.)

```
5493 \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.

```
5494 \ifcsundef{printglossary}{}%  
5495 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5496 \@gls@warnonglossdefined  
5497 \undef\printglossary  
5498 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5499 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%  
5500 \@printglossary{#1}{\@print@glossary}%  
5501 }
```

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`

```
5502 \newcommand*{\printglossaries}{%  
5503 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%  
5504 }
```

`ntnoidxglossary` 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.

```
5505 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%  
5506 \@printglossary{#1}{\@print@noidx@glossary}%  
5507 }
```

`noidxglossaries` Analogous to `\printglossaries`

```
5508 \newcommand*{\printnoidxglossaries}{%  
5509 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%  
5510 }
```

`ntgloss@setsort` Initialise to do nothing.

```
5511 \newcommand*{\@printgloss@setsort}{}%
```

`preglossaryhook`

```
5512 \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.

```
5513 \newcommand{\@printglossary}[2]{%
```

Set up defaults.

```
5514 \def\@glo@type{\glsdefaulttype}%
```

```
5515 \def\glossarytitle{\csname @glo@type\@glo@type @title\endcsname}%
```

```
5516 \def\glossarytoctitle{\glossarytitle}%
```

```
5517 \let\org@glossarytitle\glossarytitle
```

```
5518 \def\@glossarystyle{%
```

```
5519 \ifx\@glossary@default@style\relax
```

```
5520 \GlossariesWarning{No default glossary style provided \MessageBreak
```

```
5521 for the glossary ‘\@glo@type’. \MessageBreak
```

```
5522 Using deprecated fallback. \MessageBreak
```

```
5523 To fix this set the style with \MessageBreak
```

```
5524 \string\setglossarystyle\space or use the \MessageBreak
```

```
5525 style key=value option}%
```

```
5526 \fi
```

```
5527 }%
```

```
5528 \def\gls@dotoc@title{\gls@settoc@title{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
5529 \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
5530 \bgroup
```

Activate or deactivate sort key:

```
5531 \@printgloss@setsort
```

Determine settings specified in the optional argument.

```
5532 \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)

```
5533 \ifx\glossarytitle\org@glossarytitle
```

```
5534 \else
```

```
5535 \expandafter\let\csname @glo@type\@glo@type @title\endcsname
```

```
5536 \glossarytitle
```

```
5537 \fi
```

Allow a high-level user command to indicate the current glossary

```
5538 \let\currentglossary\@glo@type
```

Enable individual number lists to be suppressed.

```
5539 \let\org@glossaryentrynumbers\glossaryentrynumbers
```

```
5540 \let\glsnonextpages\glsnonextpages
```

Enable individual number list to be activated:

```
5541 \let\glsnextpages\@glsnextpages
```

Enable suppression of description terminators.

```
5542 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
5543 \gls@dotocitle
```

Set the glossary style

```
5544 \@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):

```
5545 \let\gls@org@glossaryentryfield\glossentry
5546 \let\gls@org@glossarysubentryfield\subglossentry
5547 \renewcommand{\glossentry}[1]{%
5548   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5549   \gls@org@glossaryentryfield{##1}%
5550 }%
5551 \renewcommand{\subglossentry}[2]{%
5552   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5553   \gls@org@glossarysubentryfield{##1}{##2}%
5554 }%
```

```
5555 \@gls@preglossaryhook
```

Now do the handler macro that deals with the actual glossary:

```
5556 #2%
```

End the current scope

```
5557 \egroup
```

Reset `\glossaryentrynumbers`

```
5558 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
```

Suppress warning about no `\printglossary`

```
5559 \global\let\warn@noprintglossary\relax
5560 }
```

`@print@glossary` Internal workings of `\printglossary` dealing with reading the external file.

```
5561 \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.)

```
5562 \makeatletter
```

Input the glossary file, if it exists.

```
5563 \@input@{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do \null. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
5564 \IfFileExists{\jobname.\csname @glo@type\@glo@type @in\endcsname}%
5565 {}%
5566 {\null}%
```

If xindy is being used, need to write the language dependent information to the .aux file for makeglossaries.

```
5567 \ifglxindy
5568 \ifcsundef{@xdy@\@glo@type @language}%
5569 {%
5570 \edef\@do@auxoutstuff{%
5571 \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.

```
5572 \noexpand\immediate\noexpand\write\@auxout{%
5573 \string\providecommand\string\@xdylanguage[2]{}}%
5574 \noexpand\immediate\noexpand\write\@auxout{%
5575 \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5576 }%
5577 }%
5578 }%
5579 {%
5580 \edef\@do@auxoutstuff{%
5581 \noexpand\AtEndDocument{%
5582 \noexpand\immediate\noexpand\write\@auxout{%
5583 \string\providecommand\string\@xdylanguage[2]{}}%
5584 \noexpand\immediate\noexpand\write\@auxout{%
5585 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5586 @language\endcsname}}%
5587 }%
5588 }%
5589 }%
5590 \@do@auxoutstuff
5591 \edef\@do@auxoutstuff{%
5592 \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.

```
5593 \noexpand\immediate\noexpand\write\@auxout{%
5594 \string\providecommand\string\@gls@codepage[2]{}}%
5595 \noexpand\immediate\noexpand\write\@auxout{%
5596 \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5597 }%
5598 }%
5599 \@do@auxoutstuff
5600 \fi
```

Activate warning if \makeglossaries hasn't been used.

```

5601 \renewcommand*{\@warn@nomakeglossaries}{%
5602   \GlossariesWarningNoLine{\string\makeglossaries\space
5603     hasn't been used,^^Jthe glossaries will not be updated}%
5604   }%
5605 }

```

The sort macros all have the syntax:

$$\backslash @glo@sortmacro@ \langle order \rangle \{ \langle type \rangle \}$$

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  $\backslash @glsref@ \langle type \rangle$ . The actual sorting is done by  $\backslash @glo@sortentries \{ \langle handler \rangle \} \{ \langle type \rangle \}$ .

$glo@sortentries$

```

5606 \newcommand*{\@glo@sortentries}[2]{%
5607   \glosortentrieswarning
5608   \def\@glo@sortinglist{}%
5609   \def\@glo@sortinghandler{#1}%
5610   \edef\@glo@type{#2}%
5611   \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
5612   \csdef{\@glsref@#2}{}%
5613   \@for\@this@label:=\@glo@sortinglist\do{%

```

Has this entry already been added?

```

5614   \xifinlistcs{\@this@label}{\@glsref@#2}%
5615   {}%
5616   {%
5617     \listcsxadd{\@glsref@#2}{\@this@label}%
5618     }%
5619     \ifcsdef{\@glo@sortingchildren@\@this@label}%
5620     {%
5621       \@glo@addchildren{#2}{\@this@label}%
5622       }%
5623     {}%
5624   }%
5625 }

```

$@glo@addchildren$

$$\backslash @glo@addchildren \{ \langle type \rangle \} \{ \langle parent \rangle \}$$

```

5626 \newcommand*{\@glo@addchildren}[2]{%

```

Scope to allow nesting.

```

5627   \bgroup
5628     \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
5629     \@for\@this@childlabel:=\@glo@childlist\do
5630     {%

```

Check this label hasn't already been added.

```

5631      \xifinlistcs{\@this@childlabel}{@glsref@#1}%
5632      {%
5633      {%
5634      \listcsxadd{@glsref@#1}{\@this@childlabel}%
5635      }%

```

Does this child have children?

```

5636      \ifcsdef{@glo@sortingchildren@\@this@childlabel}%
5637      {%
5638      \@glo@addchildren{#1}{\@this@childlabel}%
5639      }%
5640      {%
5641      }%
5642      }%
5643 \egroup
5644 }

```

@do@sortentries

```

5645 \newcommand*{\@glo@do@sortentries}[1]{%
5646 \ifglshasparent{#1}%
5647 {%

```

This entry has a parent, so add it to the child list

```

5648 \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
5649 \ifcsundef{@glo@sortingchildren@\@glo@parent}%
5650 {%
5651 \csdef{@glo@sortingchildren@\@glo@parent}{}%
5652 }%
5653 }%
5654 \expandafter\@glo@sortedinsert
5655 \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%

```

Has the parent been added?

```

5656 \xifinlistcs{\@glo@parent}{@glsref@\@glo@type}%
5657 {%

```

Yes, it has so do nothing.

```

5658 }%
5659 {%

```

No, it hasn't so add it now.

```

5660 \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5661 }%
5662 }%
5663 {%
5664 \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5665 }%
5666 }

```

glo@sortedinsert    `\@glo@sortedinsert{<list>}{<entry label>}`

Insert into list.

```
5667 \newcommand*{\@glo@sortedinsert}[2]{%
5668   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5669 }%
```

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

```
5670 \newcommand*{\@glo@sorthandler@word}[2]{%
5671   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5672   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5673   \edef\glo@do@compare{%
5674     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
5675     {\expandonce\@gls@sort@B}%
5676     {\expandonce\@gls@sort@A}%
5677   }%
5678   \glo@do@compare
5679 }
```

thandler@letter

```
5680 \newcommand*{\@glo@sorthandler@letter}[2]{%
5681   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5682   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5683   \edef\glo@do@compare{%
5684     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5685     {\expandonce\@gls@sort@B}%
5686     {\expandonce\@gls@sort@A}%
5687   }%
5688   \glo@do@compare
5689 }
```

orthandler@case    Case-sensitive sort.

```
5690 \newcommand*{\@glo@sorthandler@case}[2]{%
5691   \letcs\@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5692   \letcs\@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5693   \edef\glo@do@compare{%
5694     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5695     {\expandonce\@gls@sort@B}%
5696     {\expandonce\@gls@sort@A}%
5697   }%
5698   \glo@do@compare
5699 }
```

thandler@nocase    Case-insensitive sort.

```

5700 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5701   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5702   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5703   \edef\glo@do@compare{%
5704     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5705     {\expandonce\@gls@sort@B}%
5706     {\expandonce\@gls@sort@A}%
5707   }%
5708   \glo@do@compare
5709 }

```

@sortmacro@word Sort macro for ‘word’

```

5710 \newcommand*{\@glo@sortmacro@word}[1]{%
5711   \ifdefstring{\@glo@default@sorttype}{standard}%
5712   {%
5713     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5714   }%
5715   {%
5716     \PackageError{glossaries}{Conflicting sort options:^^J
5717       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5718       \string\printnoidxglossary[sort=word]}{}%
5719   }%
5720 }

```

ortmacro@letter Sort macro for ‘letter’

```

5721 \newcommand*{\@glo@sortmacro@letter}[1]{%
5722   \ifdefstring{\@glo@default@sorttype}{standard}%
5723   {%
5724     \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5725   }%
5726   {%
5727     \PackageError{glossaries}{Conflicting sort options:^^J
5728       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5729       \string\printnoidxglossary[sort=letter]}{}%
5730   }%
5731 }

```

tmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5732 \newcommand*{\@glo@sortmacro@standard}[1]{%
5733   \ifdefstring{\@glo@default@sorttype}{standard}%
5734   {%
5735     \ifcsdef{\@glo@sorthandler@\glsorder}%
5736     {%
5737       \@glo@sortentries{\csuse{\@glo@sorthandler@\glsorder}}{#1}%
5738     }%
5739     {%
5740       \PackageError{glossaries}{Unknown sort handler ‘\glsorder’}{}%
5741     }%
5742   }%

```

```

5743  {%
5744    \PackageError{glossaries}{Conflicting sort options:^^J
5745      \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5746      \string\printnoidxglossary[sort=standard]}{}%
5747  }%
5748 }

```

@sortmacro@case Sort macro for ‘case’

```

5749 \newcommand*{\@glo@sortmacro@case}[1]{%
5750   \ifdefstring{\@glo@default@sorttype}{standard}%
5751   {%
5752     \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5753   }%
5754   {%
5755     \PackageError{glossaries}{Conflicting sort options:^^J
5756       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5757       \string\printnoidxglossary[sort=case]}{}%
5758   }%
5759 }

```

@sortmacro@nocase Sort macro for ‘nocase’

```

5760 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5761   \ifdefstring{\@glo@default@sorttype}{standard}%
5762   {%
5763     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5764   }%
5765   {%
5766     \PackageError{glossaries}{Conflicting sort options:^^J
5767       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5768       \string\printnoidxglossary[sort=nocase]}{}%
5769   }%
5770 }

```

@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@<type>.

```

5771 \newcommand*{\@glo@sortmacro@def}[1]{%
5772   \def\@glo@sortinglist{%
5773     \for@gl@sentries[#1]{\@gl@thislabel}%
5774     {%
5775       \xifinlistcs{\@gl@thislabel}{\@gl@ref@#1}%
5776       {%
5777         \list@add{\@glo@sortinglist}{\@gl@thislabel}%
5778       }%
5779     }%

```

Hasn't been referenced.

```

5780   }%
5781 }%
5782 \cslet{\@gl@ref@#1}{\@glo@sortinglist}%
5783 }

```



ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5784 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5785   \ifinlistcs{#1}{\@glsref@\@glo@type}%
5786   {%
5787     {%
5788       \listcsadd{\@glsref@\@glo@type}{#1}%
5789     }%
5790   \ifcsdef{\@glo@sortingchildren@#1}%
5791     {%
5792       \@glo@addchildren{\@glo@type}{#1}%
5793     }%
5794   }%
5795 }
```

o@sortmacro@use Sort macro for 'use'. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
5796 \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.

```
5797 \newcommand*{\@print@noidx@glossary}{%
5798   \ifcsdef{\@glsref@\@glo@type}%
5799   {%
```

Sort the entries:

```
5800   \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
5801   {%
5802     \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
5803   }%
5804   {%
5805     \PackageError{glossaries}{Unknown sort handler '\@glo@sorttype'}{ }%
5806   }%
```

Do the glossary heading and preamble

```
5807   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5808   \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.

```
5809   \def\@gls@currentlettergroup{%
5810     \begin{theglossary}%
5811     \glossaryheader
5812     \glsresetentrylist
```

Iterate through the entries.

```
5813 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}}%
```

Finally end the glossary and do the postamble:

```
5814 \end{theglossary}%
5815 \glossarypostamble
5816 }%
5817 {%
5818 \@gls@noref@warn{\@glo@type}%
5819 }%
5820 }
```

\glo@grabfirst

```
5821 \def\glo@grabfirst#1#2\@nil{%
5822 \def\@gls@firsttok{#1}%
5823 \ifdefempty\@gls@firsttok
5824 {%
5825 \def\@glo@thislettergrp{0}%
5826 }%
5827 {%
```

Sanitize it:

```
5828 \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
5829 \expandafter\@glo@grabfirst\@gls@firsttok{}{}\@nil
5830 }%
5831 }
```

\@glo@grabfirst

```
5832 \def\@glo@grabfirst#1#2\@nil{%
5833 \ifdefempty\@glo@thislettergrp
5834 {%
5835 \def\@glo@thislettergrp{glssymbols}%
5836 }%
5837 {%
5838 \count@=\uccode'#1\relax
5839 \ifnum\count@=0\relax
5840 \def\@glo@thislettergrp{glssymbols}%
5841 \else
5842 \ifdefstring\@glo@sorttype{case}%
5843 {%
5844 \count@='#1\relax
5845 }%
5846 {%
5847 }%
5848 \edef\@glo@thislettergrp{\the\count@}%
5849 \fi
5850 }%
5851 }
```

\@gls@noidx@do    Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label.  
This only allows one sublevel.

```
5852 \newcommand{\@gls@noidx@do}[1]{%
```

    Get this entry's location list

```
5853   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
```

    Does this entry have a parent?

```
5854   \ifglshasparent{#1}%
5855   {%
```

        Has a parent.

```
5856   \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
5857   \ifdefvoid{\@gls@loclist}
5858   {%
5859   \subglossentry{\gls@level}{#1}{}%
5860   }%
5861   {%
5862   \subglossentry{\gls@level}{#1}%
5863   {%
5864   \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5865   }%
5866   }%
5867   }%
5868   {%
```

        Doesn't have a parent Get this entry's sort key

```
5869   \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
```

        Fetch the first letter:

```
5870   \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5871   \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5872   {}%
5873   {%
```

        Do the group header:

```
5874   \ifdefempty{\@gls@currentlettergroup}{}%
5875   {%
```

        The group skip may start a new scope, so make a global assignment.

```
5876   \global\let\@glo@thislettergrp\@glo@thislettergrp
5877   \glsgroupskip
5878   }%
5879   \glsgroupheading{\@glo@thislettergrp}%
5880   }%

5881   \global\let\@gls@currentlettergroup\@glo@thislettergrp
```

        Do this entry:

```
5882   \ifdefvoid{\@gls@loclist}
5883   {%
5884   \glossentry{#1}{}%
```

```

5885 }%
5886 {%
5887   \glossentry{#1}%
5888   {%
5889     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5890   }%
5891 }%
5892 }%
5893 }

```

`\glsnoidxloclist` `\glsnoidxloclist{<list cs>}`

Display location list.

```

5894 \newcommand*{\glsnoidxloclist}[1]{%
5895   \def\@gls@noidxloclist@sep{}%
5896   \def\@gls@noidxloclist@prev{}%
5897   \forlistloop{\glsnoidxloclisthandler}{#1}%
5898 }

```

`loclisthandler` Handler for location list iterator.

```

5899 \newcommand*{\glsnoidxloclisthandler}[1]{%
5900   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5901   {%

```

Same as previous location so skip.

```

5902 }%
5903 {%
5904   \@gls@noidxloclist@sep
5905   #1%
5906   \def\@gls@noidxloclist@sep{\delimN}%
5907   \def\@gls@noidxloclist@prev{#1}%
5908 }%
5909 }

```

`displayloclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```

5910 \newcommand*{\glsnoidxdisplayloclisthandler}[1]{%
5911   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5912   {%

```

Same as previous location so skip.

```

5913 }%
5914 {%
5915   \@gls@noidxloclist@sep
5916   \@gls@noidxloclist@prev
5917   \def\@gls@noidxloclist@prev{#1}%
5918 }%
5919 }

```

`\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```
5920 \newcommand*\glsnoidxdisplayloc[4]{%
5921   \setentrycounter[#1]{#2}%
5922   \csuse{#3}{#4}%
5923 }
```

`\@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.

```
5924 \newcommand*\@gls@reference}[3]{%
```

Add to label list

```
5925   \glsdoifexistsorwarn{#2}%
5926   {%
5927     \ifcsundef{@glsref@#1}{\csgdef{@glsref@#1}{}}{}%
5928     \ifinlistcs{#2}{@glsref@#1}%
5929     {}%
5930     {\listcsgadd{@glsref@#1}{#2}}%
5931   }
```

Add to location list

```
5931   \ifcsundef{glo@\glsdetoklabel{#2}@loclist}%
5932   {\csgdef{glo@\glsdetoklabel{#2}@loclist}{}}{}%
5933   {}%
5934   \listcsgadd{glo@\glsdetoklabel{#2}@loclist}{#3}%
5935   }%
5936 }
```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```
5937 \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`.

```
5938 \define@key{printgloss}{title}{%
5939   \def\glossarytitle{#1}%
5940   \let\gls@dotoc@title\relax
5941 }
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
5942 \define@key{printgloss}{toctitle}{%
5943   \def\glossarytoctitle{#1}%
5944   \let\gls@dotoc@title\relax
5945 }
```

The style key sets the glossary style (but only for the given glossary).

```

5946 \define@key{printgloss}{style}{%
5947   \ifcsundef{@glsstyle@#1}%
5948   {%
5949     \PackageError{glossaries}%
5950     {Glossary style ‘#1’ undefined}{}%
5951   }%
5952   {%
5953     \def\@glossarystyle{\setglossentrycompatibility
5954       \csname @glsstyle@#1\endcsname}%
5955   }%
5956 }

```

The numberedsection key determines if this glossary should be in a numbered section.

```

5957 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5958 false,nolabel,autolabel,nameref}[nolabel]{%
5959   \ifcase\nr\relax
5960     \renewcommand*{\@glossarysecstar}{*}%
5961     \renewcommand*{\@glossaryseclabel}{}%
5962   \or
5963     \renewcommand*{\@glossarysecstar}{}%
5964     \renewcommand*{\@glossaryseclabel}{}%
5965   \or
5966     \renewcommand*{\@glossarysecstar}{}%
5967     \renewcommand*{\@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
5968   \or
5969     \renewcommand*{\@glossarysecstar}{*}%
5970     \renewcommand*{\@glossaryseclabel}{%
5971       \protected@edef\@currentlabelname{\glossarytoctitle}%
5972       \label{\glsautoprefix\@glo@type}}%
5973   \fi
5974 }

```

The nogroupskip key determines whether or not there should be a vertical gap between glossary groups.

```

5975 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5976   \csuse{glsnogroupskip#1}%
5977 }

```

The nopostdot key has the same effect as the package option of the same name.

```

5978 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5979   \csuse{glsnopostdot#1}%
5980 }

```

The entrycounter key is the same as the package option but localised to the current glossary.

```

5981 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5982   \csuse{glsentrycounter#1}%
5983   \ifglsentrycounter
5984     \ifx\@gls@counterwithin\@empty
5985       \newcounter{glossaryentry}%
5986     \else

```

```

5987     \newcounter{glossaryentry}[\@gls@counterwithin]%
5988     \fi
5989     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5990     \renewcommand*\{glsresetentrycounter}{%
5991       \setcounter{glossaryentry}{0}%
5992     }%
5993     \renewcommand*\{glsstepentry}[1]{%
5994       \refstepcounter{glossaryentry}%
5995       \label{glsentry-\glsdetoklabel{##1}}%
5996     }%
5997     \renewcommand*\{glsentrycounterlabel}{\theglossaryentry.\space}%
5998     \renewcommand*\{glsentryitem}[1]{%
5999       \glsstepentry{##1}\glsentrycounterlabel
6000     }%
6001   \else
6002     \renewcommand*\{glsresetentrycounter}{}%
6003     \renewcommand*\{glsstepentry}[1]{}%
6004     \renewcommand*\{glsentrycounterlabel}{}%
6005     \renewcommand*\{glsentryitem}[1]{\glsresetsubentrycounter}
6006   \fi
6007 }

```

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.

```

6008 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
6009   \csuse{glssubentrycounter#1}%
6010   \ifglssubentrycounter
6011     \ifundef\c@glossarysubentry
6012     {%
6013       \ifglssentrycounter
6014         \newcounter{glossarysubentry}[glossaryentry]%
6015       \else
6016         \newcounter{glossarysubentry}
6017       \fi
6018     }{}%
6019     \renewcommand*\{glsstepsubentry}[1]{%
6020       \edef\currentglssubentry{\glsdetoklabel{##1}}%
6021       \refstepcounter{glossarysubentry}%
6022       \label{glsentry-\currentglssubentry}%
6023     }%
6024     \renewcommand*\{glsresetsubentrycounter}{%
6025       \setcounter{glossarysubentry}{0}%
6026     }%
6027     \renewcommand*\{glssubentryitem}[1]{%
6028       \glsstepsubentry{##1}\glssubentrycounterlabel
6029     }%
6030     \renewcommand*\{glssubentrycounterlabel}{\theglossarysubentry\space}%
6031     \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6032   \else

```

```

6033 \renewcommand*{\glssubentryitem}[1]{}%
6034 \renewcommand*{\glsstepsubentry}[1]{}%
6035 \renewcommand*{\glsresetsubentrycounter}{}%
6036 \renewcommand*{\glssubentrycounterlabel}{}%
6037 \fi
6038 }

```

The nonnumberlist key determines if this glossary should have a number list.

```

6039 \define@boolkey{printgloss}[gls]{nonnumberlist}[true]{%
6040 \ifglsnonnumberlist
6041 \def\glossaryentrynumbers##1{}%
6042 \else
6043 \def\glossaryentrynumbers##1{##1}%
6044 \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

6045 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

@assign@sortkey Issue error if used with \printglossary

```

6046 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6047 \PackageError{glossaries}{'sort' key not permitted with
6048 \string\printglossary}%
6049 {The 'sort' key may only be used with \string\printnoidxglossary}%
6050 }

```

@assign@sortkey For use with \printnoidxglossary

```

6051 \newcommand*{\@glo@assign@sortkey}[1]{%
6052 \def\@glo@sorttype{#1}%
6053 }

```

\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.

```

6054 \newcommand*{\@glsnonextpages}{%
6055 \gdef\glossaryentrynumbers##1{%
6056 \glsresetentrylist
6057 }%
6058 }

```

\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.

```

6059 \newcommand*{\@glsnextpages}{%
6060 \gdef\glossaryentrynumbers##1{%
6061 ##1\glsresetentrylist}}

```



**sresetentrylist** Resets `\glossaryentrynumbers`

```

6062 \newcommand*{\glsresetentrylist}{%
6063   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

**\glsnonextpages** Outside of `\printglossary` this does nothing.

```

6064 \newcommand*{\glsnonextpages}{}

```

**\glsnextpages** Outside of `\printglossary` this does nothing.

```

6065 \newcommand*{\glsnextpages}{}

```

**glossaryentry** If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

6066 \ifglssentrycounter
6067   \ifx\@gls@counterwithin\@empty
6068     \newcounter{glossaryentry}
6069   \else
6070     \newcounter{glossaryentry}[\@gls@counterwithin]
6071   \fi
6072   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
6073 \fi

```

**lossarysubentry** If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```

6074 \ifglsssubentrycounter
6075   \ifglssentrycounter
6076     \newcounter{glossarysubentry}[glossaryentry]
6077   \else
6078     \newcounter{glossarysubentry}
6079   \fi
6080   \def\theHglossarysubentry{\currentglsssubentry.\theglossarysubentry}
6081 \fi

```

**subentrycounter** Resets the `glossarysubentry` counter.

```

6082 \ifglsssubentrycounter
6083   \newcommand*{\glsresetsubentrycounter}{%
6084     \setcounter{glossarysubentry}{0}%
6085   }
6086 \else
6087   \newcommand*{\glsresetsubentrycounter}{}
6088 \fi

```

**subentrycounter** Resets the `glossareentry` counter.

```

6089 \ifglssentrycounter
6090   \newcommand*{\glsresetentrycounter}{%
6091     \setcounter{glossaryentry}{0}%
6092   }
6093 \else
6094   \newcommand*{\glsresetentrycounter}{}
6095 \fi

```

`\glsstepentry` Advance the glossaryentry counter if in use. The argument is the label associated with the entry.

```

6096 \ifglentrycounter
6097   \newcommand*{\glsstepentry}[1]{%
6098     \refstepcounter{glossaryentry}%
6099     \label{glentry-\glsdetoklabel{#1}}%
6100   }
6101 \else
6102   \newcommand*{\glsstepentry}[1]{%
6103 \fi

```

`glsstepsubentry` Advance the glossarysubentry counter if in use. The argument is the label associated with the subentry.

```

6104 \ifglssubentrycounter
6105   \newcommand*{\glsstepsubentry}[1]{%
6106     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6107     \refstepcounter{glossarysubentry}%
6108     \label{glentry-\currentglssubentry}%
6109   }
6110 \else
6111   \newcommand*{\glsstepsubentry}[1]{%
6112 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

6113 \ifglentrycounter
6114   \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
6115 \else
6116   \ifglssubentrycounter
6117     \newcommand*{\glsrefentry}[1]{\ref{glentry-\glsdetoklabel{#1}}}
6118   \else
6119     \newcommand*{\glsrefentry}[1]{\gls{#1}}
6120   \fi
6121 \fi

```

`trycounterlabel` Defines how to display the glossaryentry counter.

```

6122 \ifglentrycounter
6123   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
6124 \else
6125   \newcommand*{\glsentrycounterlabel}{}
6126 \fi

```

`trycounterlabel` Defines how to display the glossarysubentry counter.

```

6127 \ifglssubentrycounter
6128   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry}\space}
6129 \else
6130   \newcommand*{\glssubentrycounterlabel}{}
6131 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```
6132 \ifglssentrycounter
6133   \newcommand*{\glsentryitem}[1]{%
6134     \glstepentry{#1}\glsentrycounterlabel
6135   }
6136 \else
6137   \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
6138 \fi
```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```
6139 \ifglssubentrycounter
6140   \newcommand*{\glssubentryitem}[1]{%
6141     \glstepsubentry{#1}\glssubentrycounterlabel
6142   }
6143 \else
6144   \newcommand*{\glssubentryitem}[1]{}
6145 \fi
```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
6146 \ifcsundef{theglossary}%
6147 {%
6148   \newenvironment{theglossary}{}{}%
6149 }%
6150 {%
6151   \@gls@warnontheGLOSSdefined
6152   \renewenvironment{theglossary}{}{}%
6153 }
```

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`

```
6154 \newcommand*{\glossaryheader}{}%
```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```
6155 \newcommand*{\glstarget}[2]{\@glstarget{\glolinkprefix#1}{#2}}
```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

atibleglossentry

`\glossentry{<label>}{<page-list>}`

```
6156 \providecommand*{\compatibleglossentry}[2]{%
6157   \toks@{#2}%
6158   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6159     {\noexpand\glsnamefont
6160       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6161     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6162     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6163     {\the\toks@}}%
6164   }%
6165   \@do@glossentry
6166 }
```

\glossentryname

```
6167 \newcommand*{\glossentryname}[1]{%
6168   \glsdoifexistsorwarn{#1}%
6169   {%
6170     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6171     \expandafter\glsnamefont\expandafter{\glo@name}%
6172   }%
6173 }
```

\Glossentryname

```
6174 \newcommand*{\Glossentryname}[1]{%
6175   \glsdoifexistsorwarn{#1}%
6176   {%
6177     \glsnamefont{\Glsentryname{#1}}%
6178   }%
6179 }
```

\glossentrydesc

```
6180 \newcommand*{\glossentrydesc}[1]{%
6181   \glsdoifexistsorwarn{#1}%
6182   {%
6183     \glsentrydesc{#1}%
6184   }%
6185 }
```

\Glossentrydesc

```
6186 \newcommand*{\Glossentrydesc}[1]{%
6187   \glsdoifexistsorwarn{#1}%
6188   {%
6189     \Glsentrydesc{#1}%
6190   }%
6191 }
```

lossentrysymbol

```

6192 \newcommand*{\glossentrysymbol}[1]{%
6193   \glsdoifexistsorwarn{#1}%
6194   {%
6195     \glsentrysymbol{#1}%
6196   }%
6197 }

```

lossentrysymbol

```

6198 \newcommand*{\Glossentrysymbol}[1]{%
6199   \glsdoifexistsorwarn{#1}%
6200   {%
6201     \Glsentrysymbol{#1}%
6202   }%
6203 }

```

blesubglossentry `\subglossentry{<level>}{<label>}{<page-list>}`

```

6204 \providecommand*{\compatiblesubglossentry}[3]{%
6205   \toks@{#3}%
6206   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6207     {#2}%
6208     {\noexpand\glsnamefont
6209       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6210     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6211     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6212     {\the\toks@}}%
6213   }%
6214   \@do@subglossentry
6215 }

```

rycompatibility

```

6216 \newcommand*{\setglossentrycompatibility}{%
6217   \let\glossentry\compatibleglossentry
6218   \let\subglossentry\compatiblesubglossentry
6219 }
6220 \setglossentrycompatibility

```

ossaryentryfield `\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command formerly governed how each entry row should be formatted in the glossary.  
Now deprecated.

```

6221 \newcommand{\glossaryentryfield}[5]{%
6222   \GlossariesWarning
6223   {Deprecated use of \string\glossaryentryfield.^^J
6224     I recommend you change to \string\glossentry.^^J

```

```

6225   If you've just upgraded, try removing your gls auxiliary
6226   files^^J and recompile}%
6227   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

arysubentryfield

```

\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}{<page-list>}

```

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore *<symbol>*. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```

6228 \newcommand*{\glossarysubentryfield}[6]{%
6229   \GlossariesWarning
6230   {Deprecated use of \string\glossarysubentryfield.^^J
6231    I recommend you change to \string\subglossentry.^^J
6232    If you've just upgraded, try removing your gls auxiliary
6233    files^^J and recompile}%
6234   \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

```

6235 \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

```

6236 \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.

`lsgetgrouptitle`

```
6237 \newcommand*{\glsgetgrouptitle}[1]{%
6238   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6239   \@gls@grptitle
6240 }
```

`s@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```
6241 \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.

```
6242   \dtl@ifsingle{#1}%
6243   {%
6244     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6245   }%
6246   {%
6247     \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6248               or test{\ifstrequal{#1}{glsnumbers}}}%
6249     {%
6250       \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6251     }%
6252     {%
6253       \def#2{#1}%
6254     }%
6255   }%
6256 }
```

`x@getgrouptitle` Version for the no-indexing app option:

```
6257 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6258   \DTLifint{#1}%
6259   {\edef#2{\char#1\relax}}%
6260   {%
6261     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6262   }%
6263 }
```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`\glsgetgrouplabel`

```
6264 \newcommand*{\glsgetgrouplabel}[1]{%
6265 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6266 \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`

```
6267 \newcommand*{\setentrycounter}[2][ ]{%
6268   \def\@glo@counterprefix{#1}%
6269   \ifx\@glo@counterprefix\empty
6270     \def\@glo@counterprefix{.}%
6271   \else
6272     \def\@glo@counterprefix{.#1.}%
6273   \fi
6274   \def\glsentrycounter{#2}%
6275 }
```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```
6276 \newcommand*{\setglossarystyle}[1]{%
6277   \ifcsundef{@glsstyle@#1}%
6278   {%
6279     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6280   }%
6281   {%
6282     \csname @glsstyle@#1\endcsname
6283   }%
```

Set the default style if it's not already set.

```
6284   \ifx\@glossary@default@style\relax
6285     \protected@edef\@glossary@default@style{#1}%
6286   \fi
6287 }
```

`\glossarystyle`

```
6288 \newcommand*{\glossarystyle}[1]{%
6289   \ifcsundef{@glsstyle@#1}%
6290   {%
6291     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6292   }%
6293   {%
6294     \GlossariesWarning
```



```

6295   {Deprecated command \string\glossarystyle.^~J
6296   I recommend you switch to \string\setglossarystyle\space unless
6297   you want to maintain backward compatibility}%
6298   \setglossentrycompatibility
6299   \csname @glsstyle@#1\endcsname

6300   \ifcsdef{@glscompstyle@#1}%
6301   {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6302   {}%
6303   }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6304   \ifx\@glossary@default@style\relax
6305   \protected@edef\@glossary@default@style{#1}%
6306   \fi
6307 }

```

`\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.

```

6308 \newcommand{\newglossarystyle}[2]{%
6309   \ifcsundef{@glsstyle@#1}%
6310   {%
6311     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6312   }%
6313   {%
6314     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6315   }%
6316 }

```

`\newglossarystyle` Code for this macro supplied by Marco Daniel.

```

6317 \newcommand{\renewglossarystyle}[2]{%
6318   \ifcsundef{@glsstyle@#1}%
6319   {%
6320     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6321   }%
6322   {%
6323     \csdef{@glsstyle@#1}{#2}%
6324   }%
6325 }

```

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`

```
6326 \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`

```
6327 \ifcsundef{hyperlink}%
6328 {%
6329   \def\glshypernumber#1{#1}%
6330 }%
6331 {%
6332   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6333 }
```

`@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6334 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6335   \ifx\#1\%
6336   \else
6337     \@delimR#1\delimR\delimR\%
6338   \fi
6339   \ifx\#2\%
6340   \else
6341     #2%
6342   \fi
6343   \ifx\#3\%
6344   \else
6345     \@glshypernumber#3\@nil
6346   \fi
6347 }
```

`\@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

```

6348 \def\@delimR#1\delimR #2\delimR #3\\{%
6349 \ifx\\#2\\%
6350   \@delimN{#1}%
6351 \else
6352   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6353 \fi}

```

\@delimN displays a list of individual numbers, instead of a range:

\@delimN

```

6354 \def\@delimN#1{\@delimN#1\delimN \delimN\\}
6355 \def\@delimN#1\delimN #2\delimN#3\\{%
6356 \ifx\\#3\\%
6357   \@gls@numberlink{#1}%
6358 \else
6359   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6360 \fi
6361 }

```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

6362 \def\@gls@numberlink#1{%
6363 \begingroup
6364 \toks@={}%
6365 \@gls@removespaces#1 \@nil
6366 \endgroup}

6367 \def\@gls@removespaces#1 #2\@nil{%
6368 \toks@=\expandafter{\the\toks@#1}%
6369 \ifx\\#2\\%
6370   \edef\x{\the\toks@}%
6371   \ifx\x\empty
6372     \else

6373     \hyperlink{\glsentrycounter\@glo@counterprefix\the\toks@}%
6374               {\the\toks@}%
6375   \fi
6376 \else
6377   \@gls@ReturnAfterFi{%
6378     \@gls@removespaces#2\@nil
6379   }%
6380 \fi
6381 }
6382 \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
6383 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}

\hypersf
6384 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}

\hypertt
6385 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
6386 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}

\hypermd
6387 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}

\hyperit
6388 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
6389 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
6390 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
6391 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
6392 \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.

```

6393 \newcommand{\oldacronym}[4][\gls@label]{%
6394   \def\gls@label{#2}%
6395   \newacronym[#4]{#1}{#2}{#3}%
6396   \ifcsundef{xspace}%
6397   {%
6398     \expandafter\edef\csname#1\endcsname{%
6399       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6400   }%
6401   }%
6402   {%
6403     \expandafter\edef\csname#1\endcsname{%
6404       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6405         \noexpand\gls{#1}\noexpand\xspace}%
6406     }%
6407   }%
6408 }

```

`\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`

```

6409 \newcommand{\newacronym}[4][[]{}

```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix` Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn’t appear in small caps as it doesn’t look right. For example, `ABCS` looks as though the “s” is part of the acronym, but `ABCS` looks as though the “s” is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is need to cancel it out.

```

6410 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}

```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```

6411 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}

```

The following are defined for compatibility with version 2.07 and earlier.

```

\glsshortkey
6412 \newcommand*{\glsshortkey}{short}

sshortpluralkey
6413 \newcommand*{\glsshortpluralkey}{shortplural}

\glslongkey
6414 \newcommand*{\glslongkey}{long}

lslongpluralkey
6415 \newcommand*{\glslongpluralkey}{longplural}

\acrfull  Full form of the acronym.
6416 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\@ns@acrfull}

6417 \newcommand*{\ns@acrfull}[2][{}]{%
6418   \new@ifnextchar[{\@acrfull{#1}{#2}}]{%
6419     {\@acrfull{#1}{#2}}[{}]}%
6420 }

\@acrfull  Low-level macro:
6421 \def\@acrfull#1#2[#3]{%
  Make it easier for acronym styles to change this:
6422   \acrfullfmt{#1}{#2}{#3}%
6423 }

  Using \acrlinkfullformat and \acrfullformat is now deprecated as it can cause com-
  plications with the first letter upper case variants, but the package needs to provide backward
  compatibility support.

\acrfullfmt  No case change full format.
6424 \newcommand*{\acrfullfmt}[3]{%
6425   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6426 }

acrlinkfullformat  Format for full links like \acrfull. Syntax: \acrlinkfullformat{<long cs>}{<short cs>}
  {<options>}{<label>}{<insert>}
6427 \newcommand{\acrlinkfullformat}[5]{%
6428   \acrfullformat{#1}{#3}{#4}[#5]{#2}{#3}{#4}[{}]}%
6429 }

\acrfullformat  Default full form is <long> (<short>).
6430 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}

\glsspace  Robust space to ensure it's written to the .glsdefs file.
6431 \newrobustcmd{\glsspace}{\space}

```

## Default format for full acronym

`\Acrfull`

```
6432 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}

6433 \newcommand*\ns@Acrfull[2] [] {%
6434   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6435     {\@Acrfull{#1}{#2} []}%
6436 }
```

Low-level macro:

```
6437 \def\@Acrfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6438   \Acrfullfmt{#1}{#2}{#3}%
6439 }
```

`\Acrfullfmt` First letter upper case full format.

```
6440 \newcommand*{\Acrfullfmt}[3] {%
6441   \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%
6442 }
```

`\ACRfull`

```
6443 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}

6444 \newcommand*\ns@ACRfull[2] [] {%
6445   \new@ifnextchar[{\@ACRfull{#1}{#2}}%
6446     {\@ACRfull{#1}{#2} []}%
6447 }
```

Low-level macro:

```
6448 \def\@ACRfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6449   \ACRfullfmt{#1}{#2}{#3}%
6450 }
```

`\ACRfullfmt` All upper case full format.

```
6451 \newcommand*{\ACRfullfmt}[3] {%
6452   \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%
6453 }
```

Plural:

`\acrfullpl`

```
6454 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}

6455 \newcommand*\ns@acrfullpl[2] [] {%
6456   \new@ifnextchar[{\@acrfullpl{#1}{#2}}%
6457     {\@acrfullpl{#1}{#2} []}%
6458 }
```

Low-level macro:

```
6459 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6460 \acrfullplfmt{#1}{#2}{#3}%  
6461 }
```

\acrfullplfmt No case change plural full format.

```
6462 \newcommand*\@acrfullplfmt}[3]{%  
6463 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6464 }
```

\Acrfullpl

```
6465 \newrobustcmd*\@Acrfullpl{\@gls@hyp@opt\ns@Acrfullpl}  
  
6466 \newcommand*\ns@Acrfullpl[2][{}]{%  
6467 \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6468 {\@Acrfullpl{#1}{#2}[]}%  
6469 }
```

Low-level macro:

```
6470 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6471 \Acrfullplfmt{#1}{#2}{#3}%  
6472 }
```

\Acrfullplfmt First letter upper case plural full format.

```
6473 \newcommand*\@Acrfullplfmt}[3]{%  
6474 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6475 }
```

\ACRfullpl

```
6476 \newrobustcmd*\@ACRfullpl{\@gls@hyp@opt\ns@ACRfullpl}  
  
6477 \newcommand*\ns@ACRfullpl[2][{}]{%  
6478 \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%  
6479 {\@ACRfullpl{#1}{#2}[]}%  
6480 }
```

Low-level macro:

```
6481 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6482 \ACRfullplfmt{#1}{#2}{#3}%  
6483 }
```

\ACRfullplfmt All upper case plural full format.

```
6484 \newcommand*\@ACRfullplfmt}[3]{%  
6485 \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6486 }
```



## 1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6487 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6488 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}
```

`\acrnameformat` The styles that allow an additional description use `\acrnameformat{<short>}{<long>}` to determine what information is displayed in the name.

```
6489 \newcommand*{\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6490 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6491 \newtoks\glslabeltok
```

`\glsshorttok`

```
6492 \newtoks\glsshorttok
```

`\glslongtok`

```
6493 \newtoks\glslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
6494 \newcommand*{\newacronymhook}{}
```

`\genericNewAcronym` New improved version of setting the acronym style.

```
6495 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`

```
6496 \let\@Gls@entryname\@Gls@acrentryname
```

Change the way acronyms are defined:

```
6497 \renewcommand{\newacronym}[4][\{%
6498 \ifdefempty{\@glsacronymlists}%
6499 {%
6500 \def\@glo@type{\acronymtype}%
6501 \setkeys{glossentry}{##1}%
6502 \DeclareAcronymList{\@glo@type}%
6503 }%
6504 }%
6505 \glskeylisttok{##1}%
6506 \glslabeltok{##2}%
6507 \glsshorttok{##3}%
6508 \glslongtok{##4}%
```

```

6509 \newacronymhook
6510 \protected@edef\@do@newglossaryentry{%
6511 \noexpand\newglossaryentry{\the\glslabeltok}%
6512 {%
6513 type=\acronymtype,%
6514 name={\expandonce{\acronymentry{##2}}},%
6515 sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6516 text={\the\glsshorttok},%
6517 short={\the\glsshorttok},%
6518 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6519 long={\the\glslongtok},%
6520 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6521 \GenericAcronymFields,%
6522 \the\glskeylisttok
6523 }%
6524 }%
6525 \@do@newglossaryentry
6526 }%

```

Make sure that \acrfull etc reflects the new style:

```

6527 \renewcommand*{\acrfullfmt}[3]{%
6528 \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6529 \renewcommand*{\Acrfullfmt}[3]{%
6530 \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6531 \renewcommand*{\ACRfullfmt}[3]{%
6532 \glslink[##1]{##2}{%
6533 \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
6534 \renewcommand*{\acrfullplfmt}[3]{%
6535 \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6536 \renewcommand*{\Acrfullplfmt}[3]{%
6537 \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6538 \renewcommand*{\ACRfullplfmt}[3]{%
6539 \glslink[##1]{##2}{%
6540 \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6541 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6542 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6543 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6544 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6545 }

```

\icAcronymFields Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

6546 \newcommand*{\GenericAcronymFields}{description={\the\glslongtok}}

```

\acronymentry `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6547 \newcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{#1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```
6548 \newcommand*{\acronymsort}[2]{#1}
```

`\setacronymstyle` `\setacronymstyle{<style name>}`

```
6549 \newcommand*{\setacronymstyle}[1]{%
6550   \ifcsundef{@glsacr@dispstyle@#1}
6551   {%
6552     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6553   }%
6554   {%
6555     \ifdefempty{\@glsacronymlists}%
6556     {%
6557       \DeclareAcronymList{\acronymtype}%
6558     }%
6559     {}%
6560     \SetGenericNewAcronym
6561     \GlsUseAcrStyleDefs{#1}%
6562     \@for\@gls@type:=\@glsacronymlists\do{%
6563       \defglentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6564     }%
6565   }%
6566 }
```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```
6567 \newcommand*{\newacronymstyle}[3]{%
6568   \ifcsdef{@glsacr@dispstyle@#1}%
6569   {%
6570     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6571   }%
6572   {%
6573     \csdef{@glsacr@dispstyle@#1}{#2}%
6574     \csdef{@glsacr@styledefs@#1}{#3}%
6575   }%
6576 }
```

`\renewacronymstyle` Redefines the given acronym style.

```
6577 \newcommand*{\renewacronymstyle}[3]{%
6578   \ifcsdef{@glsacr@dispstyle@#1}%
6579   {%
```

```

6580 \csdef{@glsacr@dispstyle@#1}{#2}%
6581 \csdef{@glsacr@styledefs@#1}{#3}%
6582 }%
6583 {%
6584 \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6585 }%
6586 }

```

EntryDispStyle

```

6587 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}

```

UseAcrStyleDefs

```

6588 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}

```

Predefined acronym styles:

long-short    *<long>* (*<short>*) acronym style.

```

6589 \newacronymstyle{long-short}%
6590 {%

```

Check for long form in case this is a mixed glossary.

```

6591 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6592 }%
6593 {%
6594 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6595 \renewcommand*{\genacrfullformat}[2]{%
6596 \glsentrylong{##1}##2\space
6597 (\protect\firstacronymfont{\glsentryshort{##1}})%
6598 }%
6599 \renewcommand*{\Genacrfullformat}[2]{%
6600 \Glsentrylong{##1}##2\space
6601 (\protect\firstacronymfont{\glsentryshort{##1}})%
6602 }%
6603 \renewcommand*{\genplacrfullformat}[2]{%
6604 \glsentrylongpl{##1}##2\space
6605 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6606 }%
6607 \renewcommand*{\Genplacrfullformat}[2]{%
6608 \Glsentrylongpl{##1}##2\space
6609 (\protect\firstacronymfont{\glsentryshortpl{##1}})%
6610 }%
6611 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6612 \renewcommand*{\acronymsort}[2]{##1}%
6613 \renewcommand*{\acronymfont}[1]{##1}%
6614 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6615 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6616 }

```

long-sp-short    Similar to the previous style but allows the space between the long and short form to be customized.

```

6617 \newacronymstyle{long-sp-short}%
6618 {%
    Check for long form in case this is a mixed glossary.
6619 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6620 }%
6621 {%
6622 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6623 \renewcommand*{\genacrfullformat}[2]{%
6624 \glentrylong{##1}##2\glsacspace{##1}%
6625 (\protect\firstacronymfont{\glentryshort{##1}})%
6626 }%
6627 \renewcommand*{\Genacrfullformat}[2]{%
6628 \Glsentrylong{##1}##2\glsacspace{##1}%
6629 (\protect\firstacronymfont{\glentryshort{##1}})%
6630 }%
6631 \renewcommand*{\genplacrfullformat}[2]{%
6632 \glentrylongpl{##1}##2\glsacspace{##1}%
6633 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6634 }%
6635 \renewcommand*{\Genplacrfullformat}[2]{%
6636 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6637 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6638 }%
6639 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6640 \renewcommand*{\acronymsort}[2]{##1}%
6641 \renewcommand*{\acronymfont}[1]{##1}%
6642 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6643 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6644 }

```

`\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.

```

6645 \newcommand*{\glsacspace}[1]{%
6646 \settowidth{\dimen@}{(\firstacronymfont{\glentryshort{##1}})}%
6647 \ifdim\dimen@<3em~\else\space\fi
6648 }

```

`short-long` *(short)* (*long*) acronym style.

```

6649 \newacronymstyle{short-long}%
6650 {%
    Check for long form in case this is a mixed glossary.
6651 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6652 }%
6653 {%
6654 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6655 \renewcommand*{\genacrfullformat}[2]{%
6656 \protect\firstacronymfont{\glentryshort{##1}}##2\space
6657 (\glentrylong{##1})%

```

```

6658 }%
6659 \renewcommand*{\Genacrfullformat}[2]{%
6660   \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6661   (\glsentrylong{##1})%
6662 }%
6663 \renewcommand*{\genplacrfullformat}[2]{%
6664   \protect\firstacronymfont{\glsentryshortpl{##1}}##2\space
6665   (\glsentrylongpl{##1})%
6666 }%
6667 \renewcommand*{\Genplacrfullformat}[2]{%
6668   \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6669   (\glsentrylongpl{##1})%
6670 }%

6671 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6672 \renewcommand*{\acronymsort}[2]{##1}%
6673 \renewcommand*{\acronymfont}[1]{##1}%
6674 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6675 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6676 }

```

long-sc-short    *<long>* (\textsc{<short>}) acronym style.

```

6677 \newacronymstyle{long-sc-short}%
6678 {%
6679   \GlsUseAcrEntryDisplayStyle{long-short}%
6680 }%
6681 {%
6682   \GlsUseAcrStyleDefs{long-short}%
6683   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6684   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6685 }

```

long-sm-short    *<long>* (\textsmaller{<short>}) acronym style.

```

6686 \newacronymstyle{long-sm-short}%
6687 {%
6688   \GlsUseAcrEntryDisplayStyle{long-short}%
6689 }%
6690 {%
6691   \GlsUseAcrStyleDefs{long-short}%
6692   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6693   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6694 }

```

sc-short-long    *<short>* (\textsc{<long>}) acronym style.

```

6695 \newacronymstyle{sc-short-long}%
6696 {%
6697   \GlsUseAcrEntryDisplayStyle{short-long}%
6698 }%
6699 {%

```

```

6700 \GlsUseAcrStyleDefs{short-long}%
6701 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6702 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6703 }

```

sm-short-long *<short>* (*\textsmaller{<long>}*) acronym style.

```

6704 \newacronymstyle{sm-short-long}%
6705 {%
6706 \GlsUseAcrEntryDisplayStyle{short-long}%
6707 }%
6708 {%
6709 \GlsUseAcrStyleDefs{short-long}%
6710 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6711 \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6712 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6713 \newacronymstyle{long-short-desc}%
6714 {%
6715 \GlsUseAcrEntryDisplayStyle{long-short}%
6716 }%
6717 {%
6718 \GlsUseAcrStyleDefs{long-short}%
6719 \renewcommand*{\GenericAcronymFields}{}%
6720 \renewcommand*{\acronymsort}[2]{##2}%
6721 \renewcommand*{\acronymentry}[1]{%
6722 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6723 }

```

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`.

```

6724 \newacronymstyle{long-sp-short-desc}%
6725 {%
6726 \GlsUseAcrEntryDisplayStyle{long-sp-short}%
6727 }%
6728 {%
6729 \GlsUseAcrStyleDefs{long-sp-short}%
6730 \renewcommand*{\GenericAcronymFields}{}%
6731 \renewcommand*{\acronymsort}[2]{##2}%
6732 \renewcommand*{\acronymentry}[1]{%
6733 \glentrylong{##1}\glsacspace{##1}(\acronymfont{\glentryshort{##1}})}%
6734 }

```

g-sc-short-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6735 \newacronymstyle{long-sc-short-desc}%
6736 {%

```

```

6737 \GlsUseAcrEntryDispStyle{long-sc-short}%
6738 }%
6739 {%
6740 \GlsUseAcrStyleDefs{long-sc-short}%
6741 \renewcommand*{\GenericAcronymFields}{}%
6742 \renewcommand*{\acronymsort}[2]{##2}%
6743 \renewcommand*{\acronymentry}[1]{%
6744     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6745 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6746 \newacronymstyle{long-sm-short-desc}%
6747 {%
6748 \GlsUseAcrEntryDispStyle{long-sm-short}%
6749 }%
6750 {%
6751 \GlsUseAcrStyleDefs{long-sm-short}%
6752 \renewcommand*{\GenericAcronymFields}{}%
6753 \renewcommand*{\acronymsort}[2]{##2}%
6754 \renewcommand*{\acronymentry}[1]{%
6755     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6756 }

```

short-long-desc *<short>* ({<long>}) acronym style that has an accompanying description (which the user needs to supply).

```

6757 \newacronymstyle{short-long-desc}%
6758 {%
6759 \GlsUseAcrEntryDispStyle{short-long}%
6760 }%
6761 {%
6762 \GlsUseAcrStyleDefs{short-long}%
6763 \renewcommand*{\GenericAcronymFields}{}%
6764 \renewcommand*{\acronymsort}[2]{##2}%
6765 \renewcommand*{\acronymentry}[1]{%
6766     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6767 }

```

short-long-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6768 \newacronymstyle{sc-short-long-desc}%
6769 {%
6770 \GlsUseAcrEntryDispStyle{sc-short-long}%
6771 }%
6772 {%
6773 \GlsUseAcrStyleDefs{sc-short-long}%
6774 \renewcommand*{\GenericAcronymFields}{}%
6775 \renewcommand*{\acronymsort}[2]{##2}%
6776 \renewcommand*{\acronymentry}[1]{%

```



```

6777 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6778 }

```

short-long-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6779 \newacronymstyle{sm-short-long-desc}%
6780 {%
6781 \GlsUseAcrEntryDispStyle{sm-short-long}%
6782 }%
6783 {%
6784 \GlsUseAcrStyleDefs{sm-short-long}%
6785 \renewcommand*{\GenericAcronymFields}{}%
6786 \renewcommand*{\acronymsort}[2]{##2}%
6787 \renewcommand*{\acronymentry}[1]{%
6788 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6789 }

```

dua *<long>* only acronym style.

```

6790 \newacronymstyle{dua}%
6791 {%

```

Check for long form in case this is a mixed glossary.

```

6792 \ifdefempty\glscustomtext
6793 {%
6794 \ifglshaslong{\glslabel}%
6795 {%
6796 \glsifplural
6797 {%

```

Plural form:

```

6798 \glscapscase
6799 {%

```

Plural form, don't adjust case:

```

6800 \glsentrylongpl{\glslabel}\glsinsert
6801 }%
6802 {%

```

Plural form, make first letter upper case:

```

6803 \Glsentrylongpl{\glslabel}\glsinsert
6804 }%
6805 {%

```

Plural form, all caps:

```

6806 \mfirstucMakeUppercase
6807 {\glsentrylongpl{\glslabel}\glsinsert}%
6808 }%
6809 }%
6810 {%

```

Singular form

```
6811      \glscapscase
6812      {%
```

Singular form, don't adjust case:

```
6813      \glentrylong{\glslabel}\glsinsert
6814      }%
6815      {%
```

Subsequent singular form, make first letter upper case:

```
6816      \Glsentrylong{\glslabel}\glsinsert
6817      }%
6818      {%
```

Subsequent singular form, all caps:

```
6819      \mfirstucMakeUppercase
6820      {\glentrylong{\glslabel}\glsinsert}%
6821      }%
6822      }%
6823      }%
6824      {%
```

Not an acronym:

```
6825      \glsgenentryfmt
6826      }%
6827      }%
6828      {\glscustomtext\glsinsert}%
6829      }%
6830      {%
6831      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6832      \renewcommand*{\acrfullfmt}[3]{%
6833          \glslink[##1]{##2}{\glentrylong{##2}##3\space
6834              (\acronymfont{\glentryshort{##2}})}}%
6835      \renewcommand*{\Acrfullfmt}[3]{%
6836          \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6837              (\acronymfont{\glentryshort{##2}})}}%
6838      \renewcommand*{\ACRfullfmt}[3]{%
6839          \glslink[##1]{##2}{%
6840              \mfirstucMakeUppercase{\glentrylong{##2}##3\space
6841                  (\acronymfont{\glentryshort{##2}})}}}%

6842      \renewcommand*{\acrfullplfmt}[3]{%
6843          \glslink[##1]{##2}{\glentrylongpl{##2}##3\space
6844              (\acronymfont{\glentryshortpl{##2}})}}%

6845      \renewcommand*{\Acrfullplfmt}[3]{%
6846          \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6847              (\acronymfont{\glentryshortpl{##2}})}}%
6848      \renewcommand*{\ACRfullplfmt}[3]{%
6849          \glslink[##1]{##2}{%
```

```

6850      \mfirstucMakeUppercase{\glentrylongpl{##2}##3\space
6851      (\acronymfont{\glentryshortpl{##2}})}}}%
6852 \renewcommand*{\glentryfull}[1]{%
6853   \glentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6854 }%
6855 \renewcommand*{\Glsentryfull}[1]{%
6856   \Glsentrylong{##1}\space(\acronymfont{\glentryshort{##1}})%
6857 }%
6858 \renewcommand*{\glentryfullpl}[1]{%
6859   \glentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6860 }%
6861 \renewcommand*{\Glsentryfullpl}[1]{%
6862   \Glsentrylongpl{##1}\space(\acronymfont{\glentryshortpl{##1}})%
6863 }%
6864 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6865 \renewcommand*{\acronymsort}[2]{##1}%
6866 \renewcommand*{\acronymfont}[1]{##1}%
6867 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6868 }

```

dua-desc    *<long>* only acronym style with user-supplied description.

```

6869 \newacronymstyle{dua-desc}%
6870 {%
6871   \GlsUseAcrEntryDispStyle{dua}%
6872 }%
6873 {%
6874   \GlsUseAcrStyleDefs{dua}%
6875   \renewcommand*{\GenericAcronymFields}{}%
6876   \renewcommand*{\acronymentry}[1]{\acronymfont{\glentrylong{##1}}}%
6877   \renewcommand*{\acronymsort}[2]{##2}%
6878 }%

```

footnote    *<short>*\footnote{*<long>*} acronym style.

```

6879 \newacronymstyle{footnote}%
6880 {%
6881   Check for long form in case this is a mixed glossary.
6882   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6883 }%
6884 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6885   Need to ensure hyperlinks are switched off on first use:
6885   \glshyperfirstfalse
6886   \renewcommand*{\genacrfullformat}[2]{%
6887     \protect\firstacronymfont{\glentryshort{##1}}##2%
6888     \protect\footnote{\glentrylong{##1}}%
6889   }%
6890   \renewcommand*{\Genacrfullformat}[2]{%

```

```

6891 \firstacronymfont{\Glsentryshort{##1}}##2%
6892 \protect\footnote{\glentrylong{##1}}%
6893 }%
6894 \renewcommand*{\genplacrfullformat}[2]{%
6895 \protect\firstacronymfont{\glentryshortpl{##1}}##2%
6896 \protect\footnote{\glentrylongpl{##1}}%
6897 }%
6898 \renewcommand*{\Genplacrfullformat}[2]{%
6899 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6900 \protect\footnote{\glentrylongpl{##1}}%
6901 }%
6902 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6903 \renewcommand*{\acronymsort}[2]{##1}%
6904 \renewcommand*{\acronymfont}[1]{##1}%
6905 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6906 \renewcommand*{\acrfullfmt}[3]{%
6907 \glslink[##1]{##2}{\acronymfont{\glentryshort{##2}}##3\space
6908 (\glentrylong{##2})}%
6909 \renewcommand*{\Acrfullfmt}[3]{%
6910 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6911 (\glentrylong{##2})}%
6912 \renewcommand*{\ACRfullfmt}[3]{%
6913 \glslink[##1]{##2}{%
6914 \mfirstucMakeUppercase{\acronymfont{\glentryshort{##2}}##3\space
6915 (\glentrylong{##2})}}}%
6916 \renewcommand*{\acrfullplfmt}[3]{%
6917 \glslink[##1]{##2}{\acronymfont{\glentryshortpl{##2}}##3\space
6918 (\glentrylongpl{##2})}%
6919 \renewcommand*{\Acrfullplfmt}[3]{%
6920 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6921 (\glentrylongpl{##2})}%
6922 \renewcommand*{\ACRfullplfmt}[3]{%
6923 \glslink[##1]{##2}{%
6924 \mfirstucMakeUppercase{\acronymfont{\glentryshortpl{##2}}##3\space
6925 (\glentrylongpl{##2})}}}%

```

Similarly for \glentryfull etc:

```

6926 \renewcommand*{\glentryfull}[1]{%
6927 \acronymfont{\glentryshort{##1}}\space(\glentrylong{##1})}%
6928 \renewcommand*{\Glsentryfull}[1]{%
6929 \acronymfont{\Glsentryshort{##1}}\space(\glentrylong{##1})}%
6930 \renewcommand*{\glentryfullpl}[1]{%
6931 \acronymfont{\glentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6932 \renewcommand*{\Glsentryfullpl}[1]{%
6933 \acronymfont{\Glsentryshortpl{##1}}\space(\glentrylongpl{##1})}%
6934 }

```

footnote-sc \textsc{\short}\footnote{\long} acronym style.

```

6935 \newacronymstyle{footnote-sc}%
6936 {%
6937   \GlsUseAcrEntryDisplayStyle{footnote}%
6938 }%
6939 {%
6940   \GlsUseAcrStyleDefs{footnote}%
6941   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6942   \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6943   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6944 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6945 \newacronymstyle{footnote-sm}%
6946 {%
6947   \GlsUseAcrEntryDisplayStyle{footnote}%
6948 }%
6949 {%
6950   \GlsUseAcrStyleDefs{footnote}%
6951   \renewcommand{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}
6952   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6953   \renewcommand*{\acrpluralsuffix}{\glacrpluralsuffix}%
6954 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6955 \newacronymstyle{footnote-desc}%
6956 {%
6957   \GlsUseAcrEntryDisplayStyle{footnote}%
6958 }%
6959 {%
6960   \GlsUseAcrStyleDefs{footnote}%
6961   \renewcommand*{\GenericAcronymFields}{}%
6962   \renewcommand*{\acronymsort}[2]{##2}%
6963   \renewcommand*{\acronymentry}[1]{%
6964     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6965 }

```

footnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6966 \newacronymstyle{footnote-sc-desc}%
6967 {%
6968   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
6969 }%
6970 {%
6971   \GlsUseAcrStyleDefs{footnote-sc}%
6972   \renewcommand*{\GenericAcronymFields}{}%
6973   \renewcommand*{\acronymsort}[2]{##2}%
6974   \renewcommand*{\acronymentry}[1]{%
6975     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

```

6976 }

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

6977 \newacronymstyle{footnote-sm-desc}%

6978 {%

6979 \GlsUseAcrEntryDispStyle{footnote-sm}%

6980 }%

6981 {%

6982 \GlsUseAcrStyleDefs{footnote-sm}%

6983 \renewcommand\*{\GenericAcronymFields}{}%

6984 \renewcommand\*{\acronymsort}[2]{##2}%

6985 \renewcommand\*{\acronymentry}[1]{%

6986 \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%

6987 }

## AcronymSynonyms

6988 \newcommand\*{\DefineAcronymSynonyms}{%

### Short form

\acs

6989 \let\acs\acrshort

### First letter uppercase short form

\Acs

6990 \let\Acs\Acrshort

### Plural short form

\acsp

6991 \let\acsp\acrshortpl

### First letter uppercase plural short form

\Acsp

6992 \let\Acsp\Acrshortpl

### Long form

\acl

6993 \let\acl\aclong

### Plural long form

\aclp

6994 \let\aclp\aclongpl

### First letter upper case long form

`\Acl`  
 6995 `\let\Acl\Acrlong`  
 First letter upper case plural long form

`\Aclp`  
 6996 `\let\Aclp\Acrlongpl`  
 Full form

`\acf`  
 6997 `\let\acf\acrfull`  
 Plural full form

`\acfp`  
 6998 `\let\acfp\acrfullpl`  
 First letter upper case full form

`\Acf`  
 6999 `\let\Acf\Acrfull`  
 First letter upper case plural full form

`\Acfp`  
 7000 `\let\Acfp\Acrfullpl`  
 Standard form

`\ac`  
 7001 `\let\ac\gls`  
 First upper case standard form

`\Ac`  
 7002 `\let\Ac\Gls`  
 Standard plural form

`\acp`  
 7003 `\let\acp\glspl`  
 Standard first letter upper case plural form

`\Acp`  
 7004 `\let\Acp\Glspl`  
 7005 }  
 Define synonyms if required

7006 `\ifglsacrshortcuts`  
 7007 `\DefineAcronymSynonyms`  
 7008 `\fi`

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\setAcronymDisplayStyle` Sets the default acronym display style for given glossary.

```
7009 \newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%
7010   \def\glsentryfmt[#1]{\glsentryfmt}%
7011 }
```

`\setNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```
7012 \newcommand*{\DefaultNewAcronymDef}{%
7013   \edef\@do@newglossaryentry{%
7014     \noexpand\newglossaryentry{\the\glslabeltok}%
7015     {%
7016       type=\acronymtype,%
7017       name={\the\glsshorttok},%
7018       sort={\the\glsshorttok},%
7019       text={\the\glsshorttok},%
7020       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7021       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7022       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
7023                     {\noexpand\expandonce\noexpand\@glo@shortpl}},%
7024       short={\the\glsshorttok},%
7025       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7026       long={\the\glslongtok},%
7027       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7028       description={\the\glslongtok},%
7029       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
```

Remaining options specified by the user:

```
7030       \the\glskeylisttok
7031     }%
7032   }%
7033   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7034   \let\@org@gls@assign@plural\gls@assign@plural
7035   \let\@org@gls@assign@descplural\gls@assign@descplural
7036   \def\gls@assign@firstpl##1##2{%
7037     \@@gls@expand@field{##1}{firstpl}{##2}%
7038   }%
7039   \def\gls@assign@plural##1##2{%
7040     \@@gls@expand@field{##1}{plural}{##2}%
7041   }%
7042   \def\gls@assign@descplural##1##2{%
7043     \@@gls@expand@field{##1}{descplural}{##2}%
7044   }%
7045   \@do@newglossaryentry
7046   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7047   \let\gls@assign@plural\@org@gls@assign@plural
7048   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7049 }
```



ultAcronymStyle Set up the default acronym style:

```
7050 \newcommand*{\SetDefaultAcronymStyle}{%
```

Set the display style:

```
7051 \@for\@gls@type:=\@glsacronymlists\do{%
```

```
7052 \SetDefaultAcronymDisplayStyle{\@gls@type}%
```

```
7053 }%
```

Set up the definition of \newacronym:

```
7054 \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).

```
7055 \ifx\@glsacronymlists\@empty
```

```
7056 \def\@glo@type{\acronymtype}%
```

```
7057 \setkeys{glossentry}{##1}%
```

```
7058 \DeclareAcronymList{\@glo@type}%
```

```
7059 \SetDefaultAcronymDisplayStyle{\@glo@type}%
```

```
7060 \fi
```

```
7061 \glskeylisttok{##1}%
```

```
7062 \glslabeltok{##2}%
```

```
7063 \glsshorttok{##3}%
```

```
7064 \glslongtok{##4}%
```

```
7065 \newacronymhook
```

```
7066 \DefaultNewAcronymDef
```

```
7067 }%
```

```
7068 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
```

```
7069 }
```

\acrfootnote Used by the footnote acronym styles.

```
7070 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

acrlinkfootnote

```
7071 \newcommand*{\acrlinkfootnote}[3]{%
```

```
7072 \footnote{\glslink[#1]{#2}{#3}}%
```

```
7073 }
```

acrnolinkfootnote

```
7074 \newcommand*{\acrnolinkfootnote}[3]{%
```

```
7075 \footnote{#3}%
```

```
7076 }
```

acronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```
7077 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
```

```
7078 \defglsentryfmt[#1]{%
```

```
7079 \ifdefempty\glscustomtext
```

```
7080 {%
```

```
7081 \ifglsused{\glslabel}%
```

```

7082     {%
7083         \acronymfont{\glsgenentryfmt}%
7084     }%
7085     {%
7086         \firstacronymfont{\glsgenentryfmt}%
7087         \ifglshassymbol{\glslabel}%
7088     {%
7089         \expandafter\protect\expandafter\acrfootnote\expandafter
7090         {\@gls@link@opts}{\@gls@link@label}%
7091     {%
7092         \glsifplural
7093         {\glsentrysymbolplural{\glslabel}}%
7094         {\glsentrysymbol{\glslabel}}%
7095     }%
7096     }%
7097 }%
7098 }%
7099 {\glscustomtext\glsinsert}%
7100 }%
7101 }

```

teNewAcronymDef

```

7102 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7103     \edef\@do@newglossaryentry{%
7104         \noexpand\newglossaryentry{\the\glslabeltok}%
7105     {%
7106         type=\acronymtype,%
7107         name={\noexpand\acronymfont{\the\glsshorttok}},%
7108         sort={\the\glsshorttok},%
7109         first={\the\glsshorttok},%
7110         firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7111         text={\the\glsshorttok},%
7112         plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7113         short={\the\glsshorttok},%
7114         shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7115         long={\the\glslongtok},%
7116         longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7117         symbol={\the\glslongtok},%
7118         symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7119         \the\glskeylisttok
7120     }%
7121 }%
7122 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7123 \let\@org@gls@assign@plural\gls@assign@plural
7124 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7125 \def\gls@assign@firstpl##1##2{%
7126     \@gls@expand@field{##1}{firstpl}{##2}%
7127 }%
7128 \def\gls@assign@plural##1##2{%

```

```

7129 \@@gls@expand@field{##1}{plural}{##2}%
7130 }%
7131 \def\gls@assign@symbolplural##1##2{%
7132 \@@gls@expand@field{##1}{symbolplural}{##2}%
7133 }%
7134 \do@newglossaryentry
7135 \let\gls@assign@plural\@org@gls@assign@plural
7136 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7137 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7138 }

```

**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.

```

7139 \newcommand*\SetDescriptionFootnoteAcronymStyle{%
7140 \renewcommand{\newacronym}[4][\]{%
7141 \ifx\@glsacronymlists\@empty
7142 \def\@glo@type{\acronymtype}%
7143 \setkeys{glossentry}{##1}%
7144 \DeclareAcronymList{\@glo@type}%
7145 \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7146 \fi
7147 \glskeylisttok{##1}%
7148 \glslabeltok{##2}%
7149 \glsshorttok{##3}%
7150 \glslongtok{##4}%
7151 \newacronymhook
7152 \DescriptionFootnoteNewAcronymDef
7153 }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7154 \@for\@gls@type:=\@glsacronymlists\do{%
7155 \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7156 }%

```

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.

```

7157 \ifglsacrsmallcaps
7158 \renewcommand*\acronymfont[1]{\textsc{##1}}%
7159 \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7160 \else
7161 \ifglsacrsmaller
7162 \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
7163 \fi
7164 \fi

```

Check for package option clash

```

7165 \ifglssacrdua
7166   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7167     can’t both be set}{}%
7168 \fi
7169 }%

```

**nymDisplayStyle** Sets the acronym display style for given glossary with description and dua combination.

```

7170 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7171   \defglssentryfmt[#1]{\glsgenentryfmt}%
7172 }

```

**UANewAcronymDef**

```

7173 \newcommand*{\DescriptionDUANewAcronymDef}{%
7174   \edef\@do@newglossaryentry{%
7175     \noexpand\newglossaryentry{\the\glslabeltok}%
7176     {%
7177       type=\acronymtype,%
7178       name={\the\glslongtok},%
7179       sort={\the\glslongtok},%
7180       text={\the\glslongtok},%
7181       first={\the\glslongtok},%
7182       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7183       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7184       short={\the\glsshorttok},%
7185       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7186       long={\the\glslongtok},%
7187       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7188       symbol={\the\glsshorttok},%
7189       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7190       \the\glskeylisttok
7191     }%
7192   }%
7193   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7194   \let\@org@gls@assign@plural\gls@assign@plural
7195   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7196   \def\gls@assign@firstpl##1##2{%
7197     \@@gls@expand@field{##1}{firstpl}{##2}%
7198   }%
7199   \def\gls@assign@plural##1##2{%
7200     \@@gls@expand@field{##1}{plural}{##2}%
7201   }%
7202   \def\gls@assign@symbolplural##1##2{%
7203     \@@gls@expand@field{##1}{symbolplural}{##2}%
7204   }%
7205   \@do@newglossaryentry
7206   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7207   \let\gls@assign@plural\@org@gls@assign@plural
7208   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7209 }

```

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.

```

7210 \newcommand*{\SetDescriptionDUAACronymStyle}{%
7211   \ifglsmallcaps
7212     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
7213       can't both be set}{}%
7214   \else
7215     \ifglsmaller
7216       \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7217         can't both be set}{}%
7218     \fi
7219   \fi
7220 \renewcommand{\newacronym}[4][\]{%
7221   \ifx\@glsacronymlists\@empty
7222     \def\@gls@type{\acronymtype}%
7223     \setkeys{glossentry}{##1}%
7224     \DeclareAcronymList{\@gls@type}%
7225     \SetDescriptionDUAACronymDisplayStyle{\@gls@type}%
7226   \fi
7227   \glskeylisttok{##1}%
7228   \glslabeltok{##2}%
7229   \glsshorttok{##3}%
7230   \glslongtok{##4}%
7231   \newacronymhook
7232   \DescriptionDUANewAcronymDef
7233 }%
```

Set display.

```

7234 \@for\@gls@type:=\@glsacronymlists\do{%
7235   \SetDescriptionDUAACronymDisplayStyle{\@gls@type}%
7236 }%
7237 }%
```

nymDisplayStyle Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7238 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7239   \defglentryfmt[#1]{%

7240     \ifdefempty\glscustomtext
7241     {%
7242       \ifglused{\glslabel}%
7243     {%
```

Move the inserted text outside of \acronymfont

```

7244       \let\gls@org@insert\glsinsert
7245       \let\glsinsert\@empty
7246       \acronymfont{\glsgenentryfmt}\gls@org@insert
7247     }%
```

```

7248     {%
7249         \glsgenentryfmt
7250         \ifglshassymbol{\glslabel}%
7251         {%
7252             \glsifplural
7253             {%
7254                 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7255             }%
7256             {%
7257                 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7258             }%
7259             \space(\protect\firstacronymfont
7260             {\glscapscase
7261              {\@glo@symbol}
7262              {\@glo@symbol}
7263              {\mfirstucMakeUppercase{\@glo@symbol}}})%
7264         }%
7265     }%
7266 }%
7267 }%
7268 {\glscustomtext\glsinsert}%
7269 }%
7270 }

```

onNewAcronymDef

```

7271 \newcommand*{\DescriptionNewAcronymDef}{%
7272     \edef\@do@newglossaryentry{%
7273         \noexpand\newglossaryentry{\the\glslabeltok}%
7274         {%
7275             type=\acronymtype,%
7276             name={\noexpand
7277                 \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
7278             sort={\the\glsshorttok},%
7279             first={\the\glslongtok},%
7280             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7281             text={\the\glsshorttok},%
7282             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7283             short={\the\glsshorttok},%
7284             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7285             long={\the\glslongtok},%
7286             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7287             symbol={\noexpand\@glo@text},%
7288             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7289             \the\glskeylisttok}%
7290     }%
7291     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7292     \let\@org@gls@assign@plural\gls@assign@plural
7293     \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7294     \def\gls@assign@firstpl##1##2{%

```

```

7295 \@@gls@expand@field{##1}{firstpl}{##2}%
7296 }%
7297 \def\gls@assign@plural##1##2{%
7298 \@@gls@expand@field{##1}{plural}{##2}%
7299 }%
7300 \def\gls@assign@symbolplural##1##2{%
7301 \@@gls@expand@field{##1}{symbolplural}{##2}%
7302 }%
7303 \do@newglossaryentry
7304 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7305 \let\gls@assign@plural\@org@gls@assign@plural
7306 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7307 }

```

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.

```

7308 \newcommand*{\SetDescriptionAcronymStyle}{%
7309 \renewcommand{\newacronym}[4][\]{%
7310 \ifx\@glsacronymlists\@empty
7311 \def\@glo@type{\acronymtype}%
7312 \setkeys{glossentry}{##1}%
7313 \DeclareAcronymList{\@glo@type}%
7314 \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7315 \fi
7316 \glskeylisttok{##1}%
7317 \glslabeltok{##2}%
7318 \glsshorttok{##3}%
7319 \glslongtok{##4}%
7320 \newacronymhook
7321 \DescriptionNewAcronymDef
7322 }%

```

Set display.

```

7323 \@for\@gls@type:=\@glsacronymlists\do{%
7324 \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7325 }%

```

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.

```

7326 \ifglsacrsmallcaps
7327 \renewcommand{\acronymfont}[1]{\textsc{##1}}
7328 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7329 \else
7330 \ifglsacrsmaller
7331 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7332 \fi
7333 \fi
7334 }%

```

`\nymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7335 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7336   \defglentryfmt[#1]{%

7337     \ifdefempty\glscustomtext
7338     {%

      Move the inserted text outside of \acronymfont

7339       \let\gls@org@insert\glsinsert
7340       \let\glsinsert\@empty
7341       \ifglused{\glslabel}%
7342       {%
7343         \acronymfont{\glsgenentryfmt}\gls@org@insert
7344       }%
7345       {%
7346         \firstacronymfont{\glsgenentryfmt}\gls@org@insert
7347         \ifglshaslong{\glslabel}%
7348         {%
7349           \expandafter\protect\expandafter\acrfootnote\expandafter
7350           {\@gls@link@opts}{\@gls@link@label}%
7351           {%
7352             \glsifplural
7353             {\glstentrylongpl{\glslabel}}%
7354             {\glstentrylong{\glslabel}}%
7355           }%
7356         }%

7357       }%
7358     }%
7359   }%
7360   {\glscustomtext\glsinsert}%
7361 }%
7362 }
```

`\teNewAcronymDef`

```

7363 \newcommand*{\FootnoteNewAcronymDef}{%
7364   \edef\@do@newglossaryentry{%
7365     \noexpand\newglossaryentry{\the\glslabeltok}%
7366     {%
7367       type=\acronymtype,%
7368       name={\noexpand\acronymfont{\the\glsshorttok}},%
7369       sort={\the\glsshorttok},%
7370       text={\the\glsshorttok},%
7371       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7372       first={\the\glsshorttok},%
7373       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7374       short={\the\glsshorttok},%
7375       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7376       long={\the\glslongtok},%
```



```

7377     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7378     description={\the\glslongtok},%
7379     descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7380     \the\glskeylisttok
7381   }%
7382 }%
7383 \let\@org@gls@assign@plural\gls@assign@plural
7384 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7385 \let\@org@gls@assign@descplural\gls@assign@descplural
7386 \def\gls@assign@firstpl##1##2{%
7387   \@gls@expand@field{##1}{firstpl}{##2}%
7388 }%
7389 \def\gls@assign@plural##1##2{%
7390   \@gls@expand@field{##1}{plural}{##2}%
7391 }%
7392 \def\gls@assign@descplural##1##2{%
7393   \@gls@expand@field{##1}{descplural}{##2}%
7394 }%
7395 \do@newglossaryentry
7396 \let\gls@assign@plural\@org@gls@assign@plural
7397 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7398 \let\gls@assign@descplural\@org@gls@assign@descplural
7399 }

```

**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.

```

7400 \newcommand*{\SetFootnoteAcronymStyle}{%
7401   \renewcommand{\newacronym}[4][]{%
7402     \ifx\@glsacronymlists\@empty
7403       \def\@glo@type{\acronymtype}%
7404       \setkeys{glossentry}{##1}%
7405       \DeclareAcronymList{\@glo@type}%
7406       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7407     \fi
7408     \glskeylisttok{##1}%
7409     \glslabeltok{##2}%
7410     \glsshorttok{##3}%
7411     \glslongtok{##4}%
7412     \newacronymhook
7413     \FootnoteNewAcronymDef
7414   }%

```

Set display

```

7415   \@for\@gls@type:=\@glsacronymlists\do{%
7416     \SetFootnoteAcronymDisplayStyle{\@gls@type}%
7417   }%

```

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.

```

7418   \ifglsacrsmallcaps

```

```

7419     \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7420     \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7421 \else
7422     \ifglssacrsmaller
7423         \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7424     \fi
7425 \fi

    Check for option clash
7426 \ifglssacrdua
7427     \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7428         can’t both be set}{}%
7429 \fi
7430 }%

```

`\doparenifnotempty` Do a space followed by the argument if the argument doesn’t expand to empty or `\relax`. If argument isn’t empty (or `\relax`), apply the macro to it given in the second argument.

```

7431 \DeclareRobustCommand*{\glsdoparenifnotempty}[2]{%
7432     \protected@edef\gls@tmp{#1}%
7433     \ifdefempty\gls@tmp
7434     {%
7435     {%
7436         \ifx\gls@tmp\gls@default@value
7437         \else
7438             \space (#2{#1})%
7439         \fi
7440     }%
7441 }

```

`\setacronymdisplaystyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

7442 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
7443     \defglsenentryfmt[#1]{%

7444     \ifdefempty\glscustomtext
7445     {%

```

Move the inserted text outside of `\acronymfont`

```

7446         \let\gls@org@insert\glsinsert
7447         \let\glsinsert\@empty
7448         \ifglssused{\glslabel}%
7449         {%
7450             \acronymfont{\glsgenentryfmt}\gls@org@insert
7451         }%
7452         {%
7453             \glsgenentryfmt
7454             \ifglshassymbol{\glslabel}%
7455             {%
7456                 \glsifplural
7457                 {%

```

```

7458         \def\@glo@symbol{\glentrysymbolplural{\glslabel}}}%
7459     }%
7460     {%
7461         \def\@glo@symbol{\glentrysymbol{\glslabel}}}%
7462     }%
7463     \space
7464     (\glscapscase
7465     {\firstacronymfont{\@glo@symbol}}%
7466     {\firstacronymfont{\@glo@symbol}}%
7467     {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})%
7468 }%
7469 {}%
7470 }%
7471 }%
7472 {\glscustomtext\glsinsert}%
7473 }%
7474 }

```

# 11NewAcronymDef

```

7475 \newcommand*{\SmallNewAcronymDef}{%
7476     \edef\@do@newglossaryentry{%
7477         \noexpand\newglossaryentry{\the\glslabeltok}%
7478         {%
7479             type=\acronymtype,%
7480             name={\noexpand\acronymfont{\the\glsshorttok}},%
7481             sort={\the\glsshorttok},%
7482             text={\the\glsshorttok},%
7483             plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7484             first={\the\glslongtok},%
7485             firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7486             short={\the\glsshorttok},%
7487             shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7488             long={\the\glslongtok},%
7489             longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7490             description={\noexpand\@glo@first},%
7491             descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7492             symbol={\the\glsshorttok},%
7493             symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7494             \the\glskeylisttok
7495         }%
7496     }%
7497     \let\@org@gls@assign@firstpl\gls@assign@firstpl
7498     \let\@org@gls@assign@plural\gls@assign@plural
7499     \let\@org@gls@assign@descplural\gls@assign@descplural

```

```

7500 \let\org@gl@assign@symbolplural\gl@assign@symbolplural
7501 \def\gl@assign@firstpl##1##2{%
7502   \@gl@expand@field{##1}{firstpl}{##2}%
7503 }%
7504 \def\gl@assign@plural##1##2{%
7505   \@gl@expand@field{##1}{plural}{##2}%
7506 }%
7507 \def\gl@assign@descplural##1##2{%
7508   \@gl@expand@field{##1}{descplural}{##2}%
7509 }%
7510 \def\gl@assign@symbolplural##1##2{%
7511   \@gl@expand@field{##1}{symbolplural}{##2}%
7512 }%
7513 \do@newglossaryentry
7514 \let\gl@assign@firstpl\org@gl@assign@firstpl
7515 \let\gl@assign@plural\org@gl@assign@plural
7516 \let\gl@assign@descplural\org@gl@assign@descplural
7517 \let\gl@assign@symbolplural\org@gl@assign@symbolplural
7518 }

```

**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.

```

7519 \newcommand*{\SetSmallAcronymStyle}{%
7520   \renewcommand{\newacronym}[4][]{%
7521     \ifx\@gl@acronymlists\@empty
7522       \def\@glo@type{\acronymtype}%
7523       \setkeys{glossentry}{##1}%
7524       \DeclareAcronymList{\@glo@type}%
7525       \SetSmallAcronymDisplayStyle{\@glo@type}%
7526     \fi
7527     \gl@keylisttok{##1}%
7528     \gl@labeltok{##2}%
7529     \gl@shorttok{##3}%
7530     \gl@longtok{##4}%
7531     \newacronymhook
7532     \SmallNewAcronymDef
7533   }%

```

Change the display since first only contains long form.

```

7534 \@for\@gl@type:=\@gl@acronymlists\do{%
7535   \SetSmallAcronymDisplayStyle{\@gl@type}%
7536 }%

```

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.

```

7537 \ifgl@acrsmallcaps
7538   \renewcommand*{\acronymfont}[1]{\textsc{##1}}
7539   \renewcommand*{\acrpluralsuffix}{\gl@supacrpluralsuffix}%
7540 \else
7541   \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}

```

```

7542 \fi
      check for option clash
7543 \ifglsacrdua
7544   \ifglsacrsmallcaps
7545     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7546       can’t both be set}{}%
7547   \else
7548     \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7549       can’t both be set}{}%
7550 \fi
7551 \fi
7552 }%

```

**DUADisplayStyle** Sets the acronym display style for given glossary with dua setting.

```

7553 \newcommand*{\SetDUADisplayStyle}[1]{%
7554   \defglsentryfmt[#1]{\glsentryfmt}%
7555 }

```

**UANewAcronymDef**

```

7556 \newcommand*{\DUANewAcronymDef}{%
7557   \edef\@do@newglossaryentry{%
7558     \noexpand\newglossaryentry{\the\glslabeltok}%
7559     {%
7560       type=\acronymtype,%
7561       name={\the\glsshorttok},%
7562       text={\the\glslongtok},%
7563       first={\the\glslongtok},%
7564       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7565       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7566       short={\the\glsshorttok},%
7567       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7568       long={\the\glslongtok},%
7569       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7570       description={\the\glslongtok},%
7571       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7572       symbol={\the\glsshorttok},%
7573       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7574       \the\glskeylisttok
7575     }%
7576   }%
7577   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7578   \let\@org@gls@assign@plural\gls@assign@plural
7579   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7580   \let\@org@gls@assign@descplural\gls@assign@descplural
7581   \def\gls@assign@firstpl##1##2{%
7582     \@@gls@expand@field{##1}{firstpl}{##2}%
7583   }%
7584   \def\gls@assign@plural##1##2{%
7585     \@@gls@expand@field{##1}{plural}{##2}%

```

```

7586 }%
7587 \def\gls@assign@symbolplural##1##2{%
7588   \@gls@expand@field{##1}{symbolplural}{##2}%
7589 }%
7590 \def\gls@assign@descplural##1##2{%
7591   \@gls@expand@field{##1}{descplural}{##2}%
7592 }%
7593 \do@newglossaryentry
7594 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7595 \let\gls@assign@plural\@org@gls@assign@plural
7596 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7597 \let\gls@assign@descplural\@org@gls@assign@descplural
7598 }

```

\SetDUASStyle Always expand acronyms.

```

7599 \newcommand*{\SetDUASStyle}{%
7600   \renewcommand{\newacronym}[4][]{%
7601     \ifx\@glsacronymlists\@empty
7602       \def\@glo@type{\acronymtype}%
7603       \setkeys{glossentry}{##1}%
7604       \DeclareAcronymList{\@glo@type}%
7605       \SetDUADisplayStyle{\@glo@type}%
7606     \fi
7607     \glskeylisttok{##1}%
7608     \glslabeltok{##2}%
7609     \glsshorttok{##3}%
7610     \glslongtok{##4}%
7611     \newacronymhook
7612     \DUANewAcronymDef
7613   }%
7614   \Set the display
7615   \@for\@gls@type:=\@glsacronymlists\do{%
7616     \SetDUADisplayStyle{\@gls@type}%
7617 }%

```

SetAcronymStyle

```

7618 \newcommand*{\SetAcronymStyle}{%
7619   \SetDefaultAcronymStyle
7620   \ifglsacrdescription
7621     \ifglsacrfootnote
7622       \SetDescriptionFootnoteAcronymStyle
7623     \else
7624       \ifglsacrdua
7625         \SetDescriptionDUAAcronymStyle
7626       \else
7627         \SetDescriptionAcronymStyle
7628       \fi
7629     \fi

```

```

7630 \else
7631   \ifglsacrfootnote
7632     \SetFootnoteAcronymStyle
7633   \else
7634     \ifthenelse{\boolean{glsacrsmallcaps}}\OR
7635       \boolean{glsacrsmaller}}}%
7636   {%
7637     \SetSmallAcronymStyle
7638   }%
7639   {%
7640     \ifglsacrdua
7641       \SetDUASStyle
7642     \fi
7643   }%
7644 \fi
7645 \fi
7646 }

```

Set the acronym style according to the package options

```
7647 \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.

`\setacronymdisplaystyle` Sets the acronym display style.

```

7648 \newcommand*{\SetCustomDisplayStyle}[1]{%
7649   \defglsentryfmt[#1]{\glsentryfmt}%
7650 }

```

`\setacronymfields`

```

7651 \newcommand*{\CustomAcronymFields}{%
7652   name={\the\glsshorttok},%
7653   description={\the\glslongtok},%
7654   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7655   firstplural={\acrfullformat
7656     {\noexpand\glsentrylongpl{\the\glslabeltok}}}%
7657     {\noexpand\glsentryshortpl{\the\glslabeltok}}},%

7658   text={\the\glsshorttok},%
7659   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7660 }

```

`\setnewacronymdef`

```

7661 \newcommand*{\CustomNewAcronymDef}{%
7662   \protected@edef\@do@newglossaryentry{%
7663     \noexpand\newglossaryentry{\the\glslabeltok}%
7664     {%

```

```

7665     type=\acronymtype,%
7666     short={\the\glsshorttok},%
7667     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7668     long={\the\glslongtok},%
7669     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7670     user1={\the\glsshorttok},%
7671     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7672     user3={\the\glslongtok},%
7673     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7674     \CustomAcronymFields,%
7675     \the\glskeylisttok
7676   }%
7677 }%
7678 \@do@newglossaryentry
7679 }

```

\SetCustomStyle

```

7680 \newcommand*{\SetCustomStyle}{%
7681   \renewcommand{\newacronym}[4][]{%
7682     \ifx\@glsacronymlists\@empty
7683       \def\@gls@type{\acronymtype}%
7684       \setkeys{glossentry}{##1}%
7685       \DeclareAcronymList{\@gls@type}%
7686       \SetCustomDisplayStyle{\@gls@type}%
7687     \fi
7688     \glskeylisttok{##1}%
7689     \glslabeltok{##2}%
7690     \glsshorttok{##3}%
7691     \glslongtok{##4}%
7692     \newacronymhook
7693     \CustomNewAcronymDef
7694   }%
7695   \SetCustomDisplayStyle{\@gls@type}%
7696   \SetCustomDisplayStyle{\@gls@type}%
7697 }%
7698 }

```

## 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.

```
7699 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the nolist option is used:

```
7700 \@gls@loadlist
```



The styles that use the longtable environment. These are not loaded if the nolong package option is used.

```
7701 \@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.

```
7702 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7703 \@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.

```
7704 \ifx\@glossary@default@style\relax
```

```
7705 \else
```

```
7706   \setglossarystyle{\@glossary@default@style}
```

```
7707 \fi
```

## 1.20 Debugging Commands

```
\showgloparent \showgloparent{\label{}}
```

```
7708 \newcommand*{\showgloparent}[1]{%
```

```
7709   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7710 }
```

```
\showglolevel \showglolevel{\label{}}
```

```
7711 \newcommand*{\showglolevel}[1]{%
```

```
7712   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7713 }
```

```
\showglotext \showglotext{\label{}}
```

```
7714 \newcommand*{\showglotext}[1]{%
```

```
7715   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7716 }
```

```
\showgloplural \showgloplural{\label{}}
```

```

7717 \newcommand*{\showgloplural}[1]{%
7718   \expandafter\show\csname glo@glstetoklabel{#1}@plural\endcsname
7719 }

```

\showglofirst    \showglofirst{<label>}

```

7720 \newcommand*{\showglofirst}[1]{%
7721   \expandafter\show\csname glo@glstetoklabel{#1}@first\endcsname
7722 }

```

\showglofirstpl    \showglofirstpl{<label>}

```

7723 \newcommand*{\showglofirstpl}[1]{%
7724   \expandafter\show\csname glo@glstetoklabel{#1}@firstpl\endcsname
7725 }

```

\showgloftype    \showgloftype{<label>}

```

7726 \newcommand*{\showgloftype}[1]{%
7727   \expandafter\show\csname glo@glstetoklabel{#1}@type\endcsname
7728 }

```

\showglocounter    \showglocounter{<label>}

```

7729 \newcommand*{\showglocounter}[1]{%
7730   \expandafter\show\csname glo@glstetoklabel{#1}@counter\endcsname
7731 }

```

\showglouser    \showglouser{<label>}

```

7732 \newcommand*{\showglouser}[1]{%
7733   \expandafter\show\csname glo@glstetoklabel{#1}@user\endcsname
7734 }

```

\showglouserii    \showglouserii{<label>}

```

7735 \newcommand*{\showglouserii}[1]{%
7736   \expandafter\show\csname glo@glstdetoklabel{#1}@userii\endcsname
7737 }

```

\showglouseriii    \showglouseriii{<label>}

```

7738 \newcommand*{\showglouseriii}[1]{%
7739   \expandafter\show\csname glo@glstdetoklabel{#1}@useriii\endcsname
7740 }

```

\showglouseriv    \showglouseriv{<label>}

```

7741 \newcommand*{\showglouseriv}[1]{%
7742   \expandafter\show\csname glo@glstdetoklabel{#1}@useriv\endcsname
7743 }

```

\showglouserv    \showglouserv{<label>}

```

7744 \newcommand*{\showglouserv}[1]{%
7745   \expandafter\show\csname glo@glstdetoklabel{#1}@userv\endcsname
7746 }

```

\showglouservi    \showglouservi{<label>}

```

7747 \newcommand*{\showglouservi}[1]{%
7748   \expandafter\show\csname glo@glstdetoklabel{#1}@uservi\endcsname
7749 }

```

\showgloname    \showgloname{<label>}

```

7750 \newcommand*{\showgloname}[1]{%
7751   \expandafter\show\csname glo@glstdetoklabel{#1}@name\endcsname
7752 }

```

\showglodesc    \showglodesc{<label>}

```

7753 \newcommand*{\showglodesc}[1]{%
7754   \expandafter\show\csname glo@\glsdetoklabel{#1}@desc\endcsname
7755 }

```

showglodescplural `\showglodescplural{<label>}`

```

7756 \newcommand*{\showglodescplural}[1]{%
7757   \expandafter\show\csname glo@\glsdetoklabel{#1}@descplural\endcsname
7758 }

```

\showglosort `\showglosort{<label>}`

```

7759 \newcommand*{\showglosort}[1]{%
7760   \expandafter\show\csname glo@\glsdetoklabel{#1}@sort\endcsname
7761 }

```

\showglosymbol `\showglosymbol{<label>}`

```

7762 \newcommand*{\showglosymbol}[1]{%
7763   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbol\endcsname
7764 }

```

showglosymbolplural `\showglosymbolplural{<label>}`

```

7765 \newcommand*{\showglosymbolplural}[1]{%
7766   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolplural\endcsname
7767 }

```

\showgloshort `\showgloshort{<label>}`

```

7768 \newcommand*{\showgloshort}[1]{%
7769   \expandafter\show\csname glo@\glsdetoklabel{#1}@short\endcsname
7770 }

```

\showglolong `\showglolong{<label>}`

```

7771 \newcommand*{\showglolong}[1]{%
7772   \expandafter\show\csname glo@\glsdetoklabel{#1}@long\endcsname
7773 }

```

\showgloindex    \showgloindex{<label>}

```

7774 \newcommand*{\showgloindex}[1]{%
7775   \expandafter\show\csname glo@\glsdetoklabel{#1}@index\endcsname
7776 }

```

\showgloflag    \showgloflag{<label>}

```

7777 \newcommand*{\showgloflag}[1]{%
7778   \expandafter\show\csname ifglo@\glsdetoklabel{#1}@flag\endcsname
7779 }

```

\showgloloclist    \showgloloclist{<label>}

```

7780 \newcommand*{\showgloloclist}[1]{%
7781   \expandafter\show\csname glo@\glsdetoklabel{#1}@loclist\endcsname
7782 }

```

\showglofield    \showglofield{<label>}{<field>}

```

7783 \newcommand*{\showglofield}[2]{%
7784   \csshow{glo@\glsdetoklabel{#1}@#2}%
7785 }

```

showacronymlists    \showacronymlists

Show list of glossaries that have been flagged as a list of acronyms.

```

7786 \newcommand*{\showacronymlists}{%
7787   \show\@glsacronymlists
7788 }

```

\showglossaries    \showglossaries

Show list of defined glossaries.

```

7789 \newcommand*{\showglossaries}{%

```

```

7790 \show\@glo@types
7791 }

```

`\showglossaryin` `\showglossaryin{<glossary-label>}`

Show the ‘in’ extension for the given glossary.

```

7792 \newcommand*{\showglossaryin}[1]{%
7793 \expandafter\show\csname @glotype@#1@in\endcsname
7794 }

```

`\showglossaryout` `\showglossaryout{<glossary-label>}`

Show the ‘out’ extension for the given glossary.

```

7795 \newcommand*{\showglossaryout}[1]{%
7796 \expandafter\show\csname @glotype@#1@out\endcsname
7797 }

```

`showglossarytitle` `\showglossarytitle{<glossary-label>}`

Show the title for the given glossary.

```

7798 \newcommand*{\showglossarytitle}[1]{%
7799 \expandafter\show\csname @glotype@#1@title\endcsname
7800 }

```

`wglossarycounter` `\showglossarycounter{<glossary-label>}`

Show the counter for the given glossary.

```

7801 \newcommand*{\showglossarycounter}[1]{%
7802 \expandafter\show\csname @glotype@#1@counter\endcsname
7803 }

```

`wglossaryentries` `\showglossaryentries{<glossary-label>}`

Show the list of entry labels for the given glossary.

```

7804 \newcommand*{\showglossaryentries}[1]{%
7805 \expandafter\show\csname glolist@#1\endcsname
7806 }

```

## 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.

```
7807 \csname ifglscpatible-2.07\endcsname
7808   \RequirePackage{glossaries-compatible-207}
7809 \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.

```
7810 \NeedsTeXFormat{LaTeX2e}
```

```
7811 \ProvidesPackage{glossaries-prefix}[2017/06/11 v4.30 (NLCT)]
```

Pass all options to glossaries:

```
7812 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7813 \ProcessOptions
```

Load glossaries:

```
7814 \RequirePackage{glossaries}
```

Add the new keys:

```
7815 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7816 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7817 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7818 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\gls@keymap`:

```
7819 \appto\@gls@keymap{,%
```

```
7820   {prefixfirst}{prefixfirst},%
```

```
7821   {prefixfirstplural}{prefixfirstplural},%
```

```
7822   {prefix}{prefix},%
```

```
7823   {prefixplural}{prefixplural}}%
```

```
7824 }
```

Set the default values:

```
7825 \appto\@newglossaryentryprehook{%
```

```
7826   \def\@glo@entryprefix{}}%
```

```
7827   \def\@glo@entryprefixplural{}}%
```

```
7828   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7829   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7830 }
```

Set the assignment code:

```
7831 \appto\@newglossaryentryposthook{%
```

```
7832   \gls@assign@field{ }\@glo@label{prefix}{\@glo@entryprefix}}%
```

```
7833   \gls@assign@field{ }\@glo@label{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7834 \expandafter\gls@assign@field\expandafter
```

```
7835   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
7836   {\@glo@entryprefixfirst}}%
```



If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7837 \expandafter\gls@assign@field\expandafter
7838 {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7839 {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7840 }

```

Define commands to access these fields:

entryprefixfirst

```

7841 \newcommand*\glsentryprefixfirst[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7842 \newcommand*\glsentryprefixfirstplural[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7843 \newcommand*\glsentryprefix[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7844 \newcommand*\glsentryprefixplural[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7845 \newrobustcmd*\Glsentryprefixfirst[1]{%
7846 \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7847 \xmakefirstuc\@glo@text
7848 }

```

entryfirstplural

```

7849 \newrobustcmd*\Glsentryprefixfirstplural[1]{%
7850 \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7851 \xmakefirstuc\@glo@text
7852 }

```

\Glsentryprefix

```

7853 \newrobustcmd*\Glsentryprefix[1]{%
7854 \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7855 \xmakefirstuc\@glo@text
7856 }

```

entryprefixplural

```

7857 \newrobustcmd*\Glsentryprefixplural[1]{%
7858 \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7859 \xmakefirstuc\@glo@text
7860 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7861 \newcommand*{\ifglshasprefix}[3]{%
7862   \ifcempty{glo@#1@prefix}%
7863   {#3}%
7864   {#2}%
7865 }
```

hasprefixplural

```
7866 \newcommand*{\ifglshasprefixplural}[3]{%
7867   \ifcempty{glo@#1@prefixplural}%
7868   {#3}%
7869   {#2}%
7870 }
```

shasprefixfirst

```
7871 \newcommand*{\ifglshasprefixfirst}[3]{%
7872   \ifcempty{glo@#1@prefixfirst}%
7873   {#3}%
7874   {#2}%
7875 }
```

efixfirstplural

```
7876 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7877   \ifcempty{glo@#1@prefixfirstplural}%
7878   {#3}%
7879   {#2}%
7880 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7881 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7882 \newcommand*{\@pgls}[2][ ]{%
7883   \new@ifnextchar[%
7884   {\@pgls@{#1}{#2}}%
7885   {\@pgls@{#1}{#2}[ ]}%
7886 }
```

\@pgls@ Read in the final optional argument:

```
7887 \def\@pgls@#1#2[#3]{%
7888   \glsdoifexists{#2}%
7889   {%
7890     \ifglsused{#2}%
7891     {%
7892       \glstryprefix{#2}%
7893     }%

```

```

7894     {%
7895         \glsentryprefixfirst{#2}%
7896     }%
7897     \@gls@{#1}{#2}[#3]%
7898 }%
7899 }

```

Similarly for the plural version:

```

\pglsp1
7900 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7901 \newcommand*{\@pglsp1}[2][{}]{%
7902     \new@ifnextchar[%
7903         {\@pglsp1@{#1}{#2}}%
7904         {\@pglsp1@{#1}{#2}[]}%
7905 }

```

\@pglsp1@ Read in the final optional argument:

```

7906 \def\@pglsp1@#1#2[#3]{%
7907     \glsdoifexists{#2}%
7908     {%
7909         \ifglsused{#2}%
7910         {%
7911             \glsentryprefixplural{#2}%
7912         }%
7913         {%
7914             \glsentryprefixfirstplural{#2}%
7915         }%
7916         \@glspl@{#1}{#2}[#3]%
7917     }%
7918 }

```

Now for the first letter upper case versions:

```

\Pgls
7919 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7920 \newcommand*{\@Pgls}[2][{}]{%
7921     \new@ifnextchar[%
7922         {\@Pgls@{#1}{#2}}%
7923         {\@Pgls@{#1}{#2}[]}%
7924 }

```

\@Pgls@ Read in the final optional argument:

```

7925 \def\@Pgls@#1#2[#3]{%

```

```

7926 \glsdoifexists{#2}%
7927 {%
7928   \ifglsused{#2}%
7929   {%
7930     \ifglshasprefix{#2}%
7931     {%
7932       \Glsentryprefix{#2}%
7933       \@gls@{#1}{#2}[#3]%
7934     }%
7935     {\@Gls@{#1}{#2}[#3]}%
7936   }%
7937   {%
7938     \ifglshasprefixfirst{#2}%
7939     {%
7940       \Glsentryprefixfirst{#2}%
7941       \@gls@{#1}{#2}[#3]%
7942     }%
7943     {\@Gls@{#1}{#2}[#3]}%
7944   }%
7945 }%
7946 }

```

Similarly for the plural version:

```

\Pglspl
7947 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

7948 \newcommand*{\@Pglspl}[2] [] {%
7949   \new@ifnextchar[%
7950   {\@Pglspl@{#1}{#2}}%
7951   {\@Pglspl@{#1}{#2} []}%
7952 }

```

\@Pglspl@ Read in the final optional argument:

```

7953 \def\@Pglspl@#1#2[#3] {%
7954   \glsdoifexists{#2}%
7955   {%
7956     \ifglsused{#2}%
7957     {%
7958       \ifglshasprefixplural{#2}%
7959       {%
7960         \Glsentryprefixplural{#2}%
7961         \@glspl@{#1}{#2}[#3]%
7962       }%
7963       {\@Glspl@{#1}{#2}[#3]}%
7964     }%
7965     {%
7966       \ifglshasprefixfirstplural{#2}%

```

```

7967      {%
7968      \Glsentryprefixfirstplural{#2}%
7969      \@glsp1@{#1}{#2}[#3]%
7970      }%
7971      {\@Glspl@{#1}{#2}[#3]}%
7972      }%
7973  }%
7974 }

```

Finally the all upper case versions:

\PGLS

```

7975 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

7976 \newcommand*{\@PGLS}[2][{}]{%
7977   \new@ifnextchar[%
7978   {\@PGLS@{#1}{#2}}%
7979   {\@PGLS@{#1}{#2}[]}%
7980 }

```

\@PGLS@ Read in the final optional argument:

```

7981 \def\@PGLS@#1#2[#3]{%
7982   \glsdoifexists{#2}%
7983   {%
7984     \ifglsused{#2}%
7985     {%
7986       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7987     }%
7988     {%
7989       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7990     }%
7991     \@GLS@{#1}{#2}[#3]%
7992   }%
7993 }

```

Plural version:

\PGLSp1

```

7994 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

7995 \newcommand*{\@PGLSp1}[2][{}]{%
7996   \new@ifnextchar[%
7997   {\@PGLSp1@{#1}{#2}}%
7998   {\@PGLSp1@{#1}{#2}[]}%
7999 }

```

\@PGLSp1@ Read in the final optional argument:

```
8000 \def\@PGLSp1@#1#2[#3]{%
8001   \glsdoifexists{#2}%
8002   {%
8003     \ifglsused{#2}%
8004     {%
8005       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%
8006     }%
8007     {%
8008       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%
8009     }%
8010     \@GLSp1@{#1}{#2}[#3]%
8011   }%
8012 }
```

## 3 Glossary Styles

### 3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
8013 \ProvidesPackage{glossary-hypernav}[2017/06/11 v4.30 (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`

```
8014 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
8015   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
8016   \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

`glsnavhyperlinkname` Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
8017 \newcommand*{\glsnavhyperlinkname}[2]{\glsn:#1@#2}
```

```
\glsnavhypertarget[<type>]{<label>}{<text>}
```

This command makes `<text>` a hypertarget for the glossary group whose label is given by `<label>` in the glossary given by `<type>`. If `<type>` is omitted, `\@glo@type` is used which is set by `\printglossary` to the current glossary label.

`glsnavhypertarget`

```
8018 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
8019   \protected@write\auxout{}{\string\gls@hypergroup{#1}{#2}}%
  Add the target.
8020   \@glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
8021 \expandafter\let
8022 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
8023 \@for\@gls@elem:=\@gls@list\do{%
8024 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
8025 \if@endfor
8026 \else
```

This group was not included in the list, so issue a warning.

```
8027 \GlossariesWarningNoLine{Navigation panel
8028 for glossary type ‘#1’^^Jmissing group ‘#2’}%
8029 \gdef\gls@hypergroup@rerun{%
8030 \GlossariesWarningNoLine{Navigation panel
8031 has changed. Rerun LaTeX}}%
8032 \fi
8033 }
```

`hypergroup@rerun` Give a warning at the end if re-run required

```
8034 \let\gls@hypergroup@rerun\relax
8035 \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.

```
8036 \newcommand*{\@gls@hypergroup}[2]{%
8037 \@ifundefined{\@gls@hypergroup@list@#1}{%
8038 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}%
8039 }{%
8040 \expandafter\let\expandafter\@gls@tmp
8041 \csname @gls@hypergroup@list@#1\endcsname
8042 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
8043 \@gls@tmp,#2}%
8044 }%
8045 }
```

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. Note that 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. Now for the whole navigation bit:

`\glsnavigation`

```
8046 \newcommand*{\glsnavigation}{%
8047 \def\@gls@between{}%
8048 \ifcsundef{\@gls@hypergroup@list@\@glo@type}%

```



```

8049 {%
8050   \def\@gls@list{}%
8051 }%
8052 {%
8053   \expandafter\let\expandafter\@gls@list
8054     \csname @gls@hypergroup@list@\@glo@type\endcsname
8055 }%
8056 \@for\@gls@tmp:=\@gls@list\do{%
8057   \@gls@between

8058   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8059   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8060   \let\@gls@between\glshypernavsep
8061 }%
8062 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8063 \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`

```

8064 \newcommand*{\glssymbolnav}{%
8065   \glsnavhyperlink{glsymbols}{\glsgetgrouptitle{glsymbols}}%
8066   \glshypernavsep
8067   \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
8068   \glshypernavsep
8069 }

```

## 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.

```
8070 \ProvidesPackage{glossary-inline}[2017/06/11 v4.30 (NLCT)]
```

`inline` Define the inline style.

```

8071 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then changed by
    \glossentry)
8072   \renewenvironment{theglossary}%
8073     {%
8074       \def\gls@inlinesep{}%
8075       \def\gls@inlinesubsep{}%
8076       \def\gls@inlinepostchild{}%
8077     }%
8078     {\glspostinline}%

```

No header:

```
8079 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
8080 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```
8081 \renewcommand{\glossentry}[2]{%
8082   \glsinlinedopostchild
8083   \gls@inlinesep
8084   \glsentryitem{##1}%
8085   \glsinlinenameformat{##1}{%
8086     \glossentryname{##1}%
8087   }%
8088   \ifglsdescsuppressed{##1}%
8089   {%
8090     \glsinlineemptydescformat
8091     {%
8092       \glossentrysymbol{##1}%
8093     }%
8094     {%
8095       ##2%
8096     }%
8097   }%
8098   {%
8099     \ifglshasdesc{##1}%
8100     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8101     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8102   }%
8103   \ifglshaschildren{##1}%
8104   {%
8105     \glsresetsubentrycounter
8106     \glsinlineparentchildseparator
8107     \def\gls@inlinesubsep{}%
8108     \def\gls@inlinepostchild{\glsinlinepostchild}%
8109   }%
8110   {}%
8111   \def\gls@inlinesep{\glsinlineseparator}%
8112 }%
```

Sub-entries display description:

```
8113 \renewcommand{\subglossentry}[3]{%
8114   \gls@inlinesubsep%
8115   \glsinlinesubnameformat{##2}{%
8116     \glossentryname{##2}%
8117   }%
8118   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8119   \def\gls@inlinesubsep{\glsinlinesubseparator}%
8120 }%
```

Nothing special between groups:

```
8121 \renewcommand*{\glsgroupskip}{}%  
8122 }
```

linedopostchild

```
8123 \newcommand*{\glsinlinedopostchild}{%  
8124     \gls@inlinepostchild  
8125     \def\gls@inlinepostchild{}%  
8126 }
```

inlineseparator Separator to use between entries.

```
8127 \newcommand*{\glsinlineseparator}{;\space}
```

inlinesubseparator Separator to use between sub-entries.

```
8128 \newcommand*{\glsinlinesubseparator}{,\space}
```

parentchildseparator Separator to use between parent and children.

```
8129 \newcommand*{\glsinlineparentchildseparator}{:\space}
```

inlinepostchild Hook to use between child and next entry

```
8130 \newcommand*{\glsinlinepostchild}{}
```

\glspostinline Terminator for inline glossary.

```
8131 \newcommand*{\glspostinline}{\glspostdescription\space}
```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```
8132 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

inlinedescformat Formats the entry's description, symbol and location list:

```
8133 \newcommand*{\glsinlinedescformat}[3]{\space#1}
```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```
8134 \newcommand*{\glsinlineemptydescformat}[2]{}
```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```
8135 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```
8136 \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.

```
8137 \ProvidesPackage{glossary-list}[2017/06/11 v4.30 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8138 \providecommand{\indexspace}{%
8139   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8140 }

```

`groupheaderfmt` Provide a way of adjusting the format of the group headings.

```

8141 \newcommand*{\glslistgroupheaderfmt}[1]{#1}

```

`navigationitem` 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.

```

8142 \newcommand*{\glslistnavigationitem}[1]{\item[#1]}

```

`list` The list glossary style uses the description environment. The group separator `\glsgroupskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the glossaries package.

```

8143 \newglossarystyle{list}{%

```

Use description environment:

```

8144   \renewenvironment{theglossary}%
8145     {\begin{description}}{\end{description}}%

```

No header at the start of the environment:

```

8146   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8147   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries start a new item in the list:

```

8148   \renewcommand*{\glossentry}[2]{%
8149     \item[\glsentryitem{##1}%
8150       \glstarget{##1}{\glossentryname{##1}}]
8151     \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries continue on the same line:

```

8152   \renewcommand*{\subglossentry}[3]{%
8153     \glssubentryitem{##2}%
8154     \glstarget{##2}{\strut}\space
8155     \glossentrydesc{##2}\glspostdescription\space ##3.}%

```

Add vertical space between groups:

```

8156   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8157 }

```

**listgroup** The listgroup style is like the list style, but the glossary groups have headings.

```
8158 \newglossarystyle{listgroup}{%  
    Base it on the list style:  
8159 \setglossarystyle{list}%  
    Each group has a heading:  
8160 \renewcommand*{\glsgroupheading}[1]{%  
8161 \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.

```
8162 \newglossarystyle{listhypergroup}{%  
    Base it on the list style:  
8163 \setglossarystyle{list}%  
    Add navigation links at the start of the environment.  
8164 \renewcommand*{\glossaryheader}{%  
8165 \glslistnavigationitem{\glsnavigation}}%  
    Each group has a heading with a hypertarget:  
8166 \renewcommand*{\glsgroupheading}[1]{%  
8167 \item[\glslistgroupheaderfmt  
8168 {\glsnavigationhypertarget{##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.

```
8169 \newglossarystyle{altlist}{%  
    Base it on the list style:  
8170 \setglossarystyle{list}%  
    Main (level 0) entries start a new item in the list with a line break after the entry name:  
8171 \renewcommand*{\glossentry}[2]{%  
8172 \item[\glssentryitem{##1}%  
8173 \glstarget{##1}{\glossentryname{##1}}}%  
    Version 3.04 changed \newline to the following paragraph break stuff (thanks to Daniel Geb-  
    hardt for supplying the fix) to prevent a page break occurring at this point.  
8174 \mbox{}\par\nobreak\@afterheading  
8175 \glossentrydesc{##1}\glspostdescription\space ##2}%  
    Sub-entries start a new paragraph:  
8176 \renewcommand{\subglossentry}[3]{%  
8177 \par  
8178 \glssubentryitem{##2}%  
8179 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%  
8180 }
```

**altlistgroup** The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8181 \newglossarystyle{altlistgroup}{%
    Base it on the altlist style:
8182   \setglossarystyle{altlist}%
    Each group has a heading:
8183   \renewcommand*{\glsgroupheading}[1]{%
8184     \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.

```
8185 \newglossarystyle{altlisthypergroup}{%
    Base it on the altlist style:
8186   \setglossarystyle{altlist}%
    Add navigation links at the start of the environment.
8187   \renewcommand*{\glossaryheader}{%
8188     \glslistnavigationitem{\glsnavigation}}%
    Each group has a heading with a hypertarget:
8189   \renewcommand*{\glsgroupheading}[1]{%
8190     \item[\glslistgroupheaderfmt
8191       {\glsnavhypertarget{##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.

```
8192 \newglossarystyle{listdotted}{%
    Base it on the list style:
8193   \setglossarystyle{list}%
    Each main (level 0) entry starts a new item:
8194   \renewcommand*{\glossentry}[2]{%
8195     \item[]\makebox[\glslistdottedwidth][l]{%
8196       \glssentryitem{##1}%
8197       \glstarget{##1}{\glossentryname{##1}}%
8198       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%
    Sub entries have the same format as main entries:
8199   \renewcommand*{\subglossentry}[3]{%
8200     \item[]\makebox[\glslistdottedwidth][l]{%
8201       \glssubentryitem{##2}%
8202       \glstarget{##2}{\glossentryname{##2}}%
8203       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%
8204 }
```

listdottedwidth

```
8205 \newlength\glslistdottedwidth
8206 \setlength{\glslistdottedwidth}{.5\hsize}
```

sublistdotted This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8207 \newglossarystyle{sublistdotted}{%
    Base it on the listdotted style:
8208 \setglossarystyle{listdotted}%
    Main (level 0) entries just display the name:
8209 \renewcommand*{\glossentry}[2]{%
8210 \item[\glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
8211 }
```

### 3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8212 \ProvidesPackage{glossary-long}[2017/06/11 v4.30 (NLCT)]
```

Requires the package:

```
8213 \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`.)

```
8214 \@ifundefined{glsdescwidth}{%
8215 \newlength\glsdescwidth
8216 \setlength{\glsdescwidth}{0.6\hsize}
8217 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
8218 \@ifundefined{glspagelistwidth}{%
8219 \newlength\glspagelistwidth
8220 \setlength{\glspagelistwidth}{0.1\hsize}
8221 }{}
```

`long` The long glossary style command which uses the `longtable` environment:

```
8222 \newglossarystyle{long}{%
    Use longtable with two columns:
8223 \renewenvironment{theglossary}%
8224 {\begin{longtable}\lp{\glsdescwidth}}%
8225 {\end{longtable}}%
    Do nothing at the start of the environment:
8226 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8227 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8228 \renewcommand{\glossentry}[2]{%
8229   \glstarget{##1}\glstarget{##1}{\glossentryname{##1}} &
8230   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8231 }%
```

Sub entries displayed on the following row without the name:

```
8232 \renewcommand{\subglossentry}[3]{%
8233   &
8234   \glssubentryitem{##2}%
8235   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8236   ##3\tabularnewline
8237 }%
```

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>)

```
8238 \ifglsgroupskip
8239 \renewcommand*{\glsgroupskip}{}%
8240 \else
8241 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8242 \fi
8243 }
```

**longborder** The `longborder` style is like the above, but with horizontal and vertical lines:

```
8244 \newglossarystyle{longborder}{%
```

Base it on the `glostylelong` style:

```
8245 \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8246 \renewenvironment{theglossary}{%
8247   \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8248 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8249 }
```

**longheader** The `longheader` style is like the long style but with a header:

```
8250 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8251 \setglossarystyle{long}%
```

Set the table's header:

```
8252 \renewcommand*{\glossaryheader}{%
8253   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8254 }
```



ongheaderborder The longheaderborder style is like the long style but with a header and border:

```
8255 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
8256 \setglossarystyle{longborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8257 \renewcommand*{\glossaryheader}{%
8258 \hline\bfseries \entryname & \bfseries
8259 \descriptionname\tabularnewline\hline
8260 \endhead
8261 \hline\endfoot}%
8262 }
```

long3col The long3col style is like long but with 3 columns

```
8263 \newglossarystyle{long3col}{%
```

Use a longtable with 3 columns:

```
8264 \renewenvironment{theglossary}%
8265 {\begin{longtable}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
8266 {\end{longtable}}%
```

No table header:

```
8267 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8268 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8269 \renewcommand{\glossentry}[2]{%
8270 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8271 \glossentrydesc{##1} & ##2\tabularnewline
8272 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8273 \renewcommand{\subglossentry}[3]{%
8274 &
8275 \glssubentryitem{##2}%
8276 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8277 ##3\tabularnewline
8278 }%
```

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>)

```
8279 \ifglsnogroupskip
8280 \renewcommand*{\glsgroupskip}{}%
8281 \else
8282 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8283 \fi
8284 }
```

`long3colborder` The `long3colborder` style is like the `long3col` style but with a border:

```
8285 \newglossarystyle{long3colborder}{%
      Base it on the glostylelong3col style:
8286   \setglossarystyle{long3col}%
      Use a longtable with 3 columns with vertical lines around them:
8287   \renewenvironment{theglossary}%
8288     {\begin{longtable}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8289     {\end{longtable}}%
      Place horizontal lines at the head and foot of the table:
8290   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8291 }
```

`long3colheader` The `long3colheader` style is like `long3col` but with a header row:

```
8292 \newglossarystyle{long3colheader}{%
      Base it on the glostylelong3col style:
8293   \setglossarystyle{long3col}%
      Set the table's header:
8294   \renewcommand*{\glossaryheader}{%
8295     \bfseries\entryname&\bfseries\descriptionname&
8296     \bfseries\pagelistname\tabularnewline\endhead}%
8297 }
```

`colheaderborder` The `long3colheaderborder` style is like the above but with a border

```
8298 \newglossarystyle{long3colheaderborder}{%
      Base it on the glostylelong3colborder style:
8299   \setglossarystyle{long3colborder}%
      Set the table's header and add horizontal line at table's foot:
8300   \renewcommand*{\glossaryheader}{%
8301     \hline
8302     \bfseries\entryname&\bfseries\descriptionname&
8303     \bfseries\pagelistname\tabularnewline\hline\endhead
8304     \hline\endfoot}%
8305 }
```

`long4col` The `long4col` style has four columns where the third column contains the value of the associated symbol key.

```
8306 \newglossarystyle{long4col}{%
      Use a longtable with 4 columns:
8307   \renewenvironment{theglossary}%
8308     {\begin{longtable}{l|l|l|l}}%
8309     {\end{longtable}}%
      No table header:
8310   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
8311 \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):

```
8312 \renewcommand{\glossentry}[2]{%
8313   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8314   \glossentrydesc{##1} &
8315   \glossentrysymbol{##1} &
8316   ##2\tabularnewline
8317 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8318 \renewcommand{\subglossentry}[3]{%
8319   &
8320   \glssubentryitem{##2}%
8321   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8322   \glossentrysymbol{##2} & ##3\tabularnewline
8323 }%
```

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>)

```
8324 \ifglsnogroupskip
8325 \renewcommand*{\glsgroupskip}{}%
8326 \else
8327 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
8328 \fi
8329 }
```

**long4colheader** The long4colheader style is like long4col but with a header row.

```
8330 \newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
8331 \setglossarystyle{long4col}{%
```

Table has a header:

```
8332 \renewcommand*{\glossaryheader}{%
8333   \bfseries\entryname&\bfseries\descriptionname&
8334   \bfseries \symbolname&
8335   \bfseries\pagelistname\tabularnewline\endhead}%
8336 }
```

**long4colborder** The long4colborder style is like long4col but with a border.

```
8337 \newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```
8338 \setglossarystyle{long4col}{%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8339 \renewenvironment{theglossary}{%
```

```
8340    {\begin{longtable}{|l|l|l|l|}}%
```

```
8341    {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8342    \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
8343 }
```

**colheaderborder** The long4colheaderborder style is like the above but with a border.

```
8344 \newglossarystyle{long4colheaderborder}{%
```

Base it on the glostylelong4col style:

```
8345    \setglossarystyle{long4col}%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8346    \renewenvironment{theglossary}%
```

```
8347    {\begin{longtable}{|l|l|l|l|}}%
```

```
8348    {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8349    \renewcommand*{\glossaryheader}{%
```

```
8350    \hline\bfseries\entryname&\bfseries\descriptionname&
```

```
8351    \bfseries \symbolname&
```

```
8352    \bfseries\pagelistname\tabularnewline\hline\endhead
```

```
8353    \hline\endfoot}%
```

```
8354 }
```

**altlong4col** The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```
8355 \newglossarystyle{altlong4col}{%
```

Base it on the glostylelong4col style:

```
8356    \setglossarystyle{long4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8357    \renewenvironment{theglossary}%
```

```
8358    {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
```

```
8359    {\end{longtable}}%
```

```
8360 }
```

**altlong4colheader** The altlong4colheader style is like altlong4col but with a header row.

```
8361 \newglossarystyle{altlong4colheader}{%
```

Base it on the glostylelong4colheader style:

```
8362    \setglossarystyle{long4colheader}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8363    \renewenvironment{theglossary}%
```

```
8364    {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
```

```
8365    {\end{longtable}}%
```

```
8366 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
8367 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
8368 \setglossarystyle{long4colborder}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8369 \renewenvironment{theglossary}{%
```

```
8370 {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8371 {\end{longtable}}%
```

```
8372 }
```

`colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
8373 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
8374 \setglossarystyle{long4colheaderborder}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8375 \renewenvironment{theglossary}{%
```

```
8376 {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8377 {\end{longtable}}%
```

```
8378 }
```

### 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>

```
8379 \ProvidesPackage{glossary-longbooktabs}[2017/06/11 v4.30 (NLCT)]
```

Requires `booktabs` package:

```
8380 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8381 \RequirePackage{glossary-long}
```

```
8382 \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.

```
8383 \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.

```
8384 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8385 \setglossarystyle{long}%
```

Add a header with rules.

```
8386 \renewcommand*{\glossaryheader}{%
8387   \toprule \bfseries \entryname & \bfseries
8388   \descriptionname\tabularnewline\midrule\endhead
8389   \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.

```
8390 \ifglsgroupskip
8391   \renewcommand*{\glsgroupskip}{}%
8392 \else
8393   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8394 \fi
8395 }
```

ng3col-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.

```
8396 \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.

```
8397 \glspatchLToutput
```

Use the long3col style as a base.

```
8398 \setglossarystyle{long3col}%
```

Add a header with rules.

```
8399 \renewcommand*{\glossaryheader}{%
8400   \toprule \bfseries \entryname &
8401   \bfseries \descriptionname &
8402   \bfseries \pagelistname
8403   \tabularnewline\midrule\endhead
8404   \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.

```
8405 \ifglsgroupskip
8406   \renewcommand*{\glsgroupskip}{}%
8407 \else
8408   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8409 \fi
8410 }
```

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.

```
8411 \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.

```
8412 \glspatchLToutput
```

Use the long4col style as a base.

```
8413 \setglossarystyle{long4col}%
```

Add a header with rules.

```
8414 \renewcommand*{\glossaryheader}{%
8415   \toprule \bfseries \entryname &
8416   \bfseries \descriptionname &
8417   \bfseries \symbolname &
8418   \bfseries \pagelistname
8419   \tabularnewline\midrule\endhead
8420   \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.

```
8421 \ifglsgroupskip
8422   \renewcommand*{\glsgroupskip}{}%
8423 \else
8424   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8425 \fi
8426 }
```

long4col-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.

```
8427 \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.

```
8428 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8429 \setglossarystyle{long4col-booktabs}%
```

Change the column specifications:

```
8430 \renewenvironment{theglossary}%
8431   {\begin{longtable}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
8432   {\end{longtable}}%
8433 }
```

Ragged styles.

longragged-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.

```
8434 \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.

```
8435 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8436 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8437 \renewenvironment{theglossary}%  
8438     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}}}%  
8439     {\end{longtable}}%  
8440 }
```

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.

```
8441 \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.

```
8442 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8443 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8444 \renewenvironment{theglossary}%  
8445     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%  
8446       >{\raggedright}p{\glspagelistwidth}}}%  
8447     {\end{longtable}}%  
8448 }
```

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.

```
8449 \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.

```
8450 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8451 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8452 \renewenvironment{theglossary}%  
8453     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8454       >{\raggedright}p{\glspagelistwidth}}}%  
8455     {\end{longtable}}%  
8456 }
```

sLTpenaltycheck

```
8457 \newcommand*{\glslTpenaltycheck}{%  
8458 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8459 }
```



enaltygroupskip

```
8460 \newcommand{\glspenaltygroupskip}{%
8461   \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8462 \let\@gls@org@LT@output\LT@output
8463 \newcommand*{\glsrestoreLToutput}{\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

```
8464 \newcommand*{\glspatchLToutput}{%
8465   \renewcommand*{\LT@output}{%
8466     \ifnum\outputpenalty <-\@Mi
8467       \ifnum\outputpenalty > -\LT@end@pen
8468         \LT@err{floats and marginpars not allowed in a longtable}\@ehc
8469       \else
8470         \setbox\z@\vbox{\unvbox\@cclv}%
8471         \ifdim \ht\LT@lastfoot>\ht\LT@foot
8472           \dimen@\pagegoal
8473           \advance\dimen@-\ht\LT@lastfoot
8474           \ifdim\dimen@<\ht\z@
8475             \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
8476             \@makecol
8477             \@outputpage
8478             \setbox\z@\vbox{\box\LT@head\glsLTpenaltycheck}%
8479           \fi
8480         \fi
8481         \global\@colroom\@colht
8482         \global\vsizel\@colht
8483         {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
8484       \fi
8485     \else
8486       \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
8487       \@makecol
8488       \@outputpage
8489       \global\vsizel\@colroom
8490       \copy\LT@head
8491       \glsLTpenaltycheck
8492       \nobreak
8493     \fi
8494   }%
8495 }
```

### 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.

8496 \ProvidesPackage{glossary-longragged}[2017/06/11 v4.30 (NLCT)]

Requires the package:

8497 \RequirePackage{array}

Requires the package:

8498 \RequirePackage{longtable}

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

8499 \@ifundefined{glsdescwidth}{%

8500 \newlength{glsdescwidth}

8501 \setlength{glsdescwidth}{0.6\hsize}

8502 }{}

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

8503 \@ifundefined{glspagelistwidth}{%

8504 \newlength{glspagelistwidth}

8505 \setlength{glspagelistwidth}{0.1\hsize}

8506 }{}

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

8507 \newglossarystyle{longragged}{%

Use longtable with two columns:

8508 \renewenvironment{theglossary}{%

8509 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%

8510 {\end{longtable}}}%

Do nothing at the start of the environment:

8511 \renewcommand\*{\glossaryheader}{}%

No heading between groups:

8512 \renewcommand\*{\glsgroupheading}[1]{}%

Main (level 0) entries displayed in a row:

8513 \renewcommand{\glossentry}[2]{%

8514 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

8515 \glossentrydesc{##1}\glspostdescription\space ##2%

8516 \tabularnewline

8517 }%

Sub entries displayed on the following row without the name:

```

8518 \renewcommand{\subglossentry}[3]{%
8519     &
8520     \glssubentryitem{##2}%
8521     \glstarget{##2}{\strut}\glossentrydesc{##2}%
8522     \glspostdescription\space ##3%
8523     \tabularnewline
8524 }%
```

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>)

```

8525 \ifglsgroupskip
8526 \renewcommand*{\glsgroupskip}{}%
8527 \else
8528 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8529 \fi
8530 }
```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8531 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8532 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```

8533 \renewenvironment{theglossary}{%
8534 \begin{longtable}{|l|>\raggedright}p{\glsgdescwidth}|}%
8535 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```

8536 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8537 }
```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8538 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8539 \setglossarystyle{longragged}%
```

Set the table's header:

```

8540 \renewcommand*{\glossaryheader}{%
8541 \bfseries \entryname & \bfseries \descriptionname
8542 \tabularnewline\endhead}%
8543 }
```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
8544 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
8545 \setglossarystyle{longraggedborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8546 \renewcommand*{\glossaryheader}{%
8547 \hline\bfseries \entryname & \bfseries \descriptionname
8548 \tabularnewline\hline
8549 \endhead
8550 \hline\endfoot}%
8551 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
8552 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
8553 \renewenvironment{theglossary}%
8554 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8555 >{\raggedright}p{\glspagelistwidth}}}%
8556 {\end{longtable}}%
```

No table header:

```
8557 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8558 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8559 \renewcommand{\glossentry}[2]{%
8560 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8561 \glossentrydesc{##1} & ##2\tabularnewline
8562 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8563 \renewcommand{\subglossentry}[3]{%
8564 &
8565 \glssubentryitem{##2}%
8566 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8567 ##3\tabularnewline
8568 }%
```

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>)

```
8569 \ifglsnogroupskip
8570 \renewcommand*{\glsgroupskip}{}%
8571 \else
8572 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8573 \fi
8574 }
```

`ragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
8575 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
8576 \setglossarystyle{longragged3col}%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
8577 \renewenvironment{theglossary}%  
8578 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%  
8579 >{\raggedright}p{\glspagelistwidth}|}%  
8580 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8581 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
8582 }
```

`ragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8583 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8584 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
8585 \renewcommand*{\glossaryheader}{%  
8586 \bfseries\entryname&\bfseries\descriptionname&  
8587 \bfseries\pagelistname\tabularnewline\endhead}%  
8588 }
```

`colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8589 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8590 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8591 \renewcommand*{\glossaryheader}{%  
8592 \hline  
8593 \bfseries\entryname&\bfseries\descriptionname&  
8594 \bfseries\pagelistname\tabularnewline\hline\endhead  
8595 \hline\endfoot}%  
8596 }
```

`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.

```
8597 \newglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8598 \renewenvironment{theglossary}%  
8599 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8600 >{\raggedright}p{\glspagelistwidth}}}%  
8601 {\end{longtable}}%
```

No table header:

```
8602 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8603 \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):

```
8604 \renewcommand{\glossentry}[2]{%
8605   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8606   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8607   ##2\tabularnewline
8608 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8609 \renewcommand{\subglossentry}[3]{%
8610   &
8611   \glssubentryitem{##2}%
8612   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8613   \glossentrysymbol{##2} & ##3\tabularnewline
8614 }%
```

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>)

```
8615 \ifglsnogroupskip
8616 \renewcommand*\glsgroupskip{}{%
8617 \else
8618 \renewcommand*\glsgroupskip{ & & & \tabularnewline}%
8619 \fi
8620 }
```

agged4colheader The altlongragged4colheader style is like altlongragged4col but with a header row.

```
8621 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the glostylealtlongragged4col style:

```
8622 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8623 \renewenvironment{theglossary}%
8624   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8625     >{\raggedright}p{\glspagelistwidth}}}%
8626   {\end{longtable}}%
```

Table has a header:

```
8627 \renewcommand*\glossaryheader{%
8628   \bfseries\entryname&\bfseries\descriptionname&
8629   \bfseries \symbolname&
8630   \bfseries\pagelistname\tabularnewline\endhead}%
8631 }
```

agged4colborder The altlongragged4colborder style is like altlongragged4col but with a border.

```
8632 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8633 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8634 \renewenvironment{theglossary}%  
8635 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%  
8636 >{\raggedright}p{\glspagelistwidth}|}}%  
8637 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8638 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
8639 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8640 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8641 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8642 \renewenvironment{theglossary}%  
8643 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%  
8644 >{\raggedright}p{\glspagelistwidth}|}}%  
8645 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8646 \renewcommand*{\glossaryheader}{%  
8647 \hline\bfseries\entryname&\bfseries\descriptionname&  
8648 \bfseries \symbolname&  
8649 \bfseries\pagelistname\tabularnewline\hline\endhead  
8650 \hline\endfoot}%  
8651 }
```

### 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.

```
8652 \ProvidesPackage{glossary-mcols}[2017/06/11 v4.30 (NLCT)]
```

Required packages:

```
8653 \RequirePackage{multicol}  
8654 \RequirePackage{glossary-tree}
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8655 \providecommand{\indexspace}{%  
8656 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax  
8657 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8658 \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.)

```
8659 \newglossarystyle{mcolindex}{%
8660   \setglossarystyle{index}%
8661   \renewenvironment{theglossary}%
8662     {%
8663       \begin{multicols}{\glsmcols}
8664       \setlength{\parindent}{0pt}%
8665       \setlength{\parskip}{0pt plus 0.3pt}%
8666       \let\item\glstreeitem
8667       \let\subitem\glstreesubitem
8668       \let\subsubitem\glstreesubsubitem
8669     }%
8670     {\end{multicols}}%
8671 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8672 \newglossarystyle{mcolindexgroup}{%
8673   \setglossarystyle{mcolindex}%
8674   \renewcommand*{\glsgroupheading}[1]{%
8675     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8676 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8677 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8678   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8679   \renewcommand*{\glossaryheader}{%
8680     \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.

```
8681   \renewcommand*{\glsgroupheading}[1]{%
8682     \item\glstreegroupheaderfmt
8683       {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
8684     \indexspace}%
8685 }
```

`colindexspannav` Similar to `mcolindexhypergroup`, but puts the navigation line in the optional argument of multicols.



```

8686 \newglossarystyle{mcolindexspannav}{%
8687   \setglossarystyle{index}%
8688   \renewenvironment{theglossary}%
8689     {%

8690     \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8691     \setlength{\parindent}{0pt}%
8692     \setlength{\parskip}{0pt plus 0.3pt}%

8693     \let\item\glstreeitem}%
8694   {\end{multicols}}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

8695 \renewcommand*{\glsgroupheading}[1]{%
8696   \item\glstreegroupheaderfmt
8697     {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
8698   \indexspace}%
8699 }

```

**mcoltree** Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

8700 \newglossarystyle{mcoltree}{%
8701   \setglossarystyle{tree}%
8702   \renewenvironment{theglossary}%
8703     {%

8704     \begin{multicols}{\glsmcols}
8705     \setlength{\parindent}{0pt}%
8706     \setlength{\parskip}{0pt plus 0.3pt}%
8707   }%
8708   {\end{multicols}}%
8709 }

```

**mcoltreegroup** Like the mcoltree style but the glossary groups have headings.

```

8710 \newglossarystyle{mcoltreegroup}{%
  Base it on the glostylemcoltree style:
8711   \setglossarystyle{mcoltree}%
  Each group has a heading (in bold) followed by a vertical gap):
8712   \renewcommand{\glsgroupheading}[1]{\par
8713     \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8714 }

```

**1treehypergroup** The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8715 \newglossarystyle{mcoltreehypergroup}{%
  Base it on the glostylemcoltree style:
8716   \setglossarystyle{mcoltree}%

```

Put navigation links to the groups at the start of the theglossary environment:

```
8717 \renewcommand*{\glossaryheader}{%
8718 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8719 \renewcommand*{\glsgroupheading}[1]{%
8720 \par\noindent
8721 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8722 \indexspace}%
8723 }
```

**mcoltreespannav** Similar to the mcoltreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```
8724 \newglossarystyle{mcoltreespannav}{%
8725 \setglossarystyle{tree}%
8726 \renewenvironment{theglossary}%
8727 {%
8728 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8729 \setlength{\parindent}{0pt}%
8730 \setlength{\parskip}{0pt plus 0.3pt}%
8731 }%
8732 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8733 \renewcommand*{\glsgroupheading}[1]{%
8734 \par\noindent
8735 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8736 \indexspace}%
8737 }
```

**mcoltreenoname** Multi-column index style. Same as the treenoname, but puts the glossary in multiple columns.

```
8738 \newglossarystyle{mcoltreenoname}{%
8739 \setglossarystyle{treenoname}%
8740 \renewenvironment{theglossary}%
8741 {%
8742 \begin{multicols}{\glsmcols}
8743 \setlength{\parindent}{0pt}%
8744 \setlength{\parskip}{0pt plus 0.3pt}%
8745 }%
8746 {\end{multicols}}}%
8747 }
```

**treenonamegroup** Like the mcoltreenoname style but the glossary groups have headings.

```
8748 \newglossarystyle{mcoltreenonamegroup}{%
```

Base it on the glostylemcoltreenoname style:

```
8749 \setglossarystyle{mcoltreenoname}%
```

Give each group a heading:

```
8750 \renewcommand{\glsgroupheading}[1]{\par
8751 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8752 }
```

`onamehypergroup` The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8753 \newglossarystyle{mcoltreenonamehypergroup}{%
```

Base it on the `glostylemcoltreenoname` style:

```
8754 \setglossarystyle{mcoltreenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8755 \renewcommand*{\glossaryheader}{%
8756 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8757 \renewcommand*{\glsgroupheading}[1]{%
8758 \par\noindent
8759 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8760 \indexspace}%
8761 }
```

`eenonamespannav` Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8762 \newglossarystyle{mcoltreenonamespannav}{%
8763 \setglossarystyle{treenoname}%
8764 \renewenvironment{theglossary}%
8765 {%
8766 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8767 \setlength{\parindent}{0pt}%
8768 \setlength{\parskip}{0pt plus 0.3pt}%
8769 }%
8770 {\end{multicols}}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8771 \renewcommand*{\glsgroupheading}[1]{%
8772 \par\noindent
8773 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8774 \indexspace}%
8775 }
```

`mcolalttree` Multi-column index style. Same as the `alttree`, but puts the glossary in multiple columns.

```
8776 \newglossarystyle{mcolalttree}{%
8777 \setglossarystyle{alttree}%
8778 \renewenvironment{theglossary}%
8779 {%
8780 \begin{multicols}{\glsmcols}
8781 \def\@gls@prevlevel{-1}%
```

```

8782     \mbox{}\par
8783   }%
8784   {\par\end{multicols}}}%
8785 }

```

**colalmtreegroup** Like the mcolalmtree style but the glossary groups have headings.

```

8786 \newglossarystyle{mcolalmtreegroup}{%
      Base it on the glostylemcolalmtree style:
8787   \setglossarystyle{mcolalmtree}%
      Give each group a heading.
8788   \renewcommand{\glsgroupheading}[1]{\par
8789     \def\@gls@prevlevel{-1}%
8790     \hangindent0pt\relax
8791     \parindent0pt\relax
8792     \glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8793 }

```

**treehypergroup** The mcolalmtreehypergroup style is like the mcolalmtreegroup style, but has a set of links to the groups at the start of the glossary.

```

8794 \newglossarystyle{mcolalmtreehypergroup}{%
      Base it on the glostylemcolalmtree style:
8795   \setglossarystyle{mcolalmtree}%
      Put the navigation links in the header
8796   \renewcommand*{\glossaryheader}{%
8797     \par
8798     \def\@gls@prevlevel{-1}%
8799     \hangindent0pt\relax
8800     \parindent0pt\relax
8801     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8802   \renewcommand*{\glsgroupheading}[1]{%
8803     \par
8804     \def\@gls@prevlevel{-1}%
8805     \hangindent0pt\relax
8806     \parindent0pt\relax
8807     \glstreegroupheaderfmt{\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8808     \indexspace}%
8809 }

```

**almtreepannav** Similar to the mcolalmtreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```

8810 \newglossarystyle{mcolalmtreepannav}{%
8811   \setglossarystyle{almtree}%
8812   \renewenvironment{theglossary}%
8813   {%
8814     \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]

```

```

8815     \def\@gls@prevlevel{-1}%
8816     \mbox{}\par
8817 }%
8818 {\par\end{multicols}}}%
    Put a hypertarget at the start of each group
8819 \renewcommand*\@gls@groupheading[1]{%
8820     \par
8821     \def\@gls@prevlevel{-1}%
8822     \hangindent0pt\relax
8823     \parindent0pt\relax
8824     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8825     \indexspace}
8826 }

```

### 3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```
8827 \ProvidesPackage{glossary-super}[2017/06/11 v4.30 (NLCT)]
```

Requires the package:

```
8828 \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.

```

8829 \@ifundefined{glsdescwidth}{%
8830     \newlength\glsdescwidth
8831     \setlength{\glsdescwidth}{0.6\hsize}
8832 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if `has` has been loaded.

```

8833 \@ifundefined{glspagelistwidth}{%
8834     \newlength\glspagelistwidth
8835     \setlength{\glspagelistwidth}{0.1\hsize}
8836 }{}

```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```
8837 \newglossarystyle{super}{%
```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8838     \renewenvironment{theglossary}%
8839     {\tablehead{}\tabletail{}}%
8840     \begin{supertabular}{lp{\glsdescwidth}}}%
8841     {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
8842 \renewcommand*\glossaryheader{}\%
```

No group headings:

```
8843 \renewcommand*\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8844 \renewcommand\glossentry}[2]{%
8845   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8846   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8847 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8848 \renewcommand\subglossentry}[3]{%
8849   &
8850   \glssubentryitem{##2}%
8851   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8852   ##3\tabularnewline
8853 }%
```

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>)

```
8854 \ifglsgroupskip
8855   \renewcommand*\glsgroupskip{}\%
8856 \else
8857   \renewcommand*\glsgroupskip}{& \tabularnewline}%
8858 \fi
8859 }
```

**superborder** The superborder style is like the above, but with horizontal and vertical lines:

```
8860 \newglossarystyle{superborder}{%
```

Base it on the `glostylesuper` style:

```
8861 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8862 \renewenvironment{theglossary}%
8863   {\tablehead{\hline}\tabletail{\hline}%
8864   \begin{supertabular}{|l|p{\glsglwidth}|}%
8865   {\end{supertabular}}%
8866 }
```

**superheader** The superheader style is like the super style, but with a header:

```
8867 \newglossarystyle{superheader}{%
```

Base it on the `glostylesuper` style:

```
8868 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```
8869 \renewenvironment{theglossary}%
8870   {\tablehead{\bfseries \entryname &
8871     \bfseries\descriptionname\tabularnewline}%
8872   \tabletail{}}%
8873   \begin{supertabular}{lp{\glsgdescwidth}}%
8874   {\end{supertabular}}%
8875 }
```

**superheaderborder** The superheaderborder style is like the super style but with a header and border:

```
8876 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylessuper style:

```
8877 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```
8878 \renewenvironment{theglossary}%
8879   {\tablehead{\hline\bfseries \entryname &
8880     \bfseries \descriptionname\tabularnewline\hline}%
8881   \tabletail{\hline}
8882   \begin{supertabular}{|lp{\glsgdescwidth}|}%
8883   {\end{supertabular}}%
8884 }
```

**super3col** The super3col style is like the super style, but with 3 columns:

```
8885 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```
8886 \renewenvironment{theglossary}%
8887   {\tablehead{}\tabletail{}}%
8888   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}%
8889   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8890 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8891 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8892 \renewcommand{\glossentry}[2]{%
8893   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8894   \glossentrydesc{##1} & ##2\tabularnewline
8895   }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
8896 \renewcommand{\subglossentry}[3]{%
8897   &
8898   \glssubentryitem{##2}%
```

```

8899     \glstarget{##2}{\strut}\glossentrydesc{##2} &
8900     ##3\tabularnewline
8901 }%

```

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>)

```

8902 \ifglsnogroupskip
8903 \renewcommand*{\glsgroupskip}{}%
8904 \else
8905 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8906 \fi
8907 }

```

**super3colborder** The super3colborder style is like the super3col style, but with a border:

```
8908 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
8909 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

8910 \renewenvironment{theglossary}%
8911 {\tablehead{\hline}\tabletail{\hline}%
8912 \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth|}}%
8913 {\end{supertabular}}%
8914 }

```

**super3colheader** The super3colheader style is like the super3col style but with a header row:

```
8915 \newglossarystyle{super3colheader}{%
```

Base it on the glostylesuper3col style:

```
8916 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

8917 \renewenvironment{theglossary}%
8918 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8919 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8920 \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}%
8921 {\end{supertabular}}%
8922 }

```

**colheaderborder** The super3colheaderborder style is like the super3col style but with a header and border:

```
8923 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
8924 \setglossarystyle{super3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```

8925 \renewenvironment{theglossary}%
8926 {\tablehead{\hline

```



```

8927      \bfseries\entryname&\bfseries\descriptionname&
8928      \bfseries\pagelistname\tabularnewline\hline}%
8929      \tabletail{\hline}%
8930      \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8931      {\end{supertabular}}}%
8932 }

```

**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.

```

8933 \newglossarystyle{super4col}{%

```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

8934 \renewenvironment{theglossary}%
8935 {\tablehead{}\tabletail}%
8936 \begin{supertabular}{|l|l|l|l|}%
8937 \end{supertabular}}%

```

Do nothing at the start of the table:

```

8938 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8939 \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:

```

8940 \renewcommand{\glossentry}[2]{%
8941 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8942 \glossentrydesc{##1} &
8943 \glossentrysymbol{##1} & ##2\tabularnewline
8944 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8945 \renewcommand{\subglossentry}[3]{%
8946 &
8947 \glssubentryitem{##2}%
8948 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8949 \glossentrysymbol{##2} & ##3\tabularnewline
8950 }%

```

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>)

```

8951 \ifglsgroupskip
8952 \renewcommand*{\glsgroupskip}{}%
8953 \else
8954 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8955 \fi
8956 }

```

**super4colheader** The super4colheader style is like the super4col but with a header row.

```

8957 \newglossarystyle{super4colheader}{%

```

Base it on the `glostylesuper4col` style:

```
8958 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
8959 \renewenvironment{theglossary}%
8960 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8961 \bfseries\symbolname &
8962 \bfseries\pagelistname\tabularnewline}%
8963 \tabletail{}}%
8964 \begin{supertabular}{|l|l|l|l|}%
8965 {\end{supertabular}}%
8966 }
```

`super4colborder` The `super4colborder` style is like the `super4col` but with a border.

```
8967 \newglossarystyle{super4colborder}{%
```

Base it on the `glostylesuper4col` style:

```
8968 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8969 \renewenvironment{theglossary}%
8970 {\tablehead{\hline}\tabletail{\hline}%
8971 \begin{supertabular}{|l|l|l|l|}%
8972 {\end{supertabular}}%
8973 }
```

`colheaderborder` The `super4colheaderborder` style is like the `super4col` but with a header and border.

```
8974 \newglossarystyle{super4colheaderborder}{%
```

Base it on the `glostylesuper4col` style:

```
8975 \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:

```
8976 \renewenvironment{theglossary}%
8977 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8978 \bfseries\symbolname &
8979 \bfseries\pagelistname\tabularnewline\hline}%
8980 \tabletail{\hline}%
8981 \begin{supertabular}{|l|l|l|l|}%
8982 {\end{supertabular}}%
8983 }
```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```
8984 \newglossarystyle{altsuper4col}{%
```

Base it on the `glostylesuper4col` style:

```
8985 \setglossarystyle{super4col}%
```

Put the glossary in a supertabular environment with four columns and no head or tail:

```
8986 \renewenvironment{theglossary}%
8987 {\tablehead{}\tabletail{}}%
8988 \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
8989 {\end{supertabular}}%
8990 }
```

**super4colheader** The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```
8991 \newglossarystyle{altsuper4colheader}{%
```

Base it on the `glostylesuper4colheader` style:

```
8992 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8993 \renewenvironment{theglossary}%
8994 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8995 \bfseries\symbolname &
8996 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8997 \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
8998 {\end{supertabular}}%
8999 }
```

**super4colborder** The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```
9000 \newglossarystyle{altsuper4colborder}{%
```

Base it on the `glostylesuper4colborder` style:

```
9001 \setglossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9002 \renewenvironment{theglossary}%
9003 {\tablehead{\hline}\tabletail{\hline}%
9004 \begin{supertabular}%
9005 {lllp{\glsdescwidth}lllp{\glspagelistwidth}}}%
9006 {\end{supertabular}}%
9007 }
```

**colheaderborder** The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```
9008 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the `glostylesuper4colheaderborder` style:

```
9009 \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:

```
9010 \renewenvironment{theglossary}%
9011 {\tablehead{\hline
9012 \bfseries\entryname &
9013 \bfseries\descriptionname &
9014 \bfseries\symbolname &
9015 \bfseries\pagelistname\tabularnewline\hline}%
```

```

9016      \tabletail{\hline}%
9017      \begin{supertabular}%
9018          {ll|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%
9019      {\end{supertabular}}%
9020 }

```

### 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.

```

9021 \ProvidesPackage{glossary-superragged}[2017/06/11 v4.30 (NLCT)]

```

Requires the package:

```

9022 \RequirePackage{array}

```

Requires the package:

```

9023 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

9024 \@ifundefined{glsdescwidth}{%
9025     \newlength{glsdescwidth
9026     \setlength{glsdescwidth}{0.6\hsize}
9027 }{}

```

`glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

9028 \@ifundefined{glspagelistwidth}{%
9029     \newlength{glspagelistwidth
9030     \setlength{glspagelistwidth}{0.1\hsize}
9031 }{}

```

`superragged` The superragged glossary style uses the supertabular environment.

```

9032 \newglossarystyle{superragged}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

9033     \renewenvironment{theglossary}%
9034     {\tablehead{}\tabletail{}}%
9035     \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}%
9036     {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9037     \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9038     \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

9039 \renewcommand{\glossentry}[2]{%
9040   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9041   \glossentrydesc{##1}\glspostdescription\space ##2%
9042   \tabularnewline
9043 }%

```

Sub entries put in a row (no name, description and page list in second column):

```

9044 \renewcommand{\subglossentry}[3]{%
9045   &
9046   \glssubentryitem{##2}%
9047   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9048   ##3%
9049   \tabularnewline
9050 }%

```

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>)

```

9051 \ifglsgroupskip
9052   \renewcommand*{\glsgroupskip}{}%
9053 \else
9054   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9055 \fi
9056 }

```

**superraggedborder** The superraggedborder style is like the above, but with horizontal and vertical lines:

```

9057 \newglossarystyle{superraggedborder}{%

```

Base it on the glostylessuperragged style:

```

9058 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```

9059 \renewenvironment{theglossary}%
9060   {\tablehead{\hline}\tabletail{\hline}%
9061   \begin{supertabular}{|l|>{\raggedright}p{\glsglwidth}}}%
9062   {\end{supertabular}}%
9063 }

```

**superraggedheader** The superraggedheader style is like the super style, but with a header:

```

9064 \newglossarystyle{superraggedheader}{%

```

Base it on the glostylessuperragged style:

```

9065 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

9066 \renewenvironment{theglossary}%
9067   {\tablehead{\bfseries \entryname & \bfseries \descriptionname
9068   \tabularnewline}%
9069   \tabletail{}}%

```

```

9070 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9071 {\end{supertabular}}}%
9072 }

```

gedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```

9073 \newglossarystyle{superraggedheaderborder}{%

```

Base it on the glostylesuper style:

```

9074 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

9075 \renewenvironment{theglossary}%
9076 {\tablehead{\hline\bfseries \entryname &
9077 \bfseries \descriptionname\tabularnewline\hline}%
9078 \tabletail{\hline}
9079 \begin{supertabular}{l|>{\raggedright}p{\glsgdescwidth}|}%
9080 {\end{supertabular}}}%
9081 }

```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```

9082 \newglossarystyle{superragged3col}{%

```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

9083 \renewenvironment{theglossary}%
9084 {\tablehead{}\tabletail{}}%
9085 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
9086 >{\raggedright}p{\glspagelistwidth}}}%
9087 {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9088 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9089 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

9090 \renewcommand{\glossentry}[2]{%
9091 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9092 \glossentrydesc{##1} &
9093 ##2\tabularnewline
9094 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9095 \renewcommand{\subglossentry}[3]{%
9096 &
9097 \glssubentryitem{##2}%
9098 \glstarget{##2}{\strut}\glossentrydesc{##2} &
9099 ##3\tabularnewline
9100 }%

```

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>)

```

9101 \ifglsgroupskip
9102 \renewcommand*{\glsgroupskip}{}%
9103 \else
9104 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9105 \fi
9106 }

```

**superragged3colborder** The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```

9107 \newglossarystyle{superragged3colborder}{%

```

Base it on the `glostypesuperragged3col` style:

```

9108 \setglossarystyle{superragged3col}%

```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```

9109 \renewenvironment{theglossary}%
9110 {\tablehead{\hline}\tabletail{\hline}%
9111 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}%
9112 >{\raggedright}p{\glspagelistwidth}|}%
9113 {\end{supertabular}}%
9114 }

```

**superragged3colheader** The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```

9115 \newglossarystyle{superragged3colheader}{%

```

Base it on the `glostypesuperragged3col` style:

```

9116 \setglossarystyle{superragged3col}%

```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```

9117 \renewenvironment{theglossary}%
9118 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9119 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9120 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9121 >{\raggedright}p{\glspagelistwidth}}}%
9122 {\end{supertabular}}%
9123 }

```

**superragged3colheaderborder** The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```

9124 \newglossarystyle{superragged3colheaderborder}{%

```

Base it on the `glostypesuperragged3colborder` style:

```

9125 \setglossarystyle{superragged3colborder}%

```

Put the glossary in a `supertabular` environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```

9126 \renewenvironment{theglossary}%
9127 {\tablehead{\hline

```

```

9128      \bfseries\entryname&\bfseries\descriptionname&
9129      \bfseries\pagelistname\tabularnewline\hline}%
9130      \tabletail{\hline}%
9131      \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
9132      >{\raggedright}p{\glspagelistwidth}|}%
9133      {\end{supertabular}}%
9134 }

```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```

9135 \newglossarystyle{altsuperragged4col}{%

```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

9136 \renewenvironment{theglossary}%
9137 {\tablehead{\tabletail}{}%
9138 \begin{supertabular}{|l>{\raggedright}p{\glsdescwidth}|%
9139 >{\raggedright}p{\glspagelistwidth}|}%
9140 {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9141 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9142 \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:

```

9143 \renewcommand{\glossentry}[2]{%
9144 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9145 \glossentrydesc{##1} &
9146 \glossentrysymbol{##1} & ##2\tabularnewline
9147 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

9148 \renewcommand{\subglossentry}[3]{%
9149 &
9150 \glssubentryitem{##2}%
9151 \glstarget{##2}{\strut}\glossentrydesc{##2} &
9152 \glossentrysymbol{##2} & ##3\tabularnewline
9153 }%

```

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>)

```

9154 \ifglsgroupskip
9155 \renewcommand*{\glsgroupskip}{}%
9156 \else
9157 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9158 \fi
9159 }

```



agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
9160 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
9161 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9162 \renewenvironment{theglossary}%  
9163 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&  
9164 \bfseries\symbolname &  
9165 \bfseries\pagelistname\tabularnewline}\tabletail{}}%  
9166 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%  
9167 >{\raggedright}p{\glspagelistwidth}}}%  
9168 {\end{supertabular}}}%  
9169 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
9170 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9171 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9172 \renewenvironment{theglossary}%  
9173 {\tablehead{\hline}\tabletail{\hline}%  
9174 \begin{supertabular}%  
9175 {ll>{\raggedright}p{\glsgdescwidth}ll}%  
9176 >{\raggedright}p{\glspagelistwidth}l}}}%  
9177 {\end{supertabular}}}%  
9178 }
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
9179 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9180 \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:

```
9181 \renewenvironment{theglossary}%  
9182 {\tablehead{\hline  
9183 \bfseries\entryname &  
9184 \bfseries\descriptionname &  
9185 \bfseries\symbolname &  
9186 \bfseries\pagelistname\tabularnewline\hline}%  
9187 \tabletail{\hline}%  
9188 \begin{supertabular}%  
9189 {ll>{\raggedright}p{\glsgdescwidth}ll}%  
9190 >{\raggedright}p{\glspagelistwidth}l}}}%  
9191 }
```

```

9191     {\end{supertabular}}}%
9192 }

```

### 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.

```

9193 \ProvidesPackage{glossary-tree}[2017/06/11 v4.30 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9194 \providecommand{\indexspace}{%
9195   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9196 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glslnamefont`.) This command was previously also used to format the group headings.

```

9197 \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.

```

9198 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9199 \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.

```

9200 \ifdef\@idxitem
9201 {\newcommand{\glstreeitem}{\@idxitem}}
9202 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```

9203 \ifdef\subitem
9204 {\let\glstreesubitem\subitem}
9205 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}

```

`\glstreesubsubitem` Level 1 item used in index style.

```

9206 \ifdef\subsubitem
9207 {\let\glstreesubsubitem\subsubitem}
9208 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}

```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `almtree` style).

```

9209 \newcommand{\glstreepredesc}{\space}

```

treechildpredesc Allow the user to adjust the space before the description for sub-entries (except for the treename and alttree style).

```
9210 \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.

```
9211 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by theindex:

```
9212 \renewenvironment{theglossary}%
9213 {\setlength{\parindent}{0pt}%
9214 \setlength{\parskip}{0pt plus 0.3pt}%
9215 \let\item\glstreeitem
9216 \let\subitem\glstreesubitem
9217 \let\subsubitem\glstreesubsubitem
9218 }%
```

```
9219 {\par}%
```

Do nothing at the start of the environment:

```
9220 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9221 \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.

```
9222 \renewcommand*{\glossentry}[2]{%
9223 \item\glssentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
9224 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9225 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9226 }%
```

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.

```
9227 \renewcommand{\subglossentry}[3]{%
9228 \ifcase##1\relax
9229 % level 0
9230 \item
9231 \or
9232 % level 1
9233 \subitem
9234 \glssubentryitem{##2}%
9235 \else
9236 % all other levels
```

```

9237     \subsubitem
9238     \fi
9239     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9240     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9241     \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9242 }%

```

Vertical gap between groups is the same as that used by indices:

```

9243 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}

```

**indexgroup** The indexgroup style is like the index style but has headings.

```

9244 \newglossarystyle{indexgroup}{%

```

Base it on the glostyleindex style:

```

9245 \setglossarystyle{index}%

```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9246 \renewcommand*{\glsgroupheading}[1]{%
9247 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9248 \indexspace
9249 }%
9250 }

```

**indexhypergroup** The indexhypergroup style is like the indexgroup style but has hyper navigation.

```

9251 \newglossarystyle{indexhypergroup}{%

```

Base it on the glostyleindex style:

```

9252 \setglossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

9253 \renewcommand*{\glossaryheader}{%
9254 \item\glstreenavigationfmt{\glsgroupnavigation}\indexspace}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

9255 \renewcommand*{\glsgroupheading}[1]{%
9256 \item\glstreegroupheaderfmt
9257 {\glsgroupnavigationtarget{##1}{\glsgrouptitle{##1}}}%
9258 \indexspace}%
9259 }

```

**tree** The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9260 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9261 \renewenvironment{theglossary}%
9262 {\setlength{\parindent}{0pt}%
9263 \setlength{\parskip}{0pt plus 0.3pt}}%
9264 {}%

```

Do nothing at the start of the theglossary environment:

```

9265 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
9266 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
9267 \renewcommand{\glossentry}[2]{%
9268   \hangindent0pt\relax
9269   \parindent0pt\relax
9270   \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9271   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9272   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9273 }%
```

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.

```
9274 \renewcommand{\subglossentry}[3]{%
9275   \hangindent##1\glstreeindent\relax
9276   \parindent##1\glstreeindent\relax
9277   \ifnum##1=1\relax
9278     \glssubentryitem{##2}%
9279   \fi
9280   \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9281   \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9282   \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9283 }%
```

Vertical gap between groups is the same as that used by indices:

```
9284 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

**treegroup** Like the tree style but the glossary groups have headings.

```
9285 \newglossarystyle{treegroup}{%
```

Base it on the `glostyletree` style:

```
9286 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9287 \renewcommand{\glsgroupheading}[1]{\par
9288   \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
9289   \indexspace}%
9290 }
```

**treehypergroup** The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
9291 \newglossarystyle{treehypergroup}{%
```

Base it on the `glostyletree` style:

```
9292 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9293 \renewcommand*{\glossaryheader}{%
9294   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9295 \renewcommand*{\glsgroupheading}[1]{%
9296   \par\noindent
9297   \glstreegroupheaderfmt
9298   {\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9299   \indexspace}%
9300 }
```

`\glstreeindent` Length governing left indent for each level of the tree style.

```

9301 \newlength\glstreeindent
9302 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
9303 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```

9304 \renewenvironment{theglossary}%
9305   {\setlength{\parindent}{0pt}%
9306   \setlength{\parskip}{0pt plus 0.3pt}}%
9307   {}%
```

No header:

```
9308 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9309 \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.

```

9310 \renewcommand{\glossentry}[2]{%
9311   \hangindent0pt\relax
9312   \parindent0pt\relax
9313   \glssentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9314   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9315   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9316   }%
```

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.

```

9317 \renewcommand{\subglossentry}[3]{%
9318   \hangindent##1\glstreeindent\relax
9319   \parindent##1\glstreeindent\relax
9320   \ifnum##1=1\relax
9321     \glssubentryitem{##2}%
9322     \fi
9323     \glstarget{##2}{\strut}%
9324     \glossentrydesc{##2}\glspostdescription\space##3\par
9325   }%
```

Vertical gap between groups is the same as that used by indices:

```
9326 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9327 }
```

**treenonamegroup** Like the `treenoname` style but the glossary groups have headings.

```
9328 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
9329 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
9330 \renewcommand{\glsgroupheading}[1]{\par
9331 \noindent\glstreegroupheaderfmt
9332 {\glsgetgrouptitle{##1}}\par\indexspace}%
9333 }
```

**onamehypergroup** The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
9334 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostyletreenoname` style:

```
9335 \setglossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9336 \renewcommand*{\glossaryheader}{%
9337 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9338 \renewcommand*{\glsgroupheading}[1]{%
9339 \par\noindent
9340 \glstreegroupheaderfmt
9341 {\glsnavhypertarget{##1}}{\glsgetgrouptitle{##1}}\par
9342 \indexspace}%
9343 }
```

**esttoplevelname** Find the widest name over all parentless entries in the given glossary or glossaries.

```
9344 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9345 \dimen@=0pt\relax
9346 \gls@tmplen=0pt\relax
9347 \forallglossaries[#1]{\@gls@type}%
9348 {%
9349 \forallgsentries[\@gls@type]{\@glo@label}%
9350 {%
9351 \ifglshasparent{\@glo@label}%
9352 }%
9353 {%
9354 \settowidth{\dimen@}%
9355 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
9356 \ifdim\dimen@>\gls@tmplen
9357 \gls@tmplen=\dimen@
```

```

9358         \letcs{\@glswidestname}{glo@glsetoklabel{\@glo@label}@name}%
9359         \fi
9360     }%
9361 }%
9362 }%
9363 }

```

`\glsetwidest` `\glsetwidest[<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.

```

9364 \newcommand*{\glsetwidest}[2][0]{%
9365     \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9366         #2}%
9367 }

```

`\@glswidestname` Initialise `\@glswidestname`.

```

9368 \newcommand*{\@glswidestname}{}

```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9369 \newcommand*{\glstreenamebox}[2]{%
9370     \makebox[#1][l]{#2}%
9371 }

```

**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 `\glsetwidest`.

```

9372 \newglossarystyle{alttree}{%

```

Redefine theglossary environment.

```

9373     \renewenvironment{theglossary}%
9374     {\def\@gls@prevlevel{-1}%
9375     \mbox{}\par}%
9376     {\par}%

```

Set the header and group headers to nothing.

```

9377     \renewcommand*{\glossaryheader}{}%
9378     \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```

9379     \renewcommand{\glossentry}[2]{%
9380         \ifnum\@gls@prevlevel=0\relax
9381         \else

```

Find out how big the indentation should be by measuring the widest entry.

```

9382         \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9383         \fi

```

Set the hangindent and paragraph indent.

```

9384         \hangindent\glstreeindent
9385         \parindent\glstreeindent

```

Put the name to the left of the paragraph block.

```

9386         \makebox[0pt][r]{\glstreenamebox{\glstreeindent}{%
9387             \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%

```



If the symbol is missing, ignore it, otherwise put it in brackets.

```
9388 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9389 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9390 \def\@gls@prevlevel{0}%
9391 }%
```

Redefine the way sub-entries are displayed.

```
9392 \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.

```
9393 \ifnum##1=1\relax
9394 \glssubentryitem{##2}%
9395 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9396 \ifnum\@gls@prevlevel=##1\relax
9397 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9398 \@ifundefined{@glswidestname\romannumeral##1}{%
9399 \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%
9400 \settowidth{\gls@tmplen}{\glstreenamefmt{%
9401 \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
9402 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9403 \setlength\glstreeindent\gls@tmplen
9404 \addtolength\glstreeindent\parindent
9405 \parindent\glstreeindent
9406 \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`.

```
9407 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
9408 \settowidth{\glstreeindent}{\glstreenamefmt{%
9409 \@glswidestname\space}}}%
9410 \settowidth{\glstreeindent}{\glstreenamefmt{%
9411 \csname @glswidestname\romannumeral\@gls@prevlevel
9412 \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9413 \addtolength\parindent{-\glstreeindent}%
```

```

9414         \setlength\glstreeindent\parindent
9415     \fi
9416 \fi

    Set the hanging indentation.
9417     \hangindent\glstreeindent

    Put the name to the left of the paragraph block
9418     \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
9419         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%

    If the symbol is missing, ignore it, otherwise put it in brackets.
9420     \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%

    Do the description followed by the description terminator and location list.
9421     \glossentrydesc{##2}\glspostdescription\space ##3\par

    Set the previous level macro to the current level.
9422     \def\@gls@prevlevel{##1}%
9423 }%

    Vertical gap between groups is the same as that used by indices:
9424 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9425 }

```

**alttreegroup** Like the **alttree** style but the glossary groups have headings.

```

9426 \newglossarystyle{alttreegroup}{%
    Base it on the glostylealttree style:
9427     \setglossarystyle{alttree}%

    Give each group a heading.
9428     \renewcommand{\glsgroupheading}[1]{\par
9429         \def\@gls@prevlevel{-1}%
9430         \hangindent0pt\relax
9431         \parindent0pt\relax
9432         \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
9433         \par\indexspace}%
9434 }

```

**alttreehypergroup** The **alttreehypergroup** style is like the **alttreegroup** style, but has a set of links to the groups at the start of the glossary.

```

9435 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9436     \setglossarystyle{alttree}%

    Put the navigation links in the header
9437     \renewcommand*{\glossaryheader}{%
9438         \par
9439         \def\@gls@prevlevel{-1}%
9440         \hangindent0pt\relax
9441         \parindent0pt\relax
9442         \glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```

Put a hypertarget at the start of each group

```
9443 \renewcommand*{\glsgroupheading}[1]{%
9444   \par
9445   \def\@gls@prevlevel{-1}%
9446   \hangindent0pt\relax
9447   \parindent0pt\relax
9448   \glstreegroupheaderfmt
9449   {\\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9450   \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.

```
9451 \NeedsTeXFormat{LaTeX2e}
9452 \ProvidesPackage{glossaries-compatible-207}[2017/06/11 v4.30 (NLCT)]
```

**AddXdyAttribute** Adds an attribute in old format.

```
9453 \ifglsxindy
9454   \renewcommand*\GlsAddXdyAttribute[1]{%
9455     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9456     \expandafter\toks@\expandafter{\@xdylocref}%
9457     \edef\@xdylocref{\the\toks@ ^^J}%
9458     (markup-locref
9459     :open \string"\string~n\string\setentrycounter
9460       {\noexpand\glscounter}%
9461       \expandafter\string\csname#1\endcsname
9462       \expandafter\@gobble\string\{\string" ^^J
9463       :close \string"\expandafter\@gobble\string\}\string" ^^J
9464       :attr \string"#1\string"))}}
```

Only has an effect before `\writeist`:

```
9465 \fi
```

**sAddXdyCounters**

```
9466 \renewcommand*\GlsAddXdyCounters[1]{%
9467   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9468     in compatibility mode.}%
9469 }
```

Add predefined attributes

```
9470 \GlsAddXdyAttribute{glsnumberformat}
9471 \GlsAddXdyAttribute{textrm}
9472 \GlsAddXdyAttribute{textsf}
9473 \GlsAddXdyAttribute{texttt}
9474 \GlsAddXdyAttribute{textbf}
9475 \GlsAddXdyAttribute{textmd}
9476 \GlsAddXdyAttribute{textit}
9477 \GlsAddXdyAttribute{textup}
9478 \GlsAddXdyAttribute{textsl}
```

```

9479 \GlsAddXdyAttribute{textsc}
9480 \GlsAddXdyAttribute{emph}
9481 \GlsAddXdyAttribute{glshypernumber}
9482 \GlsAddXdyAttribute{hyperrm}
9483 \GlsAddXdyAttribute{hypersf}
9484 \GlsAddXdyAttribute{hypertt}
9485 \GlsAddXdyAttribute{hyperbf}
9486 \GlsAddXdyAttribute{hypermd}
9487 \GlsAddXdyAttribute{hyperit}
9488 \GlsAddXdyAttribute{hyperup}
9489 \GlsAddXdyAttribute{hypersl}
9490 \GlsAddXdyAttribute{hypersc}
9491 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9492 \ifglxindy
9493   \renewcommand*{\GlsAddXdyLocation}[2]{%
9494     \edef\@xdyuserlocationdefs{%
9495       \@xdyuserlocationdefs ^^J%
9496       (define-location-class \string"#1\string"^^J\space\space
9497       \space(#2))
9498     }%
9499     \edef\@xdyuserlocationnames{%
9500       \@xdyuserlocationnames^^J\space\space\space
9501       \string"#1\string"}%
9502   }
9503 \fi

```

\@do@wrglossary

```

9504 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9505 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9506   \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9507   \def\@glo@range{}%
9508   \expandafter\if\@glo@prefix(\relax
9509     \def\@glo@range{:open-range}%
9510   \else
9511     \expandafter\if\@glo@prefix)\relax
9512     \def\@glo@range{:close-range}%
9513   \fi
9514 \fi

  Get the location and escape any special characters
9515   \protected@edef\@glslocref{\theglentrycounter}%
9516   \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9517   \glossary[\csname glo@#1@type\endcsname]{%

```

```

9518 (indexentry :tkey (\csname glo@#1@index\endcsname)
9519   :locoref \string"\@glslocoref\string" %
9520   :attr \string"\@glo@suffix\string" \@glo@range
9521 )
9522 }%
9523 \else

```

Convert the format information into the format required for makeindex

```

9524 \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat

```

Write to the glossary file using makeindex syntax.

```

9525 \glossary[\csname glo@#1@type\endcsname]{%
9526 \string\glossaryentry{\csname glo@#1@index\endcsname
9527   \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9528 \fi
9529 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9530 \def\@set@glo@numformat#1#2#3{%
9531   \expandafter\@glo@check@mkidxrangechar#3\@nil
9532   \protected@edef#1{%
9533     \@glo@prefix setentrycounter[] {#2}%
9534     \expandafter\string\csname\@glo@suffix\endcsname
9535   }%
9536   \@gls@checkmkidxchars#1%
9537 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9538 \ifglxsindy
9539   \def\writeist{%
9540     \openout\glswrite=\istfilename
9541     \write\glswrite{;; xindy style file created by the glossaries
9542       package in compatible-2.07 mode}%
9543     \write\glswrite{;; for document '\jobname' on
9544       \the\year-\the\month-\the\day}%
9545     \write\glswrite{^^J; required styles^^J}
9546     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9547       \ifx\@xdystyle\@empty
9548       \else
9549         \protected@write\glswrite{{(require
9550           \string"\@xdystyle.xdy\string")}}%
9551       \fi
9552     }%
9553     \write\glswrite{^^J%
9554       ; list of allowed attributes (number formats)^^J}%
9555     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9556     \write\glswrite{^^J; user defined alphabets^^J}%
9557     \write\glswrite{\@xdyuseralphabets}%
9558     \write\glswrite{^^J; location class definitions^^J}%
9559     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9560     \string"roman-numbers-lowercase\string" :sep \string"}}%
9561 \@onelevel@sanitize\@gls@roman
9562 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9563     :sep \string"}}%
9564 \@onelevel@sanitize\@tmp
9565 \ifx\@tmp\@gls@roman
9566     \write\glswrite{(define-location-class
9567         \string"roman-page-numbers\string"^^J\space\space\space
9568         (\string"roman-numbers-lowercase\string")
9569         :min-range-length \@glsminrange)}}%
9570 \else
9571     \write\glswrite{(define-location-class
9572         \string"roman-page-numbers\string"^^J\space\space\space
9573         (:sep "\@gls@roman")
9574         :min-range-length \@glsminrange)}}%
9575 \fi
9576 \write\glswrite{(define-location-class
9577     \string"Roman-page-numbers\string"^^J\space\space\space
9578     (\string"roman-numbers-uppercase\string")
9579     :min-range-length \@glsminrange)}}%
9580 \write\glswrite{(define-location-class
9581     \string"arabic-page-numbers\string"^^J\space\space\space
9582     (\string"arabic-numbers\string")
9583     :min-range-length \@glsminrange)}}%
9584 \write\glswrite{(define-location-class
9585     \string"alpha-page-numbers\string"^^J\space\space\space
9586     (\string"alpha\string")
9587     :min-range-length \@glsminrange)}}%
9588 \write\glswrite{(define-location-class
9589     \string"Alpha-page-numbers\string"^^J\space\space\space
9590     (\string"ALPHA\string")
9591     :min-range-length \@glsminrange)}}%
9592 \write\glswrite{(define-location-class
9593     \string"Appendix-page-numbers\string"^^J\space\space\space
9594     (\string"ALPHA\string"
9595     :sep \string"\@glsAlphacompositor\string"
9596     \string"arabic-numbers\string")
9597     :min-range-length \@glsminrange)}}%
9598 \write\glswrite{(define-location-class
9599     \string"arabic-section-numbers\string"^^J\space\space\space
9600     (\string"arabic-numbers\string"
9601     :sep \string"\glscompositor\string"
9602     \string"arabic-numbers\string")
9603     :min-range-length \@glsminrange)}}%
9604 \write\glswrite{^^J; user defined location classes}%
9605 \write\glswrite{\@xdyuserlocationdefs}%
9606 \write\glswrite{^^J; define cross-reference class^^J}%
9607 \write\glswrite{(define-crossref-class \string"see\string"
9608     :unverified )}%

```

```

9609 \write\glswrite{(markup-crossref-list
9610 :class \string"see\string"^^J\space\space\space
9611 :open \string"\string\glseeformat\string"
9612 :close \string"{}\string")}%
9613 \write\glswrite{^^J; define the order of the location classes}%
9614 \write\glswrite{(define-location-class-order
9615 (\@xdylocationclassorder))}%
9616 \write\glswrite{^^J; define the glossary markup^^J}%
9617 \write\glswrite{(markup-index^^J\space\space\space
9618 :open \string"\string
9619 \glossarysection[\string\glossarytoctitle]{\string
9620 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9621 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9622 \space\space:close \string"\expandafter\@gobble
9623 \string%\string~n\string
9624 \end{theglossary}\string\glossarypostamble
9625 \string~n\string" ^^J\space\space\space
9626 :tree)}}%
9627 \write\glswrite{(markup-letter-group-list
9628 :sep \string"\string\glsgroupskip\string~n\string")}%
9629 \write\glswrite{(markup-indexentry
9630 :open \string"\string\relax \string\glresetentrylist
9631 \string~n\string")}%
9632 \write\glswrite{(markup-locclass-list :open
9633 \string"\glsoopenbrace\string\glossaryentrynumbers
9634 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9635 :sep \string", \string"
9636 :close \string"\glsclosebrace\glsclosebrace\string")}%
9637 \write\glswrite{(markup-locref-list
9638 :sep \string"\string\delimN\space\string")}%
9639 \write\glswrite{(markup-range
9640 :sep \string"\string\delimR\space\string")}%
9641 \@onelevel@sanitize\gls@suffixF
9642 \@onelevel@sanitize\gls@suffixFF
9643 \ifx\gls@suffixF\@empty
9644 \else
9645 \write\glswrite{(markup-range
9646 :close "\gls@suffixF" :length 1 :ignore-end)}%
9647 \fi
9648 \ifx\gls@suffixFF\@empty
9649 \else
9650 \write\glswrite{(markup-range
9651 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9652 \fi
9653 \write\glswrite{^^J; define format to use for locations^^J}%
9654 \write\glswrite{\@xdylocref}%
9655 \write\glswrite{^^J; define letter group list format^^J}%
9656 \write\glswrite{(markup-letter-group-list
9657 :sep \string"\string\glsgroupskip\string~n\string")}%

```



```

9658 \write\glswrite{^^J; letter group headings^^J}%
9659 \write\glswrite{(markup-letter-group
9660   :open-head \string"\string\glsgroupheading
9661   \glsoopenbrace\string"^^J\space\space\space
9662   :close-head \string"\glsclosebrace\string")}%
9663 \write\glswrite{^^J; additional letter groups^^J}%
9664 \write\glswrite{\@xdylettergroups}%
9665 \write\glswrite{^^J; additional sort rules^^J}
9666 \write\glswrite{\@xdysortrules}%
9667 \noist}
9668 \else
9669 \edef\@gls@actualchar{\string?}
9670 \edef\@gls@encapchar{\string|}
9671 \edef\@gls@levelchar{\string!}
9672 \edef\@gls@quotechar{\string"}
9673 \def\writeist{\relax
9674   \openout\glswrite=\istfilename
9675   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9676     created by the glossaries package}
9677   \write\glswrite{\expandafter\@gobble\string\% for document
9678     '\jobname' on \the\year-\the\month-\the\day}
9679   \write\glswrite{actual '\@gls@actualchar'}
9680   \write\glswrite{encap '\@gls@encapchar'}
9681   \write\glswrite{level '\@gls@levelchar'}
9682   \write\glswrite{quote '\@gls@quotechar'}
9683   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9684   \write\glswrite{preamble \string"\string\glossarysection[\string
9685     \glossarytoctitle]{\string\glossarytitle}\string
9686     \glossarypreamble\string\n\string\begin{theglossary}\string
9687     \glossaryheader\string\n\string"}
9688   \write\glswrite{postamble \string"\string%\string\n\string
9689     \end{theglossary}\string\glossarypostamble\string\n
9690     \string"}
9691   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9692     \string"}
9693   \write\glswrite{item_0 \string"\string%\string\n\string"}
9694   \write\glswrite{item_1 \string"\string%\string\n\string"}
9695   \write\glswrite{item_2 \string"\string%\string\n\string"}
9696   \write\glswrite{item_01 \string"\string%\string\n\string"}
9697   \write\glswrite{item_x1
9698     \string"\string\relax \string\glsresetentrylist\string\n
9699     \string"}
9700   \write\glswrite{item_12 \string"\string%\string\n\string"}
9701   \write\glswrite{item_x2
9702     \string"\string\relax \string\glsresetentrylist\string\n
9703     \string"}
9704   \write\glswrite{delim_0 \string"\string\{\string
9705     \glossaryentrynumbers\string\{\string\relax \string"}
9706   \write\glswrite{delim_1 \string"\string\{\string

```

```

9707      \glossaryentrynumbers\string\{\string\relax \string}
9708      \write\glswrite{delim_2 \string"\string\{\string
9709      \glossaryentrynumbers\string\{\string\relax \string}
9710      \write\glswrite{delim_t \string"\string\}\string\}\string}
9711      \write\glswrite{delim_n \string"\string\delimN \string}
9712      \write\glswrite{delim_r \string"\string\delimR \string}
9713      \write\glswrite{headings_flag 1}
9714      \write\glswrite{heading_prefix
9715          \string"\string\glsgroupheading\string\{\string}
9716      \write\glswrite{heading_suffix
9717          \string"\string\}\string\relax
9718          \string\glsgroupresetentrylist \string}
9719      \write\glswrite{symhead_positive \string"glssymbols\string}
9720      \write\glswrite{numhead_positive \string"glslnumbers\string}
9721      \write\glswrite{page_compositor \string"glscpositor\string}
9722      \@gls@escbsdq\gls@suffixF
9723      \@gls@escbsdq\gls@suffixFF
9724      \ifx\gls@suffixF\@empty
9725      \else
9726          \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9727      \fi
9728      \ifx\gls@suffixFF\@empty
9729      \else
9730          \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9731      \fi
9732      \noist
9733  }
9734 \fi

```

\noist

```

9735 \renewcommand*{\noist}{\let\writeist\relax}

```

## 4.2 glossaries-compatible-307

```

9736 \NeedsTeXFormat{LaTeX2e}
9737 \ProvidesPackage{glossaries-compatible-307}[2017/06/11 v4.30 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9738 \newcommand{\compatglossarystyle}[2]{%
9739   \ifcsundef{@glscompstyle@#1}%
9740   {%
9741     \csdef{@glscompstyle@#1}{#2}%
9742   }%
9743   {%
9744     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9745   }%
9746 }

```

Backward compatible inline style.

```

9747 \compatglossarystyle{inline}{%
9748   \renewcommand{\glossaryentryfield}[5]{%
9749     \glsinlinedopostchild
9750     \gls@inlinesep
9751     \def\glo@desc{##3}%
9752     \def\@no@post@desc{\nopostdesc}%
9753     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
9754     \ifx\glo@desc\@no@post@desc
9755       \glsinlineemptydescformat{##4}{##5}%
9756     \else
9757       \ifstrempy{##3}%
9758         {\glsinlineemptydescformat{##4}{##5}}%
9759         {\glsinlinedescformat{##3}{##4}{##5}}%
9760     \fi
9761     \ifglshaschildren{##1}%
9762     {%
9763       \glsresetsubentrycounter
9764       \glsinlineparentchildseparator
9765       \def\gls@inlinesubsep{}%
9766       \def\gls@inlinepostchild{\glsinlinepostchild}%
9767     }%
9768   }%
9769   \def\gls@inlinesep{\glsinlineseparator}%
9770 }%

```

Sub-entries display description:

```

9771 \renewcommand{\glossarysubentryfield}[6]{%
9772   \gls@inlinesubsep%
9773   \glsinlinesubnameformat{##2}{##3}%
9774   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9775   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9776 }%
9777 }

```

Backward compatible list style.

```

9778 \compatglossarystyle{list}{%
9779   \renewcommand*{\glossaryentryfield}[5]{%
9780     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
9781     ##3\glspostdescription\space ##5}%

```

Sub-entries continue on the same line:

```

9782 \renewcommand*{\glossarysubentryfield}[6]{%
9783   \glssubentryitem{##2}%
9784   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9785 }

```

Backward compatible listgroup style.

```

9786 \compatglossarystyle{listgroup}{%
9787   \csuse{@glscompstyle@list}%
9788 }%

```

Backward compatible listhypergroup style.

```
9789 \compatglossarystyle{listhypergroup}{%
9790 \csuse{@glscompstyle@list}%
9791 }%
```

Backward compatible altlist style.

```
9792 \compatglossarystyle{altlist}{%
9793 \renewcommand*{\glossaryentryfield}[5]{%
9794 \item[\glstentryitem{##1}\glstarget{##1}{##2}]%
9795 \mbox{}\par\nobreak\@afterheading
9796 ##3\glspostdescription\space ##5}%
9797 \renewcommand{\glossarysubentryfield}[6]{%
9798 \par
9799 \glssubentryitem{##2}%
9800 \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9801 }%
```

Backward compatible altlistgroup style.

```
9802 \compatglossarystyle{altlistgroup}{%
9803 \csuse{@glscompstyle@altlist}%
9804 }%
```

Backward compatible altlisthypergroup style.

```
9805 \compatglossarystyle{altlisthypergroup}{%
9806 \csuse{@glscompstyle@altlist}%
9807 }%
```

Backward compatible listdotted style.

```
9808 \compatglossarystyle{listdotted}{%
9809 \renewcommand*{\glossaryentryfield}[5]{%
9810 \item[\makebox[\glslistdottedwidth][l]{%
9811 \glstentryitem{##1}\glstarget{##1}{##2}%
9812 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9813 \renewcommand*{\glossarysubentryfield}[6]{%
9814 \item[\makebox[\glslistdottedwidth][l]{%
9815 \glssubentryitem{##2}%
9816 \glstarget{##2}{##3}%
9817 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9818 }%
```

Backward compatible sublistdotted style.

```
9819 \compatglossarystyle{sublistdotted}{%
9820 \csuse{@glscompstyle@listdotted}%
9821 \renewcommand*{\glossaryentryfield}[5]{%
9822 \item[\glstentryitem{##1}\glstarget{##1}{##2}]}%
9823 }%
```

Backward compatible long style.

```
9824 \compatglossarystyle{long}{%
9825 \renewcommand*{\glossaryentryfield}[5]{%
9826 \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9827 \renewcommand*{\glossarysubentryfield}[6]{%
9828 \glssubentryitem{##2} & ##3\glspostdescription\space ##5\\}%
9829 }%
```

```

9828      &
9829      \glssubentryitem{##2}%
9830      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9831 }%

```

Backward compatible longborder style.

```

9832 \compatglossarystyle{longborder}{%
9833   \csuse{@glscmpstyle@long}%
9834 }%

```

Backward compatible longheader style.

```

9835 \compatglossarystyle{longheader}{%
9836   \csuse{@glscmpstyle@long}%
9837 }%

```

Backward compatible longheaderborder style.

```

9838 \compatglossarystyle{longheaderborder}{%
9839   \csuse{@glscmpstyle@long}%
9840 }%

```

Backward compatible long3col style.

```

9841 \compatglossarystyle{long3col}{%
9842   \renewcommand*{\glossaryentryfield}[5]{%
9843     \glstarget{##1}{\strut}##2 & ##3 & ##5\\}%
9844   \renewcommand*{\glossarysubentryfield}[6]{%
9845     &
9846     \glssubentryitem{##2}%
9847     \glstarget{##2}{\strut}##4 & ##6\\}%
9848 }%

```

Backward compatible long3colborder style.

```

9849 \compatglossarystyle{long3colborder}{%
9850   \csuse{@glscmpstyle@long3col}%
9851 }%

```

Backward compatible long3colheader style.

```

9852 \compatglossarystyle{long3colheader}{%
9853   \csuse{@glscmpstyle@long3col}%
9854 }%

```

Backward compatible long3colheaderborder style.

```

9855 \compatglossarystyle{long3colheaderborder}{%
9856   \csuse{@glscmpstyle@long3col}%
9857 }%

```

Backward compatible long4col style.

```

9858 \compatglossarystyle{long4col}{%
9859   \renewcommand*{\glossaryentryfield}[5]{%
9860     \glstarget{##1}{\strut}##2 & ##3 & ##4 & ##5\\}%
9861   \renewcommand*{\glossarysubentryfield}[6]{%
9862     &
9863     \glssubentryitem{##2}%

```

9864 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%  
 9865 }%

Backward compatible long4colheader style.

9866 \compatglossarystyle{long4colheader}{%  
 9867 \csuse{@glscompstyle@long4col}%  
 9868 }%

Backward compatible long4colborder style.

9869 \compatglossarystyle{long4colborder}{%  
 9870 \csuse{@glscompstyle@long4col}%  
 9871 }%

Backward compatible long4colheaderborder style.

9872 \compatglossarystyle{long4colheaderborder}{%  
 9873 \csuse{@glscompstyle@long4col}%  
 9874 }%

Backward compatible altlong4col style.

9875 \compatglossarystyle{altlong4col}{%  
 9876 \csuse{@glscompstyle@long4col}%  
 9877 }%

Backward compatible altlong4colheader style.

9878 \compatglossarystyle{altlong4colheader}{%  
 9879 \csuse{@glscompstyle@long4col}%  
 9880 }%

Backward compatible altlong4colborder style.

9881 \compatglossarystyle{altlong4colborder}{%  
 9882 \csuse{@glscompstyle@long4col}%  
 9883 }%

Backward compatible altlong4colheaderborder style.

9884 \compatglossarystyle{altlong4colheaderborder}{%  
 9885 \csuse{@glscompstyle@long4col}%  
 9886 }%

Backward compatible long style.

9887 \compatglossarystyle{longragged}{%  
 9888 \renewcommand\*{\glossaryentryfield}[5]{%  
 9889 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%  
 9890 \tabularnewline}%  
 9891 \renewcommand\*{\glossarysubentryfield}[6]{%  
 9892 &  
 9893 \glssubentryitem{##2}%  
 9894 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%  
 9895 \tabularnewline}%  
 9896 }%

Backward compatible longraggedborder style.

9897 \compatglossarystyle{longraggedborder}{%  
 9898 \csuse{@glscompstyle@longragged}%  
 9899 }%

Backward compatible longraggedheader style.

```
9900 \compatglossarystyle{longraggedheader}{%
9901 \csuse{@glscompstyle@longragged}%
9902 }%
```

Backward compatible longraggedheaderborder style.

```
9903 \compatglossarystyle{longraggedheaderborder}{%
9904 \csuse{@glscompstyle@longragged}%
9905 }%
```

Backward compatible longragged3col style.

```
9906 \compatglossarystyle{longragged3col}{%
9907 \renewcommand*{\glossaryentryfield}[5]{%
9908 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
9909 \renewcommand*{\glossarysubentryfield}[6]{%
9910 &
9911 \glssubentryitem{##2}%
9912 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
9913 }%
```

Backward compatible longragged3colborder style.

```
9914 \compatglossarystyle{longragged3colborder}{%
9915 \csuse{@glscompstyle@longragged3col}%
9916 }%
```

Backward compatible longragged3colheader style.

```
9917 \compatglossarystyle{longragged3colheader}{%
9918 \csuse{@glscompstyle@longragged3col}%
9919 }%
```

Backward compatible longragged3colheaderborder style.

```
9920 \compatglossarystyle{longragged3colheaderborder}{%
9921 \csuse{@glscompstyle@longragged3col}%
9922 }%
```

Backward compatible altlongragged4col style.

```
9923 \compatglossarystyle{altlongragged4col}{%
9924 \renewcommand*{\glossaryentryfield}[5]{%
9925 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
9926 \renewcommand*{\glossarysubentryfield}[6]{%
9927 &
9928 \glssubentryitem{##2}%
9929 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
9930 }%
```

Backward compatible altlongragged4colheader style.

```
9931 \compatglossarystyle{altlongragged4colheader}{%
9932 \csuse{@glscompstyle@altlong4col}%
9933 }%
```

Backward compatible altlongragged4colborder style.

```
9934 \compatglossarystyle{altlongragged4colborder}{%
```

```

9935 \csuse{@glscompstyle@altlong4col}%
9936 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9937 \compatglossarystyle{altlongragged4colheaderborder}{%
9938 \csuse{@glscompstyle@altlong4col}%
9939 }%

```

Backward compatible index style.

```

9940 \compatglossarystyle{index}{%
9941 \renewcommand*{\glossaryentryfield}[5]{%
9942 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9943 \ifx\relax##4\relax
9944 \else
9945 \space{##4}%
9946 \fi
9947 \space ##3\glspostdescription \space ##5}%
9948 \renewcommand*{\glossarysubentryfield}[6]{%
9949 \ifcase##1\relax
9950 % level 0
9951 \item
9952 \or
9953 % level 1
9954 \subitem
9955 \glssubentryitem{##2}%
9956 \else
9957 % all other levels
9958 \subsubitem
9959 \fi
9960 \textbf{\glstarget{##2}{##3}}%
9961 \ifx\relax##5\relax
9962 \else
9963 \space{##5}%
9964 \fi
9965 \space##4\glspostdescription\space ##6}%
9966 }%

```

Backward compatible indexgroup style.

```

9967 \compatglossarystyle{indexgroup}{%
9968 \csuse{@glscompstyle@index}%
9969 }%

```

Backward compatible indexhypergroup style.

```

9970 \compatglossarystyle{indexhypergroup}{%
9971 \csuse{@glscompstyle@index}%
9972 }%

```

Backward compatible tree style.

```

9973 \compatglossarystyle{tree}{%
9974 \renewcommand{\glossaryentryfield}[5]{%
9975 \hangindent0pt\relax

```



```

9976 \parindent0pt\relax
9977 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9978 \ifx\relax##4\relax
9979 \else
9980 \space(##4)%
9981 \fi
9982 \space ##3\glspostdescription \space ##5\par}%
9983 \renewcommand{\glossarysubentryfield}[6]{%
9984 \hangindent##1\glstreeindent\relax
9985 \parindent##1\glstreeindent\relax
9986 \ifnum##1=1\relax
9987 \glssubentryitem{##2}%
9988 \fi
9989 \textbf{\glstarget{##2}{##3}}%
9990 \ifx\relax##5\relax
9991 \else
9992 \space(##5)%
9993 \fi
9994 \space##4\glspostdescription\space ##6\par}%
9995 }%

```

Backward compatible treegroup style.

```

9996 \compatglossarystyle{treegroup}{%
9997 \csuse{@glscmpstyle@tree}%
9998 }%

```

Backward compatible treehypergroup style.

```

9999 \compatglossarystyle{treehypergroup}{%
10000 \csuse{@glscmpstyle@tree}%
10001 }%

```

Backward compatible treenoname style.

```

10002 \compatglossarystyle{treenoname}{%
10003 \renewcommand{\glossaryentryfield}[5]{%
10004 \hangindent0pt\relax
10005 \parindent0pt\relax
10006 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
10007 \ifx\relax##4\relax
10008 \else
10009 \space(##4)%
10010 \fi
10011 \space ##3\glspostdescription \space ##5\par}%
10012 \renewcommand{\glossarysubentryfield}[6]{%
10013 \hangindent##1\glstreeindent\relax
10014 \parindent##1\glstreeindent\relax
10015 \ifnum##1=1\relax
10016 \glssubentryitem{##2}%
10017 \fi
10018 \glstarget{##2}{\strut}%
10019 ##4\glspostdescription\space ##6\par}%
10020 }%

```

Backward compatible treenonamegroup style.

```
10021 \compatglossarystyle{treenonamegroup}{%
10022   \csuse{@glscompstyle@treenoname}%
10023 }%
```

Backward compatible treenonamehypergroup style.

```
10024 \compatglossarystyle{treenonamehypergroup}{%
10025   \csuse{@glscompstyle@treenoname}%
10026 }%
```

Backward compatible alttree style.

```
10027 \compatglossarystyle{alttree}{%
10028   \renewcommand{\glossaryentryfield}[5]{%
10029     \ifnum\@gls@prevlevel=0\relax
10030     \else
10031       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
10032       \hangindent\glstreeindent
10033       \parindent\glstreeindent
10034     \fi
10035     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
10036       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
10037     \ifx\relax##4\relax
10038     \else
10039       (##4)\space
10040     \fi
10041     ##3\glspostdescription \space ##5\par
10042     \def\@gls@prevlevel{0}%
10043   }%
10044   \renewcommand{\glossarysubentryfield}[6]{%
10045     \ifnum##1=1\relax
10046       \glssubentryitem{##2}%
10047     \fi
10048     \ifnum\@gls@prevlevel=##1\relax
10049     \else
10050       \@ifundefined{@glswidestname\romannumeral##1}{%
10051         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10052         \settowidth{\gls@tmplen}{\textbf{%
10053           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10054         \ifnum\@gls@prevlevel<##1\relax
10055           \setlength\glstreeindent\gls@tmplen
10056           \addtolength\glstreeindent\parindent
10057           \parindent\glstreeindent
10058         \else
10059           \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10060             \settowidth{\glstreeindent}{\textbf{%
10061               \@glswidestname\space}}{%
10062             \settowidth{\glstreeindent}{\textbf{%
10063               \csname @glswidestname\romannumeral\@gls@prevlevel
10064                 \endcsname\space}}}%
10065             \addtolength\parindent{-\glstreeindent}%

```

```

10066      \setlength\glstreeindent\parindent
10067      \fi
10068      \fi
10069      \hangindent\glstreeindent
10070      \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10071        \textbf{\glstarget{##2}{##3}}}%
10072      \ifx##5\relax\relax
10073      \else
10074        (##5)\space
10075      \fi
10076      ##4\glspostdescription\space ##6\par
10077      \def\@gls@prevlevel{##1}%
10078    }%
10079 }%

```

Backward compatible alttreegroup style.

```

10080 \compatglossarystyle{alttreegroup}{%
10081   \csuse{@glscompstyle@alttree}%
10082 }%

```

Backward compatible alttreehypergroup style.

```

10083 \compatglossarystyle{alttreehypergroup}{%
10084   \csuse{@glscompstyle@alttree}%
10085 }%

```

Backward compatible mcolindex style.

```

10086 \compatglossarystyle{mcolindex}{%
10087   \csuse{@glscompstyle@index}%
10088 }%

```

Backward compatible mcolindexgroup style.

```

10089 \compatglossarystyle{mcolindexgroup}{%
10090   \csuse{@glscompstyle@index}%
10091 }%

```

Backward compatible mcolindexhypergroup style.

```

10092 \compatglossarystyle{mcolindexhypergroup}{%
10093   \csuse{@glscompstyle@index}%
10094 }%

```

Backward compatible mcoltree style.

```

10095 \compatglossarystyle{mcoltree}{%
10096   \csuse{@glscompstyle@tree}%
10097 }%

```

Backward compatible mcoltreegroup style.

```

10098 \compatglossarystyle{mcolindextreegroup}{%
10099   \csuse{@glscompstyle@tree}%
10100 }%

```

Backward compatible mcoltreehypergroup style.

```

10101 \compatglossarystyle{mcolindextreehypergroup}{%

```

```

10102 \csuse{@glscompstyle@tree}%
10103 }%

    Backward compatible mcoltreenoname style.
10104 \compatglossarystyle{mcoltreenoname}{%
10105 \csuse{@glscompstyle@tree}%
10106 }%

    Backward compatible mcoltreenonamegroup style.
10107 \compatglossarystyle{mcoltreenonamegroup}{%
10108 \csuse{@glscompstyle@tree}%
10109 }%

    Backward compatible mcoltreenonamehypergroup style.
10110 \compatglossarystyle{mcoltreenonamehypergroup}{%
10111 \csuse{@glscompstyle@tree}%
10112 }%

    Backward compatible mcolalmtree style.
10113 \compatglossarystyle{mcolalmtree}{%
10114 \csuse{@glscompstyle@almtree}%
10115 }%

    Backward compatible mcolalmtreegroup style.
10116 \compatglossarystyle{mcolalmtreegroup}{%
10117 \csuse{@glscompstyle@almtree}%
10118 }%

    Backward compatible mcolalmtreehypergroup style.
10119 \compatglossarystyle{mcolalmtreehypergroup}{%
10120 \csuse{@glscompstyle@almtree}%
10121 }%

    Backward compatible superragged style.
10122 \compatglossarystyle{superragged}{%
10123 \renewcommand*{\glossaryentryfield}[5]{%
10124 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10125 \tabularnewline}%
10126 \renewcommand*{\glossarysubentryfield}[6]{%
10127 &
10128 \glssubentryitem{##2}%
10129 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10130 \tabularnewline}%
10131 }%

    Backward compatible superraggedborder style.
10132 \compatglossarystyle{superraggedborder}{%
10133 \csuse{@glscompstyle@superragged}%
10134 }%

    Backward compatible superraggedheader style.
10135 \compatglossarystyle{superraggedheader}{%
10136 \csuse{@glscompstyle@superragged}%
10137 }%

```

Backward compatible superraggedheaderborder style.

```
10138 \compatglossarystyle{superraggedheaderborder}{%
10139   \csuse{@glscompstyle@superragged}%
10140 }%
```

Backward compatible superragged3col style.

```
10141 \compatglossarystyle{superragged3col}{%
10142   \renewcommand*{\glossaryentryfield}[5]{%
10143     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10144   \renewcommand*{\glossarysubentryfield}[6]{%
10145     &
10146     \glssubentryitem{##2}%
10147     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10148 }%
```

Backward compatible superragged3colborder style.

```
10149 \compatglossarystyle{superragged3colborder}{%
10150   \csuse{@glscompstyle@superragged3col}%
10151 }%
```

Backward compatible superragged3colheader style.

```
10152 \compatglossarystyle{superragged3colheader}{%
10153   \csuse{@glscompstyle@superragged3col}%
10154 }%
```

Backward compatible superragged3colheaderborder style.

```
10155 \compatglossarystyle{superragged3colheaderborder}{%
10156   \csuse{@glscompstyle@superragged3col}%
10157 }%
```

Backward compatible altsuperragged4col style.

```
10158 \compatglossarystyle{altsuperragged4col}{%
10159   \renewcommand*{\glossaryentryfield}[5]{%
10160     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10161   \renewcommand*{\glossarysubentryfield}[6]{%
10162     &
10163     \glssubentryitem{##2}%
10164     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10165 }%
```

Backward compatible altsuperragged4colheader style.

```
10166 \compatglossarystyle{altsuperragged4colheader}{%
10167   \csuse{@glscompstyle@altsuperragged4col}%
10168 }%
```

Backward compatible altsuperragged4colborder style.

```
10169 \compatglossarystyle{altsuperragged4colborder}{%
10170   \csuse{@glscompstyle@altsuperragged4col}%
10171 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10172 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10173 \csuse{@glscompstyle@altsuperragged4col}%
10174 }%
```

Backward compatible super style.

```
10175 \compatglossarystyle{super}{%
10176   \renewcommand*{\glossaryentryfield}[5]{%
10177     \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10178   \renewcommand*{\glossarysubentryfield}[6]{%
10179     &
10180     \glssubentryitem{##2}%
10181     \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10182 }%
```

Backward compatible superborder style.

```
10183 \compatglossarystyle{superborder}{%
10184   \csuse{@glscompstyle@super}%
10185 }%
```

Backward compatible superheader style.

```
10186 \compatglossarystyle{superheader}{%
10187   \csuse{@glscompstyle@super}%
10188 }%
```

Backward compatible superheaderborder style.

```
10189 \compatglossarystyle{superheaderborder}{%
10190   \csuse{@glscompstyle@super}%
10191 }%
```

Backward compatible super3col style.

```
10192 \compatglossarystyle{super3col}{%
10193   \renewcommand*{\glossaryentryfield}[5]{%
10194     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10195   \renewcommand*{\glossarysubentryfield}[6]{%
10196     &
10197     \glssubentryitem{##2}%
10198     \glstarget{##2}{\strut}##4 & ##6\\}%
10199 }%
```

Backward compatible super3colborder style.

```
10200 \compatglossarystyle{super3colborder}{%
10201   \csuse{@glscompstyle@super3col}%
10202 }%
```

Backward compatible super3colheader style.

```
10203 \compatglossarystyle{super3colheader}{%
10204   \csuse{@glscompstyle@super3col}%
10205 }%
```

Backward compatible super3colheaderborder style.

```
10206 \compatglossarystyle{super3colheaderborder}{%
10207   \csuse{@glscompstyle@super3col}%
10208 }%
```

Backward compatible super4col style.

```
10209 \compatglossarystyle{super4col}{%
10210   \renewcommand*{\glossaryentryfield}[5]{%
10211     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10212   \renewcommand*{\glossarysubentryfield}[6]{%
10213     &
10214     \glssubentryitem{##2}%
10215     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10216 }%
```

Backward compatible super4colheader style.

```
10217 \compatglossarystyle{super4colheader}{%
10218   \csuse{@glscompstyle@super4col}%
10219 }%
```

Backward compatible super4colborder style.

```
10220 \compatglossarystyle{super4colborder}{%
10221   \csuse{@glscompstyle@super4col}%
10222 }%
```

Backward compatible super4colheaderborder style.

```
10223 \compatglossarystyle{super4colheaderborder}{%
10224   \csuse{@glscompstyle@super4col}%
10225 }%
```

Backward compatible altsuper4col style.

```
10226 \compatglossarystyle{altsuper4col}{%
10227   \csuse{@glscompstyle@super4col}%
10228 }%
```

Backward compatible altsuper4colheader style.

```
10229 \compatglossarystyle{altsuper4colheader}{%
10230   \csuse{@glscompstyle@super4col}%
10231 }%
```

Backward compatible altsuper4colborder style.

```
10232 \compatglossarystyle{altsuper4colborder}{%
10233   \csuse{@glscompstyle@super4col}%
10234 }%
```

Backward compatible altsuper4colheaderborder style.

```
10235 \compatglossarystyle{altsuper4colheaderborder}{%
10236   \csuse{@glscompstyle@super4col}%
10237 }%
```

## 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.

```
10238 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10239 \ProvidesPackage{glossaries-accsupp}[2017/06/11 v4.30 (NLCT)]
```

```
10240 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10241 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10242 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10243 \@ifpackageloaded{glossaries-extra}
```

```
10244 {%
```

If the accsupp option was used, \glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10245 \ifx\glsxtr@doaccsupp\empty
```

```
10246 \GlossariesWarning{The 'glossaries-accsupp'
```

```
10247 package has been loaded\MessageBreak
```

```
10248 after the 'glossaries-extra' package. This\MessageBreak
```

```
10249 can cause a failure to integrate both packages. \MessageBreak
```

```
10250 Either use the 'accsupp' option when you load\MessageBreak
```

```
10251 'glossaries-extra' or load 'glossaries-accsupp'\MessageBreak
```

```
10252 before loading 'glossaries-extra'}%
```

```
10253 \fi
```

```
10254 }
```

```
10255 {}
```

tibleglossentry Override style compatibility macros:

```
10256 \def\compatibleglossentry#1#2{%
```

```
10257 \toks@{#2}%
```

```
10258 \protected@edef\do@glossentry{%
```

```
10259 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10260 {\noexpand\glsnamefont
```

```
10261 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```



```

10262     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10263     {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10264     {\the\toks@}%
10265   }%
10266   \@do@glossentry
10267 }

```

lesubglossentry

```

10268 \def\compatiblesubglossentry#1#2#3{%
10269   \toks@{#3}%
10270   \protected@edef\@do@subglossentry{%
10271     \noexpand\accsuppglossarysubentryfield{\number#1}%
10272     {#2}%
10273     {\noexpand\glsnamefont
10274      {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}}%
10275     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10276     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10277     {\the\toks@}%
10278   }%
10279   \@do@subglossentry
10280 }

```

Required packages:

```

10281 \RequirePackage{glossaries}
10282 \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:

```

10283 \define@key{glossentry}{access}{%
10284   \def\@glo@access{#1}%
10285 }

```

**textaccess** The replacement text corresponding to the text key:

```

10286 \define@key{glossentry}{textaccess}{%
10287   \def\@glo@textaccess{#1}%
10288 }

```

**firstaccess** The replacement text corresponding to the first key:

```

10289 \define@key{glossentry}{firstaccess}{%
10290   \def\@glo@firstaccess{#1}%
10291 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10292 \define@key{glossentry}{pluralaccess}{%
10293   \def\@glo@pluralaccess{#1}%
10294 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

10295 \define@key{glossentry}{firstpluralaccess}{%
10296   \def\@glo@firstpluralaccess{#1}%
10297 }
```

symbolaccess The replacement text corresponding to the symbol key:

```

10298 \define@key{glossentry}{symbolaccess}{%
10299   \def\@glo@symbolaccess{#1}%
10300 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```

10301 \define@key{glossentry}{symbolpluralaccess}{%
10302   \def\@glo@symbolpluralaccess{#1}%
10303 }
```

descriptionaccess The replacement text corresponding to the description key:

```

10304 \define@key{glossentry}{descriptionaccess}{%
10305   \def\@glo@descaccess{#1}%
10306 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```

10307 \define@key{glossentry}{descriptionpluralaccess}{%
10308   \def\@glo@descpluralaccess{#1}%
10309 }
```

shortaccess The replacement text corresponding to the short key:

```

10310 \define@key{glossentry}{shortaccess}{%
10311   \def\@glo@shortaccess{#1}%
10312 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```

10313 \define@key{glossentry}{shortpluralaccess}{%
10314   \def\@glo@shortpluralaccess{#1}%
10315 }
```

longaccess The replacement text corresponding to the long key:

```

10316 \define@key{glossentry}{longaccess}{%
10317   \def\@glo@longaccess{#1}%
10318 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```

10319 \define@key{glossentry}{longpluralaccess}{%
10320   \def\@glo@longpluralaccess{#1}%
10321 }
```

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={\glsaccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```

10322 \appto\@gls@keymap{,%
10323   {access}{access},%
10324   {textaccess}{textaccess},%
10325   {firstaccess}{firstaccess},%
10326   {pluralaccess}{pluralaccess},%
10327   {firstpluralaccess}{firstpluralaccess},%
10328   {symbolaccess}{symbolaccess},%
10329   {symbolpluralaccess}{symbolpluralaccess},%
10330   {descaccess}{descaccess},%
10331   {descpluralaccess}{descpluralaccess},%
10332   {shortaccess}{shortaccess},%
10333   {shortpluralaccess}{shortpluralaccess},%
10334   {longaccess}{longaccess},%
10335   {longpluralaccess}{longpluralaccess}%
10336 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```

10337 \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):

```

10338 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10339 \renewcommand*{\@newglossaryentryprehook}{%
10340   \@gls@oldnewglossaryentryprehook
10341   \def\@glo@access{\@glo@symbol}%
```

Initialise the other keys:

```

10342   \def\@glo@textaccess{\@glo@access}%
10343   \def\@glo@firstaccess{\@glo@access}%
10344   \def\@glo@pluralaccess{\@glo@textaccess}%
10345   \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10346   \def\@glo@symbolaccess{\relax}%
10347   \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10348   \def\@glo@descaccess{\relax}%
10349   \def\@glo@descpluralaccess{\@glo@descaccess}%
10350   \def\@glo@shortaccess{\relax}%
10351   \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10352   \def\@glo@longaccess{\relax}%
10353   \def\@glo@longpluralaccess{\@glo@longaccess}%
10354 }
```

Add to the end hook:

```

10355 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10356 \renewcommand*{\@newglossaryentryposthook}{%
10357   \@gls@oldnewglossaryentryposthook
```

Store the access information:

```

10358 \expandafter
10359 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10360 \@glo@access}%
10361 \expandafter
10362 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10363 \@glo@textaccess}%
10364 \expandafter
10365 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10366 \@glo@firstaccess}%
10367 \expandafter
10368 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10369 \@glo@pluralaccess}%
10370 \expandafter
10371 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10372 \@glo@firstpluralaccess}%
10373 \expandafter
10374 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10375 \@glo@symbolaccess}%
10376 \expandafter
10377 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10378 \@glo@symbolpluralaccess}%
10379 \expandafter
10380 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10381 \@glo@descaccess}%
10382 \expandafter
10383 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10384 \@glo@descpluralaccess}%
10385 \expandafter
10386 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10387 \@glo@shortaccess}%
10388 \expandafter
10389 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10390 \@glo@shortpluralaccess}%
10391 \expandafter
10392 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10393 \@glo@longaccess}%
10394 \expandafter
10395 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10396 \@glo@longpluralaccess}%
10397 }

```

## 5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10398 \newcommand*{\glsentryaccess}[1]{%
10399 \@gls@entry@field{#1}{access}%
10400 }

```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
10401 \newcommand*{\glsentrytextaccess}[1]{%
10402   \@gls@entry@field{#1}{textaccess}%
10403 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10404 \newcommand*{\glsentryfirstaccess}[1]{%
10405   \@gls@entry@field{#1}{firstaccess}%
10406 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10407 \newcommand*{\glsentrypluralaccess}[1]{%
10408   \@gls@entry@field{#1}{pluralaccess}%
10409 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10410 \newcommand*{\glsentryfirstpluralaccess}[1]{%
10411   \csname glo@#1@firstpluralaccess\endcsname
10412 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10413 \newcommand*{\glsentrysymbolaccess}[1]{%
10414   \@gls@entry@field{#1}{symbolaccess}%
10415 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10416 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
10417   \@gls@entry@field{#1}{symbolpluralaccess}%
10418 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10419 \newcommand*{\glsentrydescaccess}[1]{%
10420   \@gls@entry@field{#1}{descaccess}%
10421 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10422 \newcommand*{\glsentrydescpluralaccess}[1]{%
10423   \@gls@entry@field{#1}{descaccess}%
10424 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10425 \newcommand*{\glsentryshortaccess}[1]{%
10426   \@gls@entry@field{#1}{shortaccess}%
10427 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10428 \newcommand*{\glsentryshortpluralaccess}[1]{%
10429   \@gls@entry@field{#1}{shortpluralaccess}%
10430 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10431 \newcommand*{\glsentrylongaccess}[1]{%
10432   \@gls@entry@field{#1}{longaccess}%
10433 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10434 \newcommand*{\glsentrylongpluralaccess}[1]{%
10435   \@gls@entry@field{#1}{longpluralaccess}%
10436 }
```

\glsaccsupp \glsaccsupp{<replacement text>}{<text>}

This can be redefined to use E or Alt instead of ActualText. (I don't have the software to test the E or Alt options.)

```
10437 \newcommand*{\glsaccsupp}[2]{%
10438   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10439 }
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
10440 \newcommand*{\xglsaccsupp}[2]{%
10441   \protected@edef\@gls@replacementtext{#1}%
10442   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10443 }
```

@access@display

```
10444 \newcommand*{\@gls@access@display}[2]{%
10445   \protected@edef\@glo@access{#2}%
10446   \ifx\@glo@access\@gls@noaccess
10447     #1%
10448   \else
10449     \xglsaccsupp{\@glo@access}{#1}%
10450   \fi
10451 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10452 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10453   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10454 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10455 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10456   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10457 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10458 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10459   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10460 }
```

staccessdisplay As above but for the firstaccess replacement text.

```
10461 \DeclareRobustCommand*\glfirstaccessdisplay}[2]{%
10462   \@gls@access@display{#1}{\glentryfirstaccess{#2}}%
10463 }
```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```
10464 \DeclareRobustCommand*\glfirstpluralaccessdisplay}[2]{%
10465   \@gls@access@display{#1}{\glentryfirstpluralaccess{#2}}%
10466 }
```

olaccessdisplay As above but for the symbolaccess replacement text.

```
10467 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
10468   \@gls@access@display{#1}{\glentrysymbolaccess{#2}}%
10469 }
```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```
10470 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
10471   \@gls@access@display{#1}{\glentrysymbolpluralaccess{#2}}%
10472 }
```

onaccessdisplay As above but for the descriptionaccess replacement text.

```
10473 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10474   \@gls@access@display{#1}{\glentrydescaccess{#2}}%
10475 }
```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```
10476 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10477   \@gls@access@display{#1}{\glentrydescpluralaccess{#2}}%
10478 }
```

rtaccessdisplay As above but for the shortaccess replacement text.

```
10479 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10480   \@gls@access@display{#1}{\glentryshortaccess{#2}}%
10481 }
```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```
10482 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10483   \@gls@access@display{#1}{\glentryshortpluralaccess{#2}}%
10484 }
```

ngaccessdisplay As above but for the longaccess replacement text.

```
10485 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10486   \@gls@access@display{#1}{\glentrylongaccess{#2}}%
10487 }
```

alaccessdisplay As above but for the longpluralaccess replacement text.

```
10488 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10489   \@gls@access@display{#1}{\glentrylongpluralaccess{#2}}%
10490 }
```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

10491 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
10492   \@ifundefined{gls#1accessdisplay}%
10493   {%
10494     \PackageError{glossaries-accsupp}{No accessibility support
10495       for key ‘#1’}{}%
10496   }%
10497   {%
10498     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10499   }%
10500 }

```

`\default@entryfmt` Redefine the default entry format to use accessibility information

```

10501 \renewcommand*\@@gls@default@entryfmt}[2]{%
10502   \ifdefempty\glscustomtext
10503   {%
10504     \glsifplural
10505     {%

```

Plural form

```

10506     \glscapscase
10507     {%

```

Don't adjust case

```

10508     \ifglsused\glslabel
10509     {%

```

Subsequent use

```

10510       #2{\glspluralaccessdisplay
10511         {\glsentryplural{\glslabel}}{\glslabel}}%
10512       {\glsdescriptionpluralaccessdisplay
10513         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10514       {\glsymbolpluralaccessdisplay
10515         {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10516       {\glsinsert}%
10517     }%
10518     {%

```

First use

```

10519       #1{\glsfirstpluralaccessdisplay
10520         {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10521       {\glsdescriptionpluralaccessdisplay
10522         {\glsentrydescplural{\glslabel}}{\glslabel}}%
10523       {\glsymbolpluralaccessdisplay
10524         {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10525       {\glsinsert}%
10526     }%
10527     }%
10528     {%

```



Make first letter upper case

```
10529      \ifglsused\glslabel
10530      {%
```

Subsequent use.

```
10531      #2{\glspluralaccessdisplay
10532          {\Glsentryplural{\glslabel}}{\glslabel}}%
10533          {\glsdescriptionpluralaccessdisplay
10534              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10535          {\glssymbolpluralaccessdisplay
10536              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10537          {\glsinsert}}%
10538      }%
10539      {%
```

First use

```
10540      #1{\glsfirstpluralaccessdisplay
10541          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10542          {\glsdescriptionpluralaccessdisplay
10543              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10544          {\glssymbolpluralaccessdisplay
10545              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10546          {\glsinsert}}%
10547      }%
10548      }%
10549      {%
```

Make all upper case

```
10550      \ifglsused\glslabel
10551      {%
```

Subsequent use

```
10552      \MakeUppercase{%
10553      #2{\glspluralaccessdisplay
10554          {\glsentryplural{\glslabel}}{\glslabel}}%
10555          {\glsdescriptionpluralaccessdisplay
10556              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10557          {\glssymbolpluralaccessdisplay
10558              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10559          {\glsinsert}}%
10560      }%
10561      {%
```

First use

```
10562      \MakeUppercase{%
10563      #1{\glsfirstpluralaccessdisplay
10564          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10565          {\glsdescriptionpluralaccessdisplay
10566              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10567          {\glssymbolpluralaccessdisplay
10568              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10569         {\glsinsert}}}%
10570     }%
10571 }%
10572 }%
10573 {%

```

#### Singular form

```

10574     \glscapscase
10575     {%

```

#### Don't adjust case

```

10576     \ifglsused\glslabel
10577     {%

```

#### Subsequent use

```

10578     #2{\glstextaccessdisplay
10579         {\glsentrytext{\glslabel}}{\glslabel}}%
10580     {\glsdescriptionaccessdisplay
10581         {\glsentrydesc{\glslabel}}{\glslabel}}%
10582     {\glssymbolaccessdisplay
10583         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10584     {\glsinsert}}%
10585 }%
10586 {%

```

#### First use

```

10587     #1{\glsfirstaccessdisplay
10588         {\glsentryfirst{\glslabel}}{\glslabel}}%
10589     {\glsdescriptionaccessdisplay
10590         {\glsentrydesc{\glslabel}}{\glslabel}}%
10591     {\glssymbolaccessdisplay
10592         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10593     {\glsinsert}}%
10594 }%
10595 }%
10596 {%

```

#### Make first letter upper case

```

10597     \ifglsused\glslabel
10598     {%

```

#### Subsequent use

```

10599     #2{\glstextaccessdisplay
10600         {\Glsentrytext{\glslabel}}{\glslabel}}%
10601     {\glsdescriptionaccessdisplay
10602         {\Glsentrydesc{\glslabel}}{\glslabel}}%
10603     {\glssymbolaccessdisplay
10604         {\Glsentrysymbol{\glslabel}}{\glslabel}}%
10605     {\glsinsert}}%
10606 }%
10607 {%

```

#### First use

```

10608      #1{\glsfirstaccessdisplay
10609          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10610          {\glsdescriptionaccessdisplay
10611              {\glsentrydesc{\glslabel}}{\glslabel}}%
10612          {\glsymbolaccessdisplay
10613              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10614          {\glsinsert}}%
10615      }%
10616  }%
10617  {%

```

#### Make all upper case

```

10618      \ifglsused\glslabel
10619      {%

```

#### Subsequent use

```

10620      \MakeUppercase{%
10621          #2{\glstextaccessdisplay
10622              {\glsentrytext{\glslabel}}{\glslabel}}%
10623              {\glsdescriptionaccessdisplay
10624                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10625                  {\glsymbolaccessdisplay
10626                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10627                      {\glsinsert}}}%
10628      }%
10629  {%

```

#### First use

```

10630      \MakeUppercase{%
10631          #1{\glsfirstaccessdisplay
10632              {\glsentryfirst{\glslabel}}{\glslabel}}%
10633              {\glsdescriptionaccessdisplay
10634                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10635                  {\glsymbolaccessdisplay
10636                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10637                      {\glsinsert}}}%
10638      }%
10639  }%
10640  }%
10641  }%
10642  {%

```

#### Custom text provided in \glsdisp

```

10643      \ifglsused{\glslabel}%
10644      {%

```

#### Subsequent use

```

10645      #2{\glscustomtext}%
10646      {\glsdescriptionaccessdisplay
10647          {\glsentrydesc{\glslabel}}{\glslabel}}%

```

```

10648      {\glssymbolaccessdisplay
10649      {\glentrysymbol{\glslabel}}{\glslabel}}%
10650      {\glsinsert}}%
10651  }%
10652  {%

```

First use

```

10653      #1{\glscustomtext}}%
10654      {\glsdescriptionaccessdisplay
10655      {\glentrydesc{\glslabel}}{\glslabel}}%
10656      {\glssymbolaccessdisplay
10657      {\glentrysymbol{\glslabel}}{\glslabel}}%
10658      {\glsinsert}}%
10659  }%
10660  }%
10661 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10662 \renewcommand*{\glsgenentryfmt}{%
10663   \ifdefempty\glscustomtext
10664   {%
10665     \glsifplural
10666     {%

```

Plural form

```

10667     \glscapscase
10668     {%

```

Don't adjust case

```

10669     \ifglused\glslabel
10670     {%

```

Subsequent use

```

10671     \glspluralaccessdisplay
10672     {\glentryplural{\glslabel}}{\glslabel}}%
10673     \glsinsert
10674   }%
10675   {%

```

First use

```

10676     \glsfirstpluralaccessdisplay
10677     {\glentryfirstplural{\glslabel}}{\glslabel}}%
10678     \glsinsert
10679   }%
10680   }%
10681   {%

```

Make first letter upper case

```

10682     \ifglused\glslabel
10683     {%

```

Subsequent use.

```
10684      \glspluralaccessdisplay
10685      {\Glsentryplural{\glslabel}}{\glslabel}%
10686      \glsinsert
10687      }%
10688      {%
```

First use

```
10689      \glsfirstpluralaccessdisplay
10690      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10691      \glsinsert
10692      }%
10693      }%
10694      {%
```

Make all upper case

```
10695      \ifglused\glslabel
10696      {%
```

Subsequent use

```
10697      \glspluralaccessdisplay
10698      {\mfirstucMakeUppercase{\Glsentryplural{\glslabel}}}%
10699      {\glslabel}%
10700      \mfirstucMakeUppercase{\glsinsert}%
10701      }%
10702      {%
```

First use

```
10703      \glsfirstpluralaccessdisplay
10704      {\mfirstucMakeUppercase{\Glsentryfirstplural{\glslabel}}}%
10705      {\glslabel}%
10706      \mfirstucMakeUppercase{\glsinsert}%
10707      }%
10708      }%
10709      }%
10710      {%
```

Singular form

```
10711      \glscapscase
10712      {%
```

Don't adjust case

```
10713      \ifglused\glslabel
10714      {%
```

Subsequent use

```
10715      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10716      \glsinsert
10717      }%
10718      {%
```

#### First use

```

10719      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10720      \glsinsert
10721      }%
10722      }%
10723      {%

```

#### Make first letter upper case

```

10724      \ifglsused\glslabel
10725      {%

```

#### Subsequent use

```

10726      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10727      \glsinsert
10728      }%
10729      {%

```

#### First use

```

10730      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10731      \glsinsert
10732      }%
10733      }%
10734      {%

```

#### Make all upper case

```

10735      \ifglsused\glslabel
10736      {%

```

#### Subsequent use

```

10737      \glstextaccessdisplay
10738      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10739      \mfirstucMakeUppercase{\glsinsert}%
10740      }%
10741      {%

```

#### First use

```

10742      \glsfirstaccessdisplay
10743      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10744      \mfirstucMakeUppercase{\glsinsert}%
10745      }%
10746      }%
10747      }%
10748      }%
10749      {%

```

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.

```

10750      \glscustomtext\glsinsert
10751      }%
10752      }

```

`\glsgenacfmt`   Redefine to include accessibility information.

```
10753 \renewcommand*{\glsgenacfmt}{%
10754   \ifdefempty\glscustomtext
10755     {%
10756       \ifglused\glslabel
10757       {%
```

Subsequent use:

```
10758     \glsifplural
10759     {%
```

Subsequent plural form:

```
10760     \glscapscase
10761     {%
```

Subsequent plural form, don't adjust case:

```
10762     \acronymfont
10763     {\glsshortpluralaccessdisplay
10764       {\glentryshortpl{\glslabel}}{\glslabel}}%
10765     \glsinsert
10766   }%
10767   {%
```

Subsequent plural form, make first letter upper case:

```
10768     \acronymfont
10769     {\glsshortpluralaccessdisplay
10770       {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10771     \glsinsert
10772   }%
10773   {%
```

Subsequent plural form, all caps:

```
10774     \mfirstucMakeUppercase
10775     {\acronymfont
10776       {\glsshortpluralaccessdisplay
10777         {\glentryshortpl{\glslabel}}{\glslabel}}%
10778       \glsinsert}%
10779   }%
10780   }%
10781   {%
```

Subsequent singular form

```
10782     \glscapscase
10783     {%
```

Subsequent singular form, don't adjust case:

```
10784     \acronymfont
10785     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10786     \glsinsert
10787   }%
10788   {%
```

Subsequent singular form, make first letter upper case:

```
10789      \acronymfont
10790      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10791      \glsinsert
10792      }%
10793      {%
```

Subsequent singular form, all caps:

```
10794      \mfirstucMakeUppercase
10795      {\acronymfont{%
10796      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10797      \glsinsert}%
10798      }%
10799      }%
10800      }%
10801      {%
```

First use:

```
10802      \glsifplural
10803      {%
```

First use plural form:

```
10804      \glscapscase
10805      {%
```

First use plural form, don't adjust case:

```
10806      \genplacrfullformat{\glslabel}{\glsinsert}%
10807      }%
10808      {%
```

First use plural form, make first letter upper case:

```
10809      \Genplacrfullformat{\glslabel}{\glsinsert}%
10810      }%
10811      {%
```

First use plural form, all caps:

```
10812      \mfirstucMakeUppercase
10813      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10814      }%
10815      }%
10816      {%
```

First use singular form

```
10817      \glscapscase
10818      {%
```

First use singular form, don't adjust case:

```
10819      \genacrfullformat{\glslabel}{\glsinsert}%
10820      }%
10821      {%
```



First use singular form, make first letter upper case:

```
10822      \Genacrfullformat{\glslabel}{\glsinsert}%
10823      }%
10824      {%
```

First use singular form, all caps:

```
10825      \mfirstucMakeUppercase
10826      {\genacrfullformat{\glslabel}{\glsinsert}}%
10827      }%
10828      }%
10829      }%
10830      }%
10831      {%
```

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.

```
10832      \glscustomtext
10833      }%
10834 }
```

**enacrfullformat** Redefine to include accessibility information.

```
10835 \renewcommand*{\genacrfullformat}[2]{%
10836   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10837   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
10838 }
```

**enacrfullformat** Redefine to include accessibility information.

```
10839 \renewcommand*{\Genacrfullformat}[2]{%
10840   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10841   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
10842 }
```

**placrfullformat** Redefine to include accessibility information.

```
10843 \renewcommand*{\genplacrfullformat}[2]{%
10844   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10845   (\glsshortpluralaccessdisplay
10846     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10847 }
```

**placrfullformat** Redefine to include accessibility information.

```
10848 \renewcommand*{\Genplacrfullformat}[2]{%
10849   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10850   (\glsshortpluralaccessdisplay
10851     {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10852 }
```

**\@acrshort**

```
10853 \def\@acrshort#1#2[#3]{%
10854   \glsdoifexists{#2}%
```

```

10855 {%
10856   \let\do@gl@s@link@checkfirsthyper\relax

10857   \let\gl@sifplural\@secondoftwo
10858   \let\gl@scapscase\@firstofthree
10859   \let\gl@sinsert\@empty
10860   \def\glscustomtext{%
10861     \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10862   }%

   Call \@gl@s@link
10863   \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
10864 }%

10865 \glspostlinkhook
10866 }

```

\@Acrshort

```

10867 \def\@Acrshort#1#2[#3]{%
10868   \gl@sdoifexists{#2}%
10869   {%
10870     \let\do@gl@s@link@checkfirsthyper\relax

10871     \let\gl@sifplural\@secondoftwo
10872     \let\gl@scapscase\@secondofthree
10873     \let\gl@sinsert\@empty
10874     \def\glscustomtext{%
10875       \acronymfont{\glsshortaccessdisplay{\gl@sentryshort{#2}}{#2}}#3%
10876     }%

     Call \@gl@s@link
10877     \@gl@s@link[#1]{#2}{\csname gl@s\glstype @entryfmt\endcsname}%
10878   }%

10879   \glspostlinkhook
10880 }

```

\@ACRshort

```

10881 \def\@ACRshort#1#2[#3]{%
10882   \gl@sdoifexists{#2}%
10883   {%
10884     \let\do@gl@s@link@checkfirsthyper\relax

10885     \let\gl@sifplural\@secondoftwo
10886     \let\gl@scapscase\@thirdofthree
10887     \let\gl@sinsert\@empty
10888     \def\glscustomtext{%
10889       \acronymfont{\glsshortaccessdisplay
10890         {\MakeUppercase{\gl@sentryshort{#2}}}{#2}}#3%
10891     }%

```

```

    Call \@gls@link
10892   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10893   }%

10894   \glspostlinkhook
10895 }

```

\@acrlong

```

10896 \def\@acrlong#1#2[#3]{%
10897   \glsdoifexists{#2}%
10898   {%
10899     \let\do@gls@link@checkfirsthyper\relax

10900     \let\glsifplural\@secondoftwo
10901     \let\glscapscase\@firstofthree
10902     \let\glsinsert\@empty
10903     \def\glscustomtext{%
10904       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10905     }%

```

```

    Call \@gls@link
10906   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10907   }%

10908   \glspostlinkhook
10909 }

```

\@Acrlong

```

10910 \def\@Acrlong#1#2[#3]{%
10911   \glsdoifexists{#2}%
10912   {%
10913     \let\do@gls@link@checkfirsthyper\relax

10914     \let\glsifplural\@secondoftwo
10915     \let\glscapscase\@firstofthree
10916     \let\glsinsert\@empty
10917     \def\glscustomtext{%
10918       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#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\glsifscapscase\@firstofthree
10930 \let\glsinsert\@empty
10931 \def\glscustomtext{%
10932     \acronymfont{\glslongaccessdisplay{%
10933         \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10934     }%

    Call \@gls@link
10935     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10936 }%

10937 \glspostlinkhook
10938 }

```

## 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.

```

10939 \renewcommand*{\glossentryname}[1]{%
10940     \glsdoifexists{#1}%
10941     {%
10942         \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10943     }%
10944 }

10945 \renewcommand*{\glossentrydesc}[1]{%
10946     \glsdoifexists{#1}%
10947     {%
10948         \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10949     }%
10950 }

10951 \renewcommand*{\glossentrydesc}[1]{%
10952     \glsdoifexists{#1}%
10953     {%
10954         \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10955     }%
10956 }

10957 \renewcommand*{\Glossentrydesc}[1]{%
10958     \glsdoifexists{#1}%
10959     {%
10960         \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10961     }%
10962 }

```

```

10963 \renewcommand*{\glossentrysymbol}[1]{%
10964   \glsdoifexists{#1}%
10965   {%
10966     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10967   }%
10968 }

10969 \renewcommand*{\Glossentrysymbol}[1]{%
10970   \glsdoifexists{#1}%
10971   {%
10972     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10973   }%
10974 }

```

ssaryentryfield

```

10975 \newcommand*{\accsuppglossaryentryfield}[5]{%
10976   \glossaryentryfield{#1}%
10977   {\glsnameaccessdisplay{#2}{#1}}%
10978   {\glsdescriptionaccessdisplay{#3}{#1}}%
10979   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10980 }

```

rysubentryfield

```

10981 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10982   \glossarysubentryfield{#1}{#2}%
10983   {\glsnameaccessdisplay{#3}{#2}}%
10984   {\glsdescriptionaccessdisplay{#4}{#2}}%
10985   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10986 }

```

## 5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short    *<long>* (*<short>*) acronym style.

```

10987 \renewacronymstyle{long-short}%
10988 {%

```

Check for long form in case this is a mixed glossary.

```

10989   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10990 }%
10991 {%
10992   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10993   \renewcommand*{\genacrfullformat}[2]{%
10994     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10995     (\glsshortaccessdisplay
10996       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10997   }%
10998   \renewcommand*{\Genacrfullformat}[2]{%

```

```

10999 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
11000 (\glsshortaccessdisplay
11001   {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
11002 }%
11003 \renewcommand*{\genplacrfullformat}[2]{%
11004   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
11005   (\glsshortpluralaccessdisplay
11006     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
11007 }%
11008 \renewcommand*{\Genplacrfullformat}[2]{%
11009   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
11010   (\glsshortpluralaccessdisplay
11011     {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1})%
11012 }%
11013 \renewcommand*{\acronymentry}[1]{%
11014   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}
11015 \renewcommand*{\acronymsort}[2]{##1}%
11016 \renewcommand*{\acronymfont}[1]{##1}%
11017 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11018 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11019 }

```

short-long (*short*) (*long*) acronym style.

```

11020 \renewacronymstyle{short-long}%
11021 {%
11022   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11023 }%
11024 {%
11025   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11026   \renewcommand*{\genacrfullformat}[2]{%
11027     \glsshortaccessdisplay
11028       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2\space
11029     (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
11030   }%
11031   \renewcommand*{\Genacrfullformat}[2]{%
11032     \glsshortaccessdisplay
11033       {\protect\firstacronymfont{\Glsentryshort{##1}}}{##1}##2\space
11034     (\glslongaccessdisplay{\glsentrylong{##1}}{##1})%
11035   }%
11036   \renewcommand*{\genplacrfullformat}[2]{%
11037     \glsshortpluralaccessdisplay
11038       {\protect\firstacronymfont{\glsentryshortpl{##1}}}{##1}##2\space
11039     (\glslongpluralaccessdisplay
11040       {\glsentrylongpl{##1}}{##1})%
11041   }%
11042   \renewcommand*{\Genplacrfullformat}[2]{%
11043     \glsshortpluralaccessdisplay
11044       {\protect\firstacronymfont{\Glsentryshortpl{##1}}}{##1}##2\space

```

```

11045 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11046 }%
11047 \renewcommand*{\acronymentry}[1]{%
11048   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11049 \renewcommand*{\acronymsort}[2]{##1}%
11050 \renewcommand*{\acronymfont}[1]{##1}%
11051 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11052 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11053 }

```

long-short-desc *long* (*short*) acronym style that has an accompanying description (which the user needs to supply).

```

11054 \renewacronymstyle{long-short-desc}%
11055 {%
11056   \GlsUseAcrEntryDispStyle{long-short}%
11057 }%
11058 {%
11059   \GlsUseAcrStyleDefs{long-short}%
11060   \renewcommand*{\GenericAcronymFields}{}%
11061   \renewcommand*{\acronymsort}[2]{##2}%
11062   \renewcommand*{\acronymentry}[1]{%
11063     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11064     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11065 }

```

g-sc-short-desc *long* (\textsc{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11066 \renewacronymstyle{long-sc-short-desc}%
11067 {%
11068   \GlsUseAcrEntryDispStyle{long-sc-short}%
11069 }%
11070 {%
11071   \GlsUseAcrStyleDefs{long-sc-short}%
11072   \renewcommand*{\GenericAcronymFields}{}%
11073   \renewcommand*{\acronymsort}[2]{##2}%
11074   \renewcommand*{\acronymentry}[1]{%
11075     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11076     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11077 }

```

g-sm-short-desc *long* (\textsmaller{short}) acronym style that has an accompanying description (which the user needs to supply).

```

11078 \renewacronymstyle{long-sm-short-desc}%
11079 {%
11080   \GlsUseAcrEntryDispStyle{long-sm-short}%
11081 }%
11082 {%
11083   \GlsUseAcrStyleDefs{long-sm-short}%
11084   \renewcommand*{\GenericAcronymFields}{}%

```

```

11085 \renewcommand*{\acronymsort}[2]{##2}%
11086 \renewcommand*{\acronymentry}[1]{%
11087   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11088   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11089 }

```

short-long-desc    *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

11090 \renewacronymstyle{short-long-desc}%
11091 {%
11092   \GlsUseAcrEntryDispStyle{short-long}%
11093 }%
11094 {%
11095   \GlsUseAcrStyleDefs{short-long}%
11096   \renewcommand*{\GenericAcronymFields}{}%
11097   \renewcommand*{\acronymsort}[2]{##2}%
11098   \renewcommand*{\acronymentry}[1]{%
11099     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11100     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11101 }

```

short-long-desc    *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11102 \renewacronymstyle{sc-short-long-desc}%
11103 {%
11104   \GlsUseAcrEntryDispStyle{sc-short-long}%
11105 }%
11106 {%
11107   \GlsUseAcrStyleDefs{sc-short-long}%
11108   \renewcommand*{\GenericAcronymFields}{}%
11109   \renewcommand*{\acronymsort}[2]{##2}%
11110   \renewcommand*{\acronymentry}[1]{%
11111     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11112     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11113 }

```

short-long-desc    *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11114 \renewacronymstyle{sm-short-long-desc}%
11115 {%
11116   \GlsUseAcrEntryDispStyle{sm-short-long}%
11117 }%
11118 {%
11119   \GlsUseAcrStyleDefs{sm-short-long}%
11120   \renewcommand*{\GenericAcronymFields}{}%
11121   \renewcommand*{\acronymsort}[2]{##2}%
11122   \renewcommand*{\acronymentry}[1]{%
11123     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11124     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```



11125 }

dua *<long>* only acronym style.

11126 \renewacronymstyle{dua}%

11127 {%

Check for long form in case this is a mixed glossary.

11128 \ifdefempty\glscustomtext

11129 {%

11130 \ifglshaslong{\glslabel}%

11131 {%

11132 \glsifplural

11133 {%

Plural form:

11134 \glscapscase

11135 {%

Plural form, don't adjust case:

11136 \gslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%

11137 \glsinsert

11138 }%

11139 {%

Plural form, make first letter upper case:

11140 \gslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%

11141 \glsinsert

11142 }%

11143 {%

Plural form, all caps:

11144 \gslongpluralaccessdisplay

11145 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%

11146 \mfirstucMakeUppercase{\glsinsert}%

11147 }%

11148 }%

11149 {%

Singular form

11150 \glscapscase

11151 {%

Singular form, don't adjust case:

11152 \gslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert

11153 }%

11154 {%

Subsequent singular form, make first letter upper case:

11155 \gslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert

11156 }%

11157 {%

Subsequent singular form, all caps:

```

11158      \glslongaccessdisplay
11159      {\mfirstucMakeUppercase
11160       {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11161      \mfirstucMakeUppercase{\glsinsert}%
11162      }%
11163      }%
11164      }%
11165      {%

```

Not an acronym:

```

11166      \glsgenentryfmt
11167      }%
11168      }%
11169      {\glscustomtext\glsinsert}%
11170      }%
11171      {%
11172      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11173      \renewcommand*{\acrfullfmt}[3]{%
11174        \glslink[##1]{##2}{%
11175          \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11176          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11177      \renewcommand*{\Acrfullfmt}[3]{%
11178        \glslink[##1]{##2}{%
11179          \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11180          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11181      \renewcommand*{\ACRfullfmt}[3]{%
11182        \glslink[##1]{##2}{%
11183          \glslongaccessdisplay
11184          {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11185          (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11186      \renewcommand*{\acrfullplfmt}[3]{%
11187        \glslink[##1]{##2}{%
11188          \glslongpluralaccessdisplay
11189          {\glsentrylongpl{##2}}{##2}##3\space
11190          (\glsshortpluralaccessdisplay
11191          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11192      \renewcommand*{\ACRfullplfmt}[3]{%
11193        \glslink[##1]{##2}{%
11194          \glslongpluralaccessdisplay
11195          {\Glsentrylongpl{##2}}{##2}##3\space
11196          (\glsshortpluralaccessdisplay
11197          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11198      \renewcommand*{\ACRfullplplfmt}[3]{%
11199        \glslink[##1]{##2}{%
11200          \glslongpluralaccessdisplay
11201          {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11202          (\glsshortpluralaccessdisplay
11203          {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11204      \renewcommand*{\glsentryfull}[1]{%

```

```

11205 \glslongaccessdisplay{\glsentrylong{##1}}\space
11206 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11207 }%
11208 \renewcommand*{\Glsentryfull}[1]{%
11209 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11210 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11211 }%
11212 \renewcommand*{\glsentryfullpl}[1]{%
11213 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11214 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11215 }%
11216 \renewcommand*{\Glsentryfullpl}[1]{%
11217 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11218 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11219 }%
11220 \renewcommand*{\acronymentry}[1]{%
11221 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11222 \renewcommand*{\acronymsort}[2]{##1}%
11223 \renewcommand*{\acronymfont}[1]{##1}%
11224 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11225 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11226 \renewacronymstyle{dua-desc}%
11227 {%
11228 \GlsUseAcrEntryDispStyle{dua}%
11229 }%
11230 {%
11231 \GlsUseAcrStyleDefs{dua}%
11232 \renewcommand*{\GenericAcronymFields}{}%
11233 \renewcommand*{\acronymentry}[1]{%
11234 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11235 \renewcommand*{\acronymsort}[2]{##2}%
11236 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11237 \renewacronymstyle{footnote}%
11238 {%
11239 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11240 }%
11241 {%
11242 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11243 \glshyperfirstfalse
11244 \renewcommand*{\genacrfullformat}[2]{%
11245 \glsshortaccessdisplay
11246 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

11247 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11248 }%
11249 \renewcommand*{\Genacrfullformat}[2]{%
11250 \glsshortaccessdisplay
11251   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11252 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11253 }%
11254 \renewcommand*{\genplacrfullformat}[2]{%
11255 \glsshortpluralaccessdisplay
11256   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11257 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11258 }%
11259 \renewcommand*{\Genplacrfullformat}[2]{%
11260 \glsshortpluralaccessdisplay
11261   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11262 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11263 }%
11264 \renewcommand*{\acronymentry}[1]{%
11265 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11266 \renewcommand*{\acronymsort}[2]{##1}%
11267 \renewcommand*{\acronymfont}[1]{##1}%
11268 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11269 \renewcommand*{\acrfullfmt}[3]{%
11270 \glslink{##1}{##2}{%
11271 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11272 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11273 \renewcommand*{\Acrfullfmt}[3]{%
11274 \glslink{##1}{##2}{%
11275 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11276 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11277 \renewcommand*{\ACRfullfmt}[3]{%
11278 \glslink{##1}{##2}{%
11279 \glsshortaccessdisplay
11280   {\mfirstucMakeUppercase
11281   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11282   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11283 \renewcommand*{\acrfullplfmt}[3]{%
11284 \glslink{##1}{##2}{%
11285 \glsshortpluralaccessdisplay
11286   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11287   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11288 \renewcommand*{\Acrfullplfmt}[3]{%
11289 \glslink{##1}{##2}{%
11290 \glsshortpluralaccessdisplay
11291   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11292   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11293 \renewcommand*{\ACRfullplfmt}[3]{%
11294 \glslink{##1}{##2}{%

```

```

11295 \glsshortpluralaccessdisplay
11296 {\mfirstucMakeUppercase
11297 {\acronymfont{\glentryshortpl{##2}}{##2}##3\space
11298 (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

11299 \renewcommand*{\glentryfull}[1]{%
11300 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}\space
11301 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11302 \renewcommand*{\Glsentryfull}[1]{%
11303 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}{##1}\space
11304 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11305 \renewcommand*{\glentryfullpl}[1]{%
11306 \glsshortpluralaccessdisplay
11307 {\acronymfont{\glentryshortpl{##1}}{##1}\space
11308 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11309 \renewcommand*{\Glsentryfullpl}[1]{%
11310 \glsshortpluralaccessdisplay
11311 {\acronymfont{\Glsentryshortpl{##1}}{##1}\space
11312 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11313 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11314 \renewacronymstyle{footnote-sc}%
11315 {%
11316 \GlsUseAcrEntryDispStyle{footnote}%
11317 }%
11318 {%
11319 \GlsUseAcrStyleDefs{footnote}%
11320 \renewcommand{\acronymentry}[1]{%
11321 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11322 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11323 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11324 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11325 \renewacronymstyle{footnote-sm}%
11326 {%
11327 \GlsUseAcrEntryDispStyle{footnote}%
11328 }%
11329 {%
11330 \GlsUseAcrStyleDefs{footnote}%
11331 \renewcommand{\acronymentry}[1]{%
11332 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}{##1}}
11333 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11334 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11335 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11336 \renewacronymstyle{footnote-desc}%
11337 {%
11338   \GlsUseAcrEntryDisplayStyle{footnote}%
11339 }%
11340 {%
11341   \GlsUseAcrStyleDefs{footnote}%
11342   \renewcommand*{\GenericAcronymFields}{}%
11343   \renewcommand*{\acronymsort}[2]{##2}%
11344   \renewcommand*{\acronymentry}[1]{%
11345     \glslongaccessdisplay{\glstrylong{##1}}{##1}\space
11346     (\glsshortaccessdisplay{\acronymfont{\glstryshort{##1}}}{##1})}%
11347 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11348 \renewacronymstyle{footnote-sc-desc}%
11349 {%
11350   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11351 }%
11352 {%
11353   \GlsUseAcrStyleDefs{footnote-sc}%
11354   \renewcommand*{\GenericAcronymFields}{}%
11355   \renewcommand*{\acronymsort}[2]{##2}%
11356   \renewcommand*{\acronymentry}[1]{%
11357     \glslongaccessdisplay{\glstrylong{##1}}{##1}\space
11358     (\glsshortaccessdisplay{\acronymfont{\glstryshort{##1}}}{##1})}%
11359 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11360 \renewacronymstyle{footnote-sm-desc}%
11361 {%
11362   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11363 }%
11364 {%
11365   \GlsUseAcrStyleDefs{footnote-sm}%
11366   \renewcommand*{\GenericAcronymFields}{}%
11367   \renewcommand*{\acronymsort}[2]{##2}%
11368   \renewcommand*{\acronymentry}[1]{%
11369     \glslongaccessdisplay{\glstrylong{##1}}{##1}\space
11370     (\glsshortaccessdisplay{\acronymfont{\glstryshort{##1}}}{##1})}%
11371 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11372 \renewcommand*{\newacronymhook}{%
11373   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11374     \the\glskeylisttok}%
11375   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11376 }

1tNewAcronymDef Modify default style to use access text:

```
11377 \renewcommand*{\DefaultNewAcronymDef}{%
11378   \edef\@do@newglossaryentry{%
11379     \noexpand\newglossaryentry{\the\glslabeltok}%
11380     {%
11381       type=\acronymtype,%
11382       name={\the\glsshorttok},%
11383       description={\the\glslongtok},%
11384       descriptionaccess=\relax,%
11385       text={\the\glsshorttok},%
11386       access={\noexpand\@glo@textaccess},%
11387       sort={\the\glsshorttok},%
11388       short={\the\glsshorttok},%
11389       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11390       shortaccess={\the\glslongtok},%
11391       long={\the\glslongtok},%
11392       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11393       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11394       first={\noexpand\glslongaccessdisplay
11395         {\the\glslongtok}{\the\glslabeltok}\space
11396         (\noexpand\glsshortaccessdisplay
11397           {\the\glsshorttok}{\the\glslabeltok})},%
11398       plural={\the\glsshorttok\acrpluralsuffix},%
11399       firstplural={\noexpand\glslongpluralaccessdisplay
11400         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11401         (\noexpand\glsshortpluralaccessdisplay
11402           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11403       firstaccess=\relax,%
11404       firstpluralaccess=\relax,%
11405       textaccess={\noexpand\@glo@shortaccess},%
11406       \the\glskeylisttok
11407     }%
11408   }%
11409   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11410   \let\@org@gls@assign@plural\gls@assign@plural
11411   \let\@org@gls@assign@descplural\gls@assign@descplural
11412   \def\gls@assign@firstpl##1##2{%
11413     \@gls@expand@field{##1}{firstpl}{##2}%
11414   }%
11415   \def\gls@assign@plural##1##2{%
11416     \@gls@expand@field{##1}{plural}{##2}%
11417   }%
11418   \def\gls@assign@descplural##1##2{%
11419     \@gls@expand@field{##1}{descplural}{##2}%
11420   }%
11421   \@do@newglossaryentry
11422   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11423 \let\gls@assign@plural\@org@gls@assign@plural
11424 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11425 }

```

teNewAcronymDef

```

11426 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11427   \edef\@do@newglossaryentry{%
11428     \noexpand\newglossaryentry{\the\glslabeltok}%
11429     {%
11430       type=\acronymtype,%
11431       name={\noexpand\acronymfont{\the\glsshorttok}},%
11432       sort={\the\glsshorttok},%
11433       text={\the\glsshorttok},%
11434       short={\the\glsshorttok},%
11435       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11436       shortaccess={\the\glslongtok},%
11437       long={\the\glslongtok},%
11438       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11439       access={\noexpand\@glo@textaccess},%
11440       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11441       symbol={\the\glslongtok},%
11442       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11443       firstpluralaccess=\relax,
11444       textaccess={\noexpand\@glo@shortaccess},%
11445       \the\glskeylisttok
11446     }%
11447   }%
11448   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11449   \let\@org@gls@assign@plural\gls@assign@plural
11450   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11451   \def\gls@assign@firstpl##1##2{%
11452     \@@gls@expand@field{##1}{firstpl}{##2}%
11453   }%
11454   \def\gls@assign@plural##1##2{%
11455     \@@gls@expand@field{##1}{plural}{##2}%
11456   }%
11457   \def\gls@assign@symbolplural##1##2{%
11458     \@@gls@expand@field{##1}{symbolplural}{##2}%
11459   }%
11460   \@do@newglossaryentry
11461   \let\gls@assign@plural\@org@gls@assign@plural
11462   \let\gls@assign@firstpl\@org@gls@assign@firstpl
11463   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11464 }

```

onNewAcronymDef

```

11465 \renewcommand*{\DescriptionNewAcronymDef}{%
11466   \edef\@do@newglossaryentry{%
11467     \noexpand\newglossaryentry{\the\glslabeltok}%

```



```

11468 {%
11469     type=\acronymtype,%
11470     name={\noexpand
11471         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11472     access={\noexpand\@glo@textaccess},%
11473     sort={\the\glsshorttok},%
11474     short={\the\glsshorttok},%
11475     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11476     shortaccess={\the\glslongtok},%
11477     long={\the\glslongtok},%
11478     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11479     first={\the\glslongtok},%
11480     firstaccess=\relax,
11481     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11482     text={\the\glsshorttok},%
11483     textaccess={\the\glslongtok},%
11484     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11485     symbol={\noexpand\@glo@text},%
11486     symbolaccess={\noexpand\@glo@textaccess},%
11487     symbolplural={\noexpand\@glo@plural},%
11488     firstpluralaccess=\relax,
11489     textaccess={\noexpand\@glo@shortaccess},%
11490     \the\glskeylisttok}%
11491 }%
11492 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11493 \let\@org@gls@assign@plural\gls@assign@plural
11494 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11495 \def\gls@assign@firstpl##1##2{%
11496     \@gls@expand@field{##1}{firstpl}{##2}%
11497 }%
11498 \def\gls@assign@plural##1##2{%
11499     \@gls@expand@field{##1}{plural}{##2}%
11500 }%
11501 \def\gls@assign@symbolplural##1##2{%
11502     \@gls@expand@field{##1}{symbolplural}{##2}%
11503 }%
11504 \@do@newglossaryentry
11505 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11506 \let\gls@assign@plural\@org@gls@assign@plural
11507 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11508 }

```

teNewAcronymDef

```

11509 \renewcommand*{\FootnoteNewAcronymDef}{%
11510     \edef\@do@newglossaryentry{%
11511         \noexpand\newglossaryentry{\the\glslabeltok}%
11512         {%
11513             type=\acronymtype,%
11514             name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11515     sort={\the\glsshorttok},%
11516     text={\the\glsshorttok},%
11517     textaccess={\the\glslongtok},%
11518     access={\noexpand\@glo@textaccess},%
11519     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11520     short={\the\glsshorttok},%
11521     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11522     long={\the\glslongtok},%
11523     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11524     description={\the\glslongtok},%
11525     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11526     \the\glskeylisttok
11527   }%
11528 }%
11529 \let\@org@gls@assign@plural\gls@assign@plural
11530 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11531 \let\@org@gls@assign@descplural\gls@assign@descplural
11532 \def\gls@assign@firstpl##1##2{%
11533   \@@gls@expand@field{##1}{firstpl}{##2}%
11534 }%
11535 \def\gls@assign@plural##1##2{%
11536   \@@gls@expand@field{##1}{plural}{##2}%
11537 }%
11538 \def\gls@assign@descplural##1##2{%
11539   \@@gls@expand@field{##1}{descplural}{##2}%
11540 }%
11541 \do@newglossaryentry
11542 \let\gls@assign@plural\@org@gls@assign@plural
11543 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11544 \let\gls@assign@descplural\@org@gls@assign@descplural
11545 }

```

# 11NewAcronymDef

```

11546 \renewcommand*{\SmallNewAcronymDef}{%
11547   \edef\@do@newglossaryentry{%
11548     \noexpand\newglossaryentry{\the\glslabeltok}%
11549     {%
11550       type=\acronymtype,%
11551       name={\noexpand\acronymfont{\the\glsshorttok}},%
11552       access={\noexpand\@glo@symbolaccess},%
11553       sort={\the\glsshorttok},%
11554       short={\the\glsshorttok},%
11555       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11556       shortaccess={\the\glslongtok},%
11557       long={\the\glslongtok},%
11558       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11559       text={\noexpand\@glo@short},%
11560       textaccess={\noexpand\@glo@shortaccess},%
11561       plural={\noexpand\@glo@shortpl},%

```

```

11562     first={\the\glslongtok},%
11563     firstaccess=\relax,
11564     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11565     description={\noexpand\@glo@first},%
11566     descriptionplural={\noexpand\@glo@firstplural},%
11567     symbol={\the\glsshorttok},%
11568     symbolaccess={\the\glslongtok},%
11569     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11570     \the\glskeylisttok
11571   }%
11572 }%
11573 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11574 \let\@org@gls@assign@plural\gls@assign@plural
11575 \let\@org@gls@assign@descplural\gls@assign@descplural
11576 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11577 \def\gls@assign@firstpl##1##2{%
11578   \@@gls@expand@field{##1}{firstpl}{##2}%
11579 }%
11580 \def\gls@assign@plural##1##2{%
11581   \@@gls@expand@field{##1}{plural}{##2}%
11582 }%
11583 \def\gls@assign@descplural##1##2{%
11584   \@@gls@expand@field{##1}{descplural}{##2}%
11585 }%
11586 \def\gls@assign@symbolplural##1##2{%
11587   \@@gls@expand@field{##1}{symbolplural}{##2}%
11588 }%
11589 \do@newglossaryentry
11590 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11591 \let\gls@assign@plural\@org@gls@assign@plural
11592 \let\gls@assign@descplural\@org@gls@assign@descplural
11593 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11594 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
11595 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
11596 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
11597 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
11598 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

## 5.5 Debugging Commands

owglonameaccess

```
11599 \newcommand*{\showglonameaccess}[1]{%  
11600   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname  
11601 }
```

owglotextaccess

```
11602 \newcommand*{\showglotextaccess}[1]{%  
11603   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname  
11604 }
```

glopluralaccess

```
11605 \newcommand*{\showglopluralaccess}[1]{%  
11606   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname  
11607 }
```

wglofirstaccess

```
11608 \newcommand*{\showglofirstaccess}[1]{%  
11609   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname  
11610 }
```

rstpluralaccess

```
11611 \newcommand*{\showglofirstpluralaccess}[1]{%  
11612   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname  
11613 }
```

glosymbolaccess

```
11614 \newcommand*{\showglosymbolaccess}[1]{%  
11615   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname  
11616 }
```

bolpluralaccess

```
11617 \newcommand*{\showglosymbolpluralaccess}[1]{%  
11618   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname  
11619 }
```

owglodescaccess

```
11620 \newcommand*{\showglodescaccess}[1]{%  
11621   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname  
11622 }
```

escpluralaccess

```
11623 \newcommand*{\showglodescpluralaccess}[1]{%  
11624   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname  
11625 }
```

wgloshortaccess

```
11626 \newcommand*{\showgloshortaccess}[1]{%  
11627   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11628 }
```

ortpluralaccess

```
11629 \newcommand*{\showgloshortpluralaccess}[1]{%  
11630   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11631 }
```

owglolongaccess

```
11632 \newcommand*{\showglolongaccess}[1]{%  
11633   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11634 }
```

ongpluralaccess

```
11635 \newcommand*{\showglolongpluralaccess}[1]{%  
11636   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11637 }
```

## 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.

```
11638 \NeedsTeXFormat{LaTeX2e}
11639 \ProvidesPackage{glossaries-babel}[2017/06/11 v4.30 (NLCT)]
```

Load tracklang to obtain language settings.

```
11640 \RequirePackage{tracklang}
11641 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11642 \AnyTrackedLanguages
11643 {%
11644   \ForEachTrackedDialect{\this@dialect}{%
11645     \IfTrackedLanguageFileExists{\this@dialect}%
11646       {glossaries-}% prefix
11647       {.ldf}%
11648       {%
11649         \RequireGlossariesLang{\CurrentTrackedTag}%
11650       }%
11651       {%
11652         \PackageWarningNoLine{glossaries}%
11653           {No language module detected for ‘\this@dialect’.\MessageBreak
11654             Language modules need to be installed separately.\MessageBreak
11655             Please check on CTAN for a bundle called\MessageBreak
11656             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11657       }%
11658     }%
11659   }%
11660 }
```

### 6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11661 \NeedsTeXFormat{LaTeX2e}
11662 \ProvidesPackage{glossaries-polyglossia}[2017/06/11 v4.30 (NLCT)]
```

Load tracklang to obtain language settings.

```
11663 \RequirePackage{tracklang}
11664 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11665 \AnyTrackedLanguages
```

```

11666 {%
11667     \ForEachTrackedDialect{\this@dialect}{%
11668         \IfTrackedLanguageFileExists{\this@dialect}%
11669         {glossaries-}% prefix
11670         {.ldf}%
11671         {%
11672             \RequireGlossariesLang{\CurrentTrackedTag}%
11673         }%
11674         {%
11675             \PackageWarningNoLine{glossaries}%
11676             {No language module detected for ‘\this@dialect’.\MessageBreak
11677             Language modules need to be installed separately.\MessageBreak
11678             Please check on CTAN for a bundle called\MessageBreak
11679             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11680         }%
11681     }%
11682 }%
11683 {}%

```

# Glossary

`makeindex` An indexing application. [10](#), [26](#), [27](#), [175](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [10](#), [26](#), [27](#), [175](#)



# Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 6
General: Added range facility in format key . . . . . 111	
\writeist: Added spaces after \delimN and \delimR in ist file . . . . . 157	
1.04 (2007-08-03)	1.12 (2008-03-08)
General: Added \glstextformat . . . . . 95	\@GLSpl: now uses
1.05 (2007-08-10)	\glentrydescplural and
\glossarysection: added \@mkboth to \glossarysection . . . . . 38	\glentrysymbolplural instead of
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key . . . . . 79	\glentrydesc and
1.07 (2007-09-13)	\glentrysymbol . . . . . 125
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon . . . . . 109	\@Glspl@: now uses
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon . . . . . 155	\glentrydescplural and
1.08 (2007-10-13)	\glentrysymbolplural instead of
General: Added babel support . . . . . 32	\glentrydesc and
listgroup: changed listgroup style to use \glsgetgrouptitle . . . . . 269	\glentrysymbol . . . . . 123
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle . . . . . 270	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) . . . . . 119
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) . . . . . 79
\@gls@tmpb: changed \toksdef to \newtoks . . . . . 113	descriptionplural support added . . . . . 79
\@gls@toc: numberline added . . . . . 41	symbolplural support added . . . . . 79
\@p@glossarysection: numbered sections and auto label added . . . . . 40	\Glsentrydescplural: New . . . . . 148
General: amsgen now loaded (\new@ifnextchar needed) . . . . . 4	\glentrydescplural: New . . . . . 148
translate: translate option added . . . . . 24	\Glsentrysymbolplural: New . . . . . 149
\setglossarysection: new . . . . . 39	\glentrysymbolplural: New . . . . . 149
numberedsection: numberedsection package option added . . . . . 7	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink . . . . . 235
	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink . . . . . 241
	symbolplural: new . . . . . 63

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter .....	126–133
\ACRfullpl: new .....	216
\Acrfullpl: new .....	216
\acrfullpl: new .....	215
\acrpluralsuffix: New .....	213
\gls@defglossaryentry: Changed default first value .....	79
Changed default firstplural value .....	79
Removed restriction on only using \newglossaryentry in the preamble	84
\newacronym: Removed restriction on only using \newacronym in the preamble .....	213
1.14 (2008-06-17)	
\@gls@hypergroup: new .....	264
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 .....	263
\glsnavhypertarget: added write to aux file .....	263
\glsnavigation: changed to only use labels for groups that are present ..	264
1.15 (2008-08-15)	
\@gls@link: added \glslabel .....	109
\gls@defglossaryentry: check for \@glo@first in description .....	83
check for \@glo@text in symbol .....	83
\gls@hypergroup: new .....	264
\glsnavhypertarget: added check if rerun required .....	263
\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 .....	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 .....	122
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	124
\@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 .....	125
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used .....	123
\@gls@target: raised the hypertarget so the target text doesn't scroll off the top of the page .....	119
\gls@defglossaryentry: Changed def to let .....	79
1.17 (2008-12-26)	
\@do@wrglossary: new .....	179
\@do@seeglossary: new .....	181
\@glo@storeentry: new .....	85
\@gls@glossary: changed definition to use \index instead of \@index ....	176
\@glsdefaultplural: new .....	66
\@glsdefaultsort: new .....	67
\@gls@hypernumber: new .....	210
\@gls@noname: new .....	66
\@gls@nonextpages: new .....	200
General: added xindy support .....	26
parent: new .....	64
see: new .....	63
\gls@defglossaryentry: added nonumberlist key .....	79
added parent key .....	79
added see key .....	79
Stored main part of entry format when entry is defined .....	84
\gls@suffixF: new .....	36
\gls@suffixFF: new .....	37
\gls@wrglossary: modified to allow for xindy support .....	176

\glshyperlink: new .....	154	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	235
\glshypernumber: modified to allow material to be attached to location	210	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller .....	241
\glshnavhyperlink: replaced \hyperlink to \@glslink .....	263	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller .....	244
\glshnavhypertarget: replaced \hypertarget to \@glstarget ...	263	2.01 (2009 May 30)	
\glsssee: new .....	182	\@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit .....	110
\glssseeformat: new .....	182	\forall glossaries: replaced \ifthenelse with \ifx .....	50
\glssSetSuffixF: new .....	37	\forall glossentries: replaced \ifthenelse with \ifx .....	51
\glssSetSuffixFF: new .....	37	\glssdefmain: new .....	13
\ifglsxindy: new .....	26	\glssdescwidth: changed \linewidth to \hsize .....	271, 293
\istfilename: added xindy support ...	35	\glsslistdottedwidth: changed \linewidth to \hsize .....	271
\newglossarystyle: made \newglossarystyle long .....	209	\glsspagelistwidth: changed \linewidth to \hsize .....	271, 293
\nopostdesc: new .....	35	nomain: added nomain package option	14
nonumberlist: new .....	64	\writeist: removed item_02 - no such makeindex key .....	162
\printglossary: added check to determine if \printglossary is already defined .....	184	2.02 (2007-07-13)	
added print language to aux file .....	184	\@printglossary: suppressed warning globally rather than locally .....	186
order: order package option added	26	2.02 (2009-07-13)	
\writeist: added xindy support	157	\glossarysection: changed \@mkboth to \glossarymark .....	38
1.18 (2009-01-14)		\glsglossarymark: New .....	39
\@glsl@loadlist: new .....	9	2.03 (2009-09-23)	
\@glsl@loadlong: new .....	8	\@GLS@: Added check for hyperfirst	122
\@glsl@loadsuper: new .....	9	\@GLSpl: Added check for hyperfirst	125
\@glsl@loadtree: new .....	9	\@Gls@: Added check for hyperfirst	122
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort .....	79	\@Glspl@: Added check for hyperfirst	124
moved sort sanitization to \newglossaryentry .....	83	\@glsl@: Added check for hyperfirst	121
\glstarget: new .....	203	\@glsl@link: new .....	108
\oldacronym: new .....	213	\@glsl@link: added \leavevmode	109
nolist: new .....	9	Moved entry existence check to avoid duplicate code .....	109
nolong: new .....	8	\@glsl@disp: Added check for hyperfirst	125
sort: moved sanitization to \newglossaryentry .....	62	\@glspl@: Added check for hyperfirst	123
nostyles: new .....	9	\glsglossarymark: Added check to see if it's already defined .....	39
nosuper: new .....	9	hyperfirst: new .....	25
notree: new .....	9		
1.19 (2009-03-02)			
\glsclearpage: new .....	41		
\glsl@disp: new .....	125		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	239		

2.04 (2009-11-10)	
\@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
\@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 .....	124
\@glossaryentryfield: new .....	85
\@glossarysubentryfield: new .....	85
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms .....	121
\@glsacronymlists: new .....	15
\@glsdisp: Changed test to check if glossary type has been identified as a list of acronyms .....	125
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms .....	123
\@newglossaryentryposthook: new ..	84
\@newglossaryentryprehook: new ...	84
acronymlists: new .....	17
\DeclareAcronymList: new .....	16
\DefineAcronymSynonyms: new .....	230
\gls@defglossaryentry: added user1-6 keys .....	79
\glsadd: fixed bug that ignored counter	155
\Glsentryuseri: new .....	150
\glsentryuseri: new .....	150
\Glsentryuserii: new .....	151
\glsentryuserii: new .....	151
\Glsentryuseriii: new .....	151
\glsentryuseriii: new .....	151
\Glsentryuseriv: new .....	151
\glsentryuseriv: new .....	151
\Glsentryuserv: new .....	151
\glsentryuserv: new .....	151
\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 .....	16
\SetDefaultAcronymDisplayStyle: new .....	232
\SetDefaultAcronymStyle: new ....	233
\SetDescriptionAcronymDisplayStyle: new .....	237
\SetDescriptionDUAAcronymDisplayStyle: new .....	236
\SetDescriptionFootnoteAcronymDisplayStyle: new .....	233
\SetDUADisplayStyle: new .....	245
\SetFootnoteAcronymDisplayStyle: new .....	240
\SetSmallAcronymDisplayStyle: new	242
2.05 (2010-02-06)	
\@glsdisp: Added closing brace. Patch provided by Sergiu Dotenco .....	125
Removed spurious brace. Patch provided by Sergiu Dotenco .....	126
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco .....	162
2.06 (2010-06-14)	
\altnewglossary: new .....	59
\CustomAcronymFields: new .....	247
\CustomNewAcronymDef: new .....	247
\SetCustomDisplayStyle: new .....	247
\SetCustomStyle: new .....	248
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 .....	176
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument .....	111
3.0 (2011-04-02)	
\@do@wrglossary: added check for hyper location prefix .....	179
modified to use new format .....	179
\@@glossarysec: replaced \@ifundefined with \ifcsundef ...	6
\@do@seeglossary: Sanitize and escape cross-referencing information ....	181
\@gls@counterwithin: new .....	10

\@gls@ifinlist: new .....	42	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter .....	155
\@gls@saveentrycounter .....	110	\GlsAddXdyCounters: new .....	42
added \@gls@setsort .....	110	\glentrycounterlabel: new .....	202
\@gls@saveentrycounter: new .....	110	\glentryitem: new .....	203
\@gls@setupsort@def: new .....	11	\Glentrylong: new .....	152
\@gls@setupsort@standard: new ....	11	\glentrylong: new .....	152
\@gls@setupsort@use: new .....	12	\Glentrylongpl: new .....	152
\@gls@xdy@locationlist: new .....	45	\glentrylongpl: new .....	152
\@glslink: replaced \@ifundefined		\Glentryshort: new .....	152
with \ifcsundef .....	119	\glentryshort: new .....	152
\@glsnextpages: new .....	200	\Glentryshortpl: new .....	152
\@print@glossary: replaced		\glentryshortpl: new .....	152
\@ifundefined with \ifcsundef ..	187	\glsgetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	207
\currentglossary .....	185	\gls glossarymark: replaced	
added \glsnextpages .....	186	\@ifundefined with \ifcsundef ..	39
make toctitle default to title .....	185	\glshyperlink: changed default from	
\@xdyattributelist: new .....	42	\glentryname to \glentrytext ..	154
General: added prefix to hyperlink ....	211	\glshypernumber: replaced	
etoolbox now loaded .....	4	\@ifundefined with \ifcsundef ..	210
replaced \@ifundefined with		\glsnumberformat: replaced	
\ifcsundef .....	30, 33, 106, 198	\@ifundefined with \ifcsundef ..	37
\acrfootnote: new .....	233	\glsrefentry: new .....	202
\ACRfull: added starred version .....	215	\glsresetsubentrycounter: new ...	201
\Acrfull: added starred version .....	215	\glsseeitem: hyperlink uses	
\acrfull: added starred version .....	214	\glsseeitemformat instead of	
\ACRfullpl: added starred version ...	216	\glentryname .....	183
\Acrfullpl: added starred version ...	216	\glsseeitemformat: new .....	183
\acrfullpl: added starred version ...	215	\glssortnumberfmt: new .....	11
\acrlinkfootnote: new .....	233	\glsstepentry: new .....	202
\acrno linkfootnote: new .....	233	\glsstepsubentry: new .....	202
savewrites: new .....	28	\glssubentrycounterlabel: new ...	202
see: added \@glo@seeautonumberlist ..	63	\glssubentryitem: new .....	203
seeautonumberlist: new .....	8	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef .....	203
\@ifundefined with \ifcsundef ..	38	short: new .....	65
\glossarystyle: replaced		shortplural: new .....	66
\@ifundefined with \ifcsundef ..	208	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	51
\@ifundefined with \ifcsundef ..	27	\ifglentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	52
\@gls@defsort .....	83	\istfile: deprecated .....	175
added short and long keys .....	80	glossaryentry: new .....	201
replaced \@ifundefined with		glossarysubentry: new .....	201
\ifcsundef .....	80	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	40	\newrobustcmd .....	69

<code>\newglossarystyle</code> : replaced		<code>\showglouseriii</code> : new	251
<code>\@ifundefined</code> with <code>\ifcsundef</code>	209	<code>\showglouseriv</code> : new	251
<code>\ns@newglossary</code> : added		<code>\showglouserv</code> : new	251
<code>\@gls@defsortcount</code>	59	<code>\showglouservi</code> : new	251
replaced <code>\@ifundefined</code> with		<code>subentrycounter</code> : new	10
<code>\ifcsundef</code>	59	<code>\writeist</code> : added xindy-only macro	
<code>entrycounter</code> : new	10	definitions to glossary open tag	159
<code>entrycounterwithin</code> : new	10	modified to support new format	157
<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	28	General: made robust	122
<code>long</code> : new	66	<code>\ACRfull</code> : made robust	215
<code>longplural</code> : new	66	<code>\Acrfull</code> : made robust	215
<code>nonumberlist</code> : now boolean	64	<code>\acrfull</code> : made robust	214
<code>sort</code> : new	10	<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.	214
<code>\printglossary</code> : replaced		<code>\ACRfullpl</code> : made robust	216
<code>\@ifundefined</code> with <code>\ifcsundef</code>	184	<code>\Acrfullpl</code> : made robust	216
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>		<code>\acrfullpl</code> : made robust	215
expanded options link options	233	<code>\ACRlong</code> : made robust	143
<code>\setentrycounter</code> : added optional		<code>\Acrlong</code> : made robust	142
argument	208	<code>\acrlong</code> : made robust	142
<code>\showacronymlists</code> : new	253	<code>\ACRlongpl</code> : made robust	145
<code>\showglocounter</code> : new	250	<code>\Acrlongpl</code> : made robust	144
<code>\showgloDESC</code> : new	251	<code>\acrlongpl</code> : made robust	144
<code>\showgloDESCplural</code> : new	252	<code>\ACRshort</code> : made robust	139
<code>\showglofirst</code> : new	250	<code>\Acrshort</code> : made robust	139
<code>\showglofirstpl</code> : new	250	<code>\acrshort</code> : made robust	138
<code>\showgloflag</code> : new	253	<code>\ACRshortpl</code> : made robust	141
<code>\showgloindex</code> : new	253	<code>\Acrshortpl</code> : made robust	141
<code>\showglolevel</code> : new	249	<code>\acrshortpl</code> : made robust	140
<code>\showglongame</code> : new	251	<code>\Gls</code> : made robust	121
<code>\showgloparent</code> : new	249	<code>\glsadd</code> : made robust	155
<code>\showgloplural</code> : new	249	<code>\glsaddall</code> : made robust	155
<code>\showglosort</code> : new	252	<code>\GLSdesc</code> : made robust	131
<code>\showglossaries</code> : new	253	<code>\Glsdesc</code> : made robust	130
<code>\showglossarycounter</code> : new	254	<code>\glsdesc</code> : made robust	130
<code>\showglossaryentries</code> : new	254	<code>\GLSdescplural</code> : made robust	132
<code>\showglossaryin</code> : new	254	<code>\Glsdescplural</code> : made robust	131
<code>\showglossaryout</code> : new	254	<code>\glsdescplural</code> : made robust	131
<code>\showglossarytitle</code> : new	254	<code>\glsfirst</code> : made robust	127
<code>\showglosymbol</code> : new	252	<code>\GLSfirstplural</code> : made robust	129
<code>\showglosymbolplural</code> : new	252	<code>\Glsfirstplural</code> : made robust	129
<code>\showglotext</code> : new	249	<code>\glsfirstplural</code> : made robust	129
<code>\showglotype</code> : new	250	<code>\glslink</code> : made robust	108
<code>\showglouserI</code> : new	250	<code>\GLSname</code> : made robust	130
<code>\showglouserII</code> : new	250	<code>\Glsname</code> : made robust	130

\glsname: made robust .....	129	\@printglossary: add a way to fetch current entry label .....	186
\GLSpl: made robust .....	124	savenumberlist: new .....	8
\Glspl: made robust .....	123	ucmark: new .....	10
\glspl: made robust .....	123	\gls@defglossaryentry: added numberlist element .....	83
\GLSplural: made robust .....	128	\gls@save@numberlist: new .....	183
\GLSsymbol: made robust .....	132	\gls@wrglossary: added check for glossary file defined .....	177
\Glsymbol: made robust .....	132	\glsdisplaynumberlist: new .....	153
\glssymbol: made robust .....	132	\glsentrycounter: set default value ..	110
\GLSsymbolplural: made robust .....	133	\Glsentryfull: fixed bug (replaced \glsentryshortpl with \glsentryshort) .....	153
\Glsymbolplural: made robust .....	133	\glsentryfullpl: fixed bug (replaced \glsentryshort with \glsentryshortpl) .....	153
\Glstext: made robust .....	127	\glsentrynumberlist: new .....	153
\glstext: made robust .....	126	\glsmoveentry: new .....	85
\GLSuseri: made robust .....	134	\glsresetsubentrycounter: new ...	201
\Glsuseri: made robust .....	134	\ifglshaschildren: new .....	53
\glsuseri: made robust .....	134	\ifglshasparent: new .....	54
\GLSuserii: made robust .....	135	\makeglossaries: added list parser ..	170
\Glsuserii: made robust .....	135	indexonlyfirst: new .....	25
\glsuserii: made robust .....	134	\renewglossarystyle: new .....	209
\GLSuseriii: made robust .....	136	\showglossaryentries: fixed misspelt command .....	254
\Glsuseriii: made robust .....	135	\SmallNewAcronymDef: fixed broken short and long plural .....	243
\glsuseriii: made robust .....	135		
\GLSuseriv: made robust .....	136		
\Glsuseriv: made robust .....	136		
\glsuseriv: made robust .....	136		
\GLSuserv: made robust .....	137		
\Glsuserv: made robust .....	137		
\glsuserv: made robust .....	137		
\GLSuservi: made robust .....	138		
\Glsuservi: made robust .....	138		
\glsuservi: made robust .....	138		
3.02 (2012-05-19)			
\glsnumlistlastsep: new .....	154		
\glsnumlistsep: new .....	154		
3.02 (2012-05-21)			
\@do@wrglossary: changed \@glslocref to \theglsentrycounter .....	180		
\@do@wrglossary: changed \@do@wr@glossary to test for indexonlyfirst option; put old \@do@wr@glossary code into \@do@wrglossary .....	177		
\@gls@missingnumberlist: new .....	66		
\@gls@writefiles: added check for existence of token in case \makeglossaries has been omitted .....	175		
		\@gls@sanitizesort: new .....	19
		\@gls@setupsort@standard: used \@gls@sanitizesort .....	11
		\@printglossary: allow title to override default toctitle .....	185
		General: allow title to set toctitle .....	197
		\glsinlinedescformat: new .....	267
		\glsinlineemptydescformat: new ..	267
		\glsinlinenameformat: new .....	267
		\glsinlinepostchild: new .....	267
		\glsinlinesubdescformat: new ....	267
		\glsinlinesubnameformat: new ....	267
		\glspostinline: replaced “.” with \glspostdescription .....	267
		list: added check for glsnogroupskip ..	268
		altlongragged4col: added check for glsnogroupskip .....	286
		altsuperragged4col: added check for glsnogroupskip .....	304

alttree: added check for		\gls@disablepagerefexpansion: new	177
glsnogroupskip .....	314	\gls@numberpage: new .....	178
index: added check for glsnogroupskip	308	\gls@protected@pagefmts: new ....	177
nogroupskip: new .....	9	\gls@romanpage: new .....	178
long: added check for glsnogroupskip	272	\glsdefmain: added check for doc	
long3col: added check for		package .....	13
glsnogroupskip .....	273	\glsorg@endtheglossary: new .....	5
long4col: added check for		\glsorg@theglossary: new .....	5
glsnogroupskip .....	275	\PrintChanges: new .....	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip .....	283	@@do@wrglossary: add Roman case.	
longragged3col: added check for		Fixed bugs in the else statements ..	179
glsnogroupskip .....	284	\@gls@link: added check for	
no postdot: new .....	9	“nohypertypes” .....	109
tree: added check for glsnogroupskip	309	mcolalttree: replaced ‘2’ with	
treenoname: added check for		\glsmcols .....	291
glsnogroupskip .....	311	mcolindex: replaced ‘2’ with \glsmcols	288
super: added check for glsnogroupskip	294	mcolindexspannav: replaced ‘2’ with	
super3col: added check for		\glsmcols .....	289
glsnogroupskip .....	296	mcoltree: replaced ‘2’ with \glsmcols	289
super4col: added check for		mcoltreenoname: replaced ‘2’ with	
glsnogroupskip .....	297	\glsmcols .....	290
superragged: added check for		mcoltreesspannav: replaced ‘2’ with	
glsnogroupskip .....	301	\glsmcols .....	290
superragged3col: added check for		\gls@protected@pagefmts: added	
glsnogroupskip .....	303	Roman to list .....	177
3.04 (2012-11-11)		\gls@Romanpage: new .....	178
altlist: replaced \newline with		\glsgetgrouplabel: fixed bug (typo in	
paragraph break .....	269	\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 .....	17
\@glslocref .....	180	\GlsDeclareNoHyperList: new .....	17
\@do@wrglossary: modified to		nohypertypes: new .....	17
compensate for possible incorrect		3.06 (2013/06/17)	
page number .....	179	\@xdy@main@language: Changed back to	
\@gls@escbsdq: unsanitize		using \language name .....	26
\gls@numberpage, \gls@alphpage,		\findrootlanguage: Obsoleted .....	49
\gls@Alphpage and		3.07 (2013-07-05)	
\gls@romanpage .....	112	\@gls@link: fixed bug that failed to find	
\@print@glossary: Moved aux write to		entry in list .....	109
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 .....	40
package .....	4	\glspostdescription: Added	
added local key .....	106	spacefactor code .....	9
\gls@Alphpage: new .....	178	\GlsSetXdyCodePage: Added check for	
\gls@alphpage: new .....	177	fontspec .....	49



\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	237	\ifglshasdesc: new	54
\setglossarypreamble: new	38	\ifglshassymbol: new	54
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	286
list: updated list style to use \glossentry and \subglossentry	268	alttree: updated to use \glossentry and \subglossentry	312
listdotted: updated listdotted style to use \glossentry and \subglossentry	270	index: added paragraph break at end of environment	307
altlist: updated altlist style to use \glossentry and \subglossentry	269	updated to use \glossentry and \subglossentry	307
inline: updated inline style to use \glossentry and \subglossentry	266	long: updated to use \glossentry and \subglossentry	272
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	282
\@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	284
updated for \glossentry	86	tree: updated to use \glossentry and \subglossentry	309
\@glossaryentryfield: switched to \glossentry	85	\setglossarystyle: new	208
\@glossarysubentryfield: switched to \subglossentry	85	\setglossentrycompatibility: new	205
General: added nogroupskip key to \printglossary	198	superragged: updated to use \glossentry and \subglossentry	301
removed definition of \@glossaryentryfield	356	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield	356	\@gls@assign@symbolplural@field: new	19
\compatibleglossentry: new	204	\@gls@default@value: new	62
\compatiblesubglossentry: new	205	\Glsentrydesc: made robust	148
\glossaryentryfield: deprecated	205	\Glsentrydescplural: made robust	148
\Glossentrydesc: new	204	\Glsentryfirst: made robust	149
\glossentrydesc: new	204	\Glsentryfirstplural: made robust	150
\Glossentryname: new	204	\Glsentryfull: made robust	153
\glossentryname: new	204	\Glsentryfullpl: made robust	153
\Glossentrysymbol: new	205	\Glsentrylong: made robust	152
\glossentrysymbol: new	204	\Glsentrylongpl: made robust	152
\gls@assign@desc@field: new	19	\Glsentryname: made robust	147
\gls@assign@descplural@field: new	19	\Glsentryplural: made robust	149
\gls@assign@field: new	68	\Glsentryshort: made robust	152
\gls@ifnotmeasuring: new	87	\Glsentryshortpl: made robust	152
\glsaddallunused: new	156	\Glsentrysymbol: made robust	149
\glsexpandfields: new	68	\Glsentrysymbolplural: made robust	149
\glsnoexpandfields: new	69	\Glsentrytext: made robust	148
\glssee: made robust	182	\Glsentryuseri: made robust	150
\glsseeformat: made robust	182	\Glsentryuserii: made robust	151
\glsseeitem: made robust	183	\Glsentryuseriii: made robust	151
\glsseelist: made robust	182	\Glsentryuseriv: made robust	151
\ifglsdessuppressed: new	54	\Glsentryuserv: made robust	151
		\Glsentryuservi: made robust	152
		\glstextup: new	213

\ifglshassymbol: changed test to check for \@gls@default@symbol .....	54	\@Gls@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	121
3.10a (2013-09-28)		change to using \glentryfmt style commands .....	121
\gls@assign@type@field: new .....	19	removed \makefirstuc (now dealt with in \glentryfmt) .....	122
3.10a (2013-10-13)		\@Glspl@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	124
\@gls@keymap: new .....	70	change to using \glentryfmt style commands .....	124
\@gls@provide@newglossary: new ...	57	removed \makefirstuc (now dealt with in \glentryfmt) .....	124
\@gls@writedef: new .....	70	\@acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	355
\@glsdefaultplural: Obsolete .....	66	\@acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	354
\@glsnodelsc: new .....	66	\@gls@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	121
\@print@glossary: Added providecommand code to aux file ..	187	change to using \glentryfmt style commands .....	121
\gls@defglossaryentry: Changed to using \@gls@default@value .....	79	\@gls@noexpand@fields: Fixed bug expand replaced with noexpand ....	67
new .....	79	\@glsdisp: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	125
\gls.writedefhook: new .....	78	change to using \glentryfmt style commands .....	125
\makeglossaries: Added providecommand code to aux file ..	169	\@Glspl@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	123
\new@glossaryentry: new .....	69	change to using \glentryfmt style commands .....	123
\ns@newglossary: added \@gls@provide@newglossary ....	59	General: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext .....	139–145
3.11a (2013-10-15)		changed to just use \Glsentrydescplural .....	131
\@ACRlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	356	changed to just use \glentrydescplural .....	131, 132
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	354	changed to just use \Glsentrydesc .	131
\@Acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	355	changed to just use \glentrydesc .....	130, 131
\@Acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	354		
\@GLS@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert .....	122		
change to using \glentryfmt style commands .....	122		
removed \MakeUppercase (now moved to \glentryfmt) .....	122		
\@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		
\Glsentryfirstplural	129	
changed to just use		
\glentryfirstplural	129	
changed to just use \Glsentryfirst	127	
changed to just use		
\glentryfirst	127, 128	
changed to just use \Glsentryname	130	
changed to just use \glentryname	130	
changed to just use \Glsentryplural	128	
changed to just use \glentryplural	128	
changed to just use		
\Glsentrysymbolplural	133	
changed to just use		
\glentrysymbolplural	133	
changed to just use \Glsentrysymbol	132	
changed to just use		
\glentrysymbol	132, 133	
Changed to just use \Glsentrytext	127	
changed to just use \glentrytext	126	
changed to just use		
\Glsentryuseriii	136	
changed to just use		
\glentryuseriii	135, 136	
changed to just use \Glsentryuserii	135	
changed to just use		
\glentryuserii	134, 135	
changed to just use \Glsentryuseriv	136	
changed to just use		
\glentryuseriv	136, 137	
changed to just use \Glsentryuseri	134	
changed to just use \glentryuseri	134	
changed to just use \Glsentryuservi	138	
changed to just use \glentryuservi	138	
changed to just use \Glsentryuserv	137	
changed to just use \glentryuserv	137	
Now requires textcase	4	
acronymlists: replaced		
\@addtoacronymlists with		
\DeclareAcronymList	17	
\defglsgdisplay: obsoleted	105	
\defglsgdisplayfirst: obsoleted	105	
\defglsgentryfmt: new	57	
\forglsgentries: replaced \ifx with		
\ifdefempty	51	
\glsgassign@desc: new	78	
\glsgdefglossaryentry: Fixed default		
counter if none supplied	82	
\glsgdoentryfmt: new	57	
\glsgdisplay: obsoleted	104	
\glsgdisplayfirst: obsoleted	104	
\glsgenentryfmt: new	99	
\glsggetgrouptitle: Added check in		
case non-Latin alphabet in use	207	
\glsglossarymark: replaced		
\MakeUppercase with		
\mfirstucMakeUppercase	39	
\glsgnavigation: switched to using		
\@glsggetgrouptitle	265	
\ifglsghasdesc: replaced \ifdefempty		
with \ifcseempty	54	
\ifglsghaslong: new	55	
\ifglsghasshort: new	55	
\ifglsghasymbol: replaced		
\ifdefempty with \ifcseempty	54	
\ifglsgused: replaced \ifthenelse with		
\ifbool	52	
\longnewglossaryentry: new	78	
\ns@newglossary: replaced		
\glsgdisplay and		
\glsgdisplayfirst with		
\glsgentryfmt	59	
compatible-3.07: cnew	28	
\SetCustomDisplayStyle: updated to		
use \defglsgentryfmt	247	
\SetDefaultAcronymDisplayStyle:		
changed to use \defglsgentryfmt	232	
\SetDescriptionAcronymDisplayStyle:		
updated to use \defglsgentryfmt	237	
\SetDescriptionDUAAcronymDisplayStyle:		
updated to use \defglsgentryfmt	236	
\SetDescriptionFootnoteAcronymDisplayStyle:		
updated to use \defglsgentryfmt	233	
\SetDUADisplayStyle: updated to use		
\defglsgentryfmt	245	
\SetFootnoteAcronymDisplayStyle:		
updated to use \defglsgentryfmt	240	
\SetSmallAcronymDisplayStyle:		
updated to use \defglsgentryfmt	242	
\setupglossaries: new	30	
\showglolong: new	252	
\showgloshort: new	252	
numbers: new	29	
symbols: new	28	
3.12a (2013-10-16)		
\glsgdefglossaryentry: added		
\glsglabel	79	
\glsgaddkey: new	73	

3.13a (2013-11-05)		long: switched to \tabularnewline ..	272
\@gls@assign@symbol@field: changed		long3col: switched to	
to use \glsetnoexpandfield ....	19	\tabularnewline .....	273
\@gls@assign@symbolplural@field:		long3colheader: switched to	
changed to use		\tabularnewline .....	274
\glsetnoexpandfield .....	19	long3colheaderborder: switched to	
\@gls@link: removed \relax .....	110	\tabularnewline .....	274
\@gls@notranslatorhook: new .....	23	long4col: switched to	
\@gls@setupsort@standard: moved		\tabularnewline .....	275
\@gls@santizesort to		long4colheader: switched to	
\glsprestandardsort .....	11	\tabularnewline .....	275
ucmark: added check for memoir .....	10	longheader: switched to	
see: added \gls@checkseeallowed ...	63	\tabularnewline .....	272
\glossarysection: changed		longheaderborder: switched to	
\glossarymark to		\tabularnewline .....	273
\glsglossarymark .....	39	\SetFootnoteAcronymDisplayStyle:	
\glossarystyle: fixed bug caused by		fixed missing argument bug .....	240
using \ifdef instead of \ifcsdef .	209	super: switched to \tabularnewline .	294
\gls@assign@desc@field: changed to		super3col: switched to	
use \glsetnoexpandfield .....	19	\tabularnewline .....	295
\gls@assign@descplural@field:		super3colheader: switched to	
changed to use		\tabularnewline .....	296
\glsetnoexpandfield .....	19	super4col: switched to	
\gls@assign@name@field: changed to		\tabularnewline .....	297
use \glsetnoexpandfield .....	19	super4colheader: switched to	
\gls@assign@type@field: changed to		\tabularnewline .....	298
use \glsetexpandfield .....	19	super4colheaderborder: switched to	
\gls@checkseeallowed: new .....	64	\tabularnewline .....	298
\glsaddallunused: set default to		superheader: switched to	
\@glo@types .....	156	\tabularnewline .....	295
\Glsentryfull: changed to use		superheaderborder: switched to	
\acrfullformat .....	153	\tabularnewline .....	295
\Glsentryfull: changed to use			
\acrfullformat .....	153	3.14a (2013-11-12)	
\Glsentryfullpl: changed to use		\@glswritefiles: renamed	
\acrfullformat .....	153	\glswritefiles to	
\Glsentryfullpl: changed to use		\@glswritefiles and used	
\acrfullformat .....	153	“savewrites” option to set	
\glsglossarymark: renamed		\glswritefiles .....	175
\glossarymark to		General: new .....	256
\glsglossarymark to avoid conflict		acronyms: new .....	15
with memoir .....	39	\gls@defglossaryentry: added check	
\glsprestandardsort: new .....	10	for existence of default glossary ....	80
\glsetexpandfield: new .....	18	set the default for firstplural to be the	
\glsetnoexpandfield: new .....	19	value of plural .....	82
altsuper4colheader: switched to		xindygloss: new .....	27
\tabularnewline .....	299	\longprovideglossaryentry: new ...	78
altsuper4colheaderborder: switched		compatible-2.07: added check for 2.07	
to \tabularnewline .....	299	before setting 3.07 compatibility ....	28
		notranslate: new .....	24

\provideglossaryentry:new .....	69	index:new .....	29
4.0 (2013-11-14)		\newacronymstyle:new .....	219
\gls@defglossaryentry: added check		long-sc-short:new .....	222
for first key .....	82	long-sc-short-desc:new .....	223
super: fixed typo in \subglossentry		long-short:new .....	220
(\glossentrydesc) .....	294	long-short-desc:new .....	223
4.01 (2013-11-16)		long-sm-short:new .....	222
General: fixed non-value options so that		long-sm-short-desc:new .....	224
they can be passed to document class .	8	long-sp-short-desc:new .....	223
\CustomAcronymFields: inserted		footnote:new .....	227
missing comma .....	247	footnote-desc:new .....	229
4.02 (2013-12-05)		footnote-sc:new .....	228
\@acrfull: now using \acrfullfmt ..	214	footnote-sc-desc:new .....	229
\@gls@indexdef:new .....	29	footnote-sm:new .....	229
\@gls@numbersdef:new .....	29	footnote-sm-desc:new .....	230
\@gls@symbolsdef:new .....	28	\setacronymstyle:new .....	219
General: Removed \acronymfont .	142–146	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new .....	215	Moved check for empty custom text to	
\Acrfullfmt:new .....	215	prevent unwanted parenthetical	
\acrfullfmt:new .....	214	material .....	237
\ACRfullplfmt:new .....	216	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new .....	216	Moved check for empty custom text to	
\acrfullplfmt:new .....	216	prevent unwanted parenthetical	
\acronymentry:new .....	218	material .....	233
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here .....	23	Moved check for empty custom text to	
sc-short-long:new .....	222	prevent unwanted parenthetical	
sc-short-long-desc:new .....	224	material .....	240
\Genacrfullformat:new .....	104	\SetGenericNewAcronym:new .....	217
\genacrfullformat:new .....	104	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new .....	218	Moved check for empty custom text to	
\Genplacrfullformat:new .....	104	prevent unwanted parenthetical	
\genplacrfullformat:new .....	104	material .....	242
\Glsentryfull: bug fix: added missing		dua:new .....	225
\acronymfont .....	153	dua-desc:new .....	227
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont .....	153	option .....	7
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont .....	153	\makeglossaries: made preamble only	170
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont .....	153	General: changed default to \@empty	
\glsgenacfmt:new .....	102	instead of \relax .....	28
\GlsUseAcrEntryDisplayStyle:new ...	220	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new .....	220	\@do@wrglossary: added	
short-long:new .....	221	\glsdetoklabel .....	180
short-long-desc:new .....	224	\@ACRlong: removed \glslabel	
xindynoglsnumbers:new .....	27	(defined in \@gls@link) .....	356
sm-short-long:new .....	223	\@ACRshort: removed \glslabel	
sm-short-long-desc:new .....	225	(defined in \@gls@link) .....	354

\@Acrlong: removed \glslabel (defined in \@gls@link) .....	355	\genplacrfullformat: redefined to use accessibility information .....	353
\@Acrshort: removed \glslabel (defined in \@gls@link) .....	354	\glossentryname: added \glsdetoklabel .....	204
\@GLS@: removed \glslabel (defined in \@gls@link) .....	122	\gls@defglossaryentry: added \glsdetoklabel .....	79
\@GLSpl: removed \glslabel (defined in \@gls@link) .....	125	replaced #1 with \@gls@label .....	80
\@Gls@: removed \glslabel (defined in \@gls@link) .....	121	replaced \ifthenelse with \ifdefequal .....	81
\@Gls@entry@field: new .....	146	\glsadd: added \glsdetoklabel ....	155
\@Glspl@: removed \glslabel (defined in \@gls@link) .....	124	\glsaddkey: switched to using \@gls@field@link .....	73
\@acrlong: removed \glslabel (defined in \@gls@link) .....	355	\glsdetoklabel: new .....	52
\@acrshort: removed \glslabel (defined in \@gls@link) .....	354	\glsdisplaynumberlist: added \glsdetoklabel .....	154
\@gls@: removed \glslabel (defined in \@gls@link) .....	121	\glsdoifexistsorwarn: new .....	53
\@gls@access@display: new .....	342	\glsentryaccess: switched to using \@gls@entry@field .....	340
\@gls@entry@field: new .....	146	\glsentrydescaccess: switched to using \@gls@entry@field .....	341
\@gls@fetchfield: new .....	71	\glsentrydescpluralaccess: switched to using \@gls@entry@field .....	341
\@gls@field@link: new .....	126	\glsentryfirstaccess: switched to using \@gls@entry@field .....	341
\@gls@link: added \glsdetoklabel .	109	\glsentryfirstplural: added \glsdetoklabel .....	150
moved \@gls@link@opts and \@gls@link@label to \@gls@link	109	\glsentrylongaccess: switched to using \@gls@entry@field .....	342
\@gls@writedef: added \glsdetoklabel .....	70	\glsentrylongpluralaccess: switched to using \@gls@entry@field .....	342
\@glsdisp: removed \glslabel (defined in \@gls@link) .....	125	\glsentrypluralaccess: switched to using \@gls@entry@field .....	341
\@glspl@: removed \glslabel (defined in \@gls@link) .....	123	\glsentryshortaccess: switched to using \@gls@entry@field .....	341
\@printglossary: added \glsdetoklabel .....	186	\glsentryshortpluralaccess: switched to using \@gls@entry@field .....	341
General: removed \glslabel (defined in \@gls@link) .....	139	\glsentrysymbolaccess: switched to using \@gls@entry@field .....	341
sc-short-long-desc: redefined to use accessibility information .....	360	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field .....	341
\compatibleglossentry: added \glsdetoklabel .....	336	\glsentrytextaccess: switched to using \@gls@entry@field .....	341
\compatiblesubglossentry: added \glsdetoklabel .....	337	\glsngenacfmt: redefined to use accessibility information .....	351
\Genacrfullformat: redefined to use accessibility information .....	353	\glsgenentryfmt: redefined to use accessibility information .....	348
\genacrfullformat: redefined to use accessibility information .....	353		
\Genplacrfullformat: redefined to use accessibility information .....	353		

\glshyperlink: added		long-sc-short-desc: redefined to use	
\glsdetoklabel .....	154	accessibility information .....	359
\glslocalreset: added		long-short: redefined to use	
\glsdetoklabel .....	87	accessibility information .....	357
\glslocalunset: added		long-short-desc: redefined to use	
\glsdetoklabel .....	88	accessibility information .....	359
\glsmoveentry: added		long-sm-short-desc: redefined to use	
\glsdetoklabel .....	85	accessibility information .....	359
replaced \ifthenelse with		footnote: redefined to use accessibility	
\ifdefequal .....	85	information .....	363
\glsrefentry: added \glsdetoklabel	202	footnote-desc: redefined to use	
\glsreset: added \glsdetoklabel ...	87	accessibility information .....	365
\glsseelist: added \expandafter		footnote-sc: redefined to use	
commands .....	183	accessibility information .....	365
\glsstepentry: added		footnote-sc-desc: redefined to use	
\glsdetoklabel .....	202	accessibility information .....	366
\glsstepsubentry: added		footnote-sm: redefined to use	
\glsdetoklabel .....	202	accessibility information .....	365
\glsunset: added \glsdetoklabel ...	88	footnote-sm-desc: redefined to use	
short-long: commented spurious EOL	222	accessibility information .....	366
redefined to use accessibility		\renewacronymstyle: new .....	219
information .....	358	\showglocounter: added	
short-long-desc: redefined to use		\glsdetoklabel .....	250
accessibility information .....	360	\showglodesc: added \glsdetoklabel	251
\ifglshdescsuppressed: added		\showglodescaccess: added	
\glsdetoklabel .....	54	\glsdetoklabel .....	372
fixed typo .....	54	\showglodescplural: added	
\ifglshentryexists: added		\glsdetoklabel .....	252
\glsdetoklabel .....	52	\showglodescpluralaccess: added	
\ifglshaschildren: added		\glsdetoklabel .....	372
\glsdetoklabel .....	53	\showglofirst: added	
\ifglshasdesc: added		\glsdetoklabel .....	250
\glsdetoklabel .....	54	\showglofirstaccess: added	
\ifglshasfield: new .....	55	\glsdetoklabel .....	372
\ifglshaslong: added		\showglofirstpl: added	
\glsdetoklabel .....	55	\glsdetoklabel .....	250
\ifglshasparent: added		\showglofirstpluralaccess: added	
\glsdetoklabel .....	54	\glsdetoklabel .....	372
\ifglshasshort: added		\showgloflag: added \glsdetoklabel	253
\glsdetoklabel .....	55	\showgloindex: added	
\ifglshassymbol: added		\glsdetoklabel .....	253
\glsdetoklabel .....	54	\showglolevel: added	
replaced \ifcempty with		\glsdetoklabel .....	249
\ifdefempty and replaced \ifx with		\showglolong: added \glsdetoklabel	252
\ifdefequal .....	54	\showglolongaccess: added	
\ifglshused: added \glsdetoklabel ..	52	\glsdetoklabel .....	373
sm-short-long-desc: redefined to use		\showglolongpluralaccess: added	
accessibility information .....	360	\glsdetoklabel .....	373
		\showglongname: added \glsdetoklabel	251

\showglongnameaccess: added	4.04 (2014-03-04)	
\glsdetoklabel .....	372	\@gls@getcounterprefix: added warning if no prefix can be formed . 181
\showgloparent: added		4.04 (2014-03-06)
\glsdetoklabel .....	249	\@@gls@noidx@nosanitizesort: new . 20
\showgloplural: added		\@@gls@noidx@sanitizesort: new ... 20
\glsdetoklabel .....	249	\@@gls@nosanitizesort: new ..... 20
\showglopluralaccess: added		\@@gls@sanitizesort: new ..... 20
\glsdetoklabel .....	372	\@glo@addchildren: new ..... 188
\showgloshort: added		\@glo@do@sortentries: new ..... 189
\glsdetoklabel .....	252	\@glo@grabfirst: new ..... 194
\showgloshortaccess: added		\@glo@sortedinsert: new ..... 190
\glsdetoklabel .....	373	\@glo@sortentries: new ..... 188
\showgloshortpluralaccess: added		\@glo@sorthandler@case: new ..... 190
\glsdetoklabel .....	373	\@glo@sorthandler@letter: new ... 190
\showglosort: added \glsdetoklabel	252	\@glo@sorthandler@nocase: new ... 190
\showglosymbol: added		\@glo@sorthandler@word: new ..... 190
\glsdetoklabel .....	252	\@glo@sortmacro@case: new ..... 192
\showglosymbolaccess: added		\@glo@sortmacro@def: new ..... 192
\glsdetoklabel .....	372	\@glo@sortmacro@def@do: new ..... 193
\showglosymbolplural: added		\@glo@sortmacro@letter: new ..... 191
\glsdetoklabel .....	252	\@glo@sortmacro@nocase: new ..... 192
\showglosymbolpluralaccess: added		\@glo@sortmacro@standard: new ... 191
\glsdetoklabel .....	372	\@glo@sortmacro@use: new ..... 193
\showglotext: added \glsdetoklabel	249	\@glo@sortmacro@word: new ..... 191
\showglotextaccess: added		\@gls@noidx@do: new ..... 195
\glsdetoklabel .....	372	\@gls@noidx@getgrouptitle: new .. 207
\showglotype: added \glsdetoklabel	250	\@gls@noref@warn: new ..... 174
\showglouser: added		\@gls@reference: new ..... 197
\glsdetoklabel .....	250	\@gls@warnonglossdefined: new .... 18
\showglouserii: added		\@gls@warnontheGLOSSdefined: new . 18
\glsdetoklabel .....	250	\@no@makeglossaries: new ..... 174
\showglouseriii: added		\@print@glossary: new ..... 186
\glsdetoklabel .....	251	\@print@noidx@glossary: new ..... 193
\showglouseriv: added		\@printgloss@setsort: new ..... 184
\glsdetoklabel .....	251	\@printglossary: new ..... 185
\showglouserv: added		General: added sort key to printgloss
\glsdetoklabel .....	251	group ..... 200
\showglouservi: added		\compatibleglossentry: changed
\glsdetoklabel .....	251	\newcommand to \def as is may or may not be defined ..... 336
dua: fixed bug in \acrfullfmt .....	226	\compatiblesubglossentry: changed
fixed bug in \Acrrfullplfmt .....	226	\newcommand to \def as is may or may not be defined ..... 337
fixed bug in \acrfullplfmt .....	226	\defglsdisplayfirst: fixed unwanted space ..... 105
redefined to use accessibility information .....	361	\glo@grabfirst: new ..... 194
dua-desc: commented spurious EOL ..	227	\gls@defglossaryentry: replaced \ifx with \ifdefvoid ..... 84
redefined to use accessibility information .....	363	



\glsnoidxdisplayloc: new .....	197	\@ACRshort: added	
\glsnoidxdisplayloclisthandler:		\do@gl@link@checkfirsthyper	354
new .....	196	\@Acrlong: added	
\glsnoidxloclist: new .....	196	\do@gl@link@checkfirsthyper	355
\glsnoidxloclisthandler: new ....	196	\@Acrshort: added	
\glsnoidxstripaccents: new .....	20	\do@gl@link@checkfirsthyper	354
alttree: moved hangindent and		\@GLS@: moved \glsifhyper .....	122
parindent assignments outside level		moved check for first use to	
test .....	312	\@gl@link .....	122
\makeglossaries: Moved definition of		\@GLSpl: moved \glsifhyper .....	125
\glswrite to \makeglossaries ..	168	moved check for first use to	
\makenoidxglossaries: new .....	171	\@gl@link .....	125
\printglossary: changed to use new		\@Gls@: moved \glsifhyper .....	121
\@printglossary .....	184	moved check for first use to	
\printnoidxglossaries: new .....	184	\@gl@link .....	121
\printnoidxglossary: new .....	184	\@Glspl@: moved \glsifhyper .....	124
\showgloclist: new .....	253	moved check for first use to	
\warn@noprintglossary: Activate		\@gl@link .....	124
warning in \makeglossaries ....	183	\@acrlong: added	
\writeist: checked for definition of		\do@gl@link@checkfirsthyper	355
\glswrite .....	157, 162	\@acrshort: added	
4.06 (2014-03-12)		\do@gl@link@checkfirsthyper	353
\@GLS@: added \glsifhyper .....	122	\@closegls: new .....	167
\@GLSpl: added \glsifhyper .....	125	\@gls@: moved \glsifhyper .....	121
\@Gls@: added \glsifhyper .....	121	moved check for first use to	
\@Glspl@: added \glsifhyper .....	124	\@gl@link .....	121
\@gls@: added \glsifhyper .....	121	\@gls@automake: new .....	167
\@gls@numbersdef: added hook to set		\@gls@doautomake: new .....	28
toc title .....	29	\@gls@field@link: added assignment	
\@gls@symbolsdef: added hook to set		of	
toc title .....	28	\do@gl@link@checkfirsthyper	126
\@gl@disp: added \glsifhyper .....	125	\@gls@forbidtexext: new .....	57
\@gl@spl@: added \glsifhyper .....	123	\@gls@hyp@opt: new .....	107
General: added \glsifhyper ....	139–146	\@gl@link: removed redundancy ....	109
acronym: added hook to set toc title ....	14	renamed \gls@type to \glstype ...	109
acronyms: added hook to set toc title ...	15	\@gl@link@checkfirsthyper: new .	108
\gl@defmain: added hook to set toc title	14	\@gl@disp: moved \glsifhyper .....	125
4.07 (2014-04-04)		moved check for first use to	
\@glossarysection: added optional		\@gl@link .....	125
argument when using unstarred		\@gl@spl@: moved \glsifhyper .....	123
version .....	40	moved check for first use to	
\@gls@noidx@do: added \global in case		\@gl@link .....	123
it's used in a tabular-like style .....	195	\@ignored@glossaries: new .....	61
\Acrfullplfmt: fixed no case change		General: added entrycounter option to	
bug .....	216	printgloss family .....	198
\gl@setentryfield: new .....	146	added nopostdot option to	
4.08 (2014-07-30)		printgloss family .....	198
\@ACRlong: added		added subentrycounter option to	
\do@gl@link@checkfirsthyper	355	printgloss family .....	199

explicitly initialise hyper key .....	106	removed \@sGLSuserii .....	135
moved \glsifhyper .....	139–146	removed \@sGlsuserii .....	135
removed \@sACRlongpl .....	145	removed \@sglsuserii .....	134
removed \@sAcrlongpl .....	145	removed \@sGLSuseriv .....	137
removed \@sacrlongpl .....	144	removed \@sGlsuseriv .....	136
removed \@sACRlong .....	143	removed \@sglsuseriv .....	136
removed \@sAcrlong .....	143	removed \@sGLSuseri .....	134
removed \@sacrlong .....	142	removed \@sGlsuseri .....	134
removed \@sACRshortpl .....	141	removed \@sglsuseri .....	134
removed \@sAcrshortpl .....	141	removed \@sGLSservi .....	138
removed \@sacrshortpl .....	140	removed \@sGlsservi .....	138
removed \@sACRshort .....	140	removed \@sglsuseri .....	138
removed \@sAcrshort .....	139	removed \@sGLSserv .....	137
removed \@sacrshort .....	138	removed \@sGlsuseri .....	137
removed \@sgls@link .....	108	removed \@sglsuseri .....	137
removed \@sGLSdescplural .....	132	removed \@sGLS .....	122
removed \@sGlsdescplural .....	131	removed \@sGls .....	121
removed \@sglsdescplural .....	131	removed \@sgls .....	120
removed \@sGLSdesc .....	131	removed \@thirdofthree (defined in	
removed \@sGlsdesc .....	131	kernel) .....	120
removed \@sglsdesc .....	130	removed sPGLS .....	261
removed \@sglsdisp .....	125	removed sPgls .....	259
removed \@sGLSfirstplural .....	129	removed spgls .....	258
removed \@sGlsfirstplural .....	129	removed sPGLSpl .....	261
removed \@sglsfirstplural .....	129	removed sPglspl .....	260
removed \@sGLSfirst .....	128	removed spglspl .....	259
removed \@sGlsfirst .....	127	\ACRfull: removed \s@ACRfull .....	215
removed \@sglsfirst .....	127	switched to using \@gls@hyp@opt ..	215
removed \@sGLSname .....	130	\Acrfull: removed \@sAcrfull .....	215
removed \@sGlsname .....	130	switched to using \@gls@hyp@opt ..	215
removed \@sglsname .....	129	\acrfull: removed \@sacrfull .....	214
removed \@sGLSplural .....	128	switched to using \@gls@hyp@opt ..	214
removed \@sGlsplural .....	128	\ACRfullpl: removed \s@ACRfullpl ..	216
removed \@sglsplural .....	128	switched to using \@gls@hyp@opt ..	216
removed \@sGLSpl .....	124	\Acrfullpl: removed \s@Acrfullpl ..	216
removed \@sGlspl .....	124	switched to using \@gls@hyp@opt ..	216
removed \@sglspl .....	123	\acrfullpl: removed \s@acrfullpl ..	215
removed \@sGLSsymbolplural .....	133	switched to using \@gls@hyp@opt ..	215
removed \@sGlsymbolplural .....	133	\ACRlong: switched to using	
removed \@sglsymbolplural .....	133	\@gls@hyp@opt .....	143
removed \@sGLSsymbol .....	132	\Acrlong: switched to using	
removed \@sGlsymbol .....	132	\@gls@hyp@opt .....	142
removed \@sglsymbol .....	132	\acrlong: switched to using	
removed \@sGLStext .....	126	\@gls@hyp@opt .....	142
removed \@sGlstext .....	127	\ACRlongpl: switched to using	
removed \@sglstext .....	126	\@gls@hyp@opt .....	145
removed \@sGLSuseriii .....	136	\Acrlongpl: switched to using	
removed \@sGlsuseriii .....	135	\@gls@hyp@opt .....	144
removed \@sglsuseriii .....	135		

\acrlongpl: switched to using \@gls@hyp@opt .....	144	\GLSfirst: switched to using \@gls@hyp@opt .....	127
\ACRshort: switched to using \@gls@hyp@opt .....	139	\Glsfirst: switched to using \@gls@hyp@opt .....	127
\Acrshort: switched to using \@gls@hyp@opt .....	139	\glsfirst: switched to using \@gls@hyp@opt .....	127
\acrshort: switched to using \@gls@hyp@opt .....	138	\GLSfirstplural: switched to using \@gls@hyp@opt .....	129
\ACRshortpl: switched to using \@gls@hyp@opt .....	141	\Glsfirstplural: switched to using \@gls@hyp@opt .....	129
\Acrshortpl: switched to using \@gls@hyp@opt .....	141	\glsfirstplural: switched to using \@gls@hyp@opt .....	129
\acrshortpl: switched to using \@gls@hyp@opt .....	140	\glsifhyper: deprecated .....	107
\forallacronyms: new .....	50	\glslink: switched to using \@gls@hyp@opt .....	108
\GLS: switched to using \@gls@hyp@opt	122	\glslinkcheckfirsthyperhook: new	109
\Gls: switched to using \@gls@hyp@opt	121	\glslinkvar: new .....	107
\gls: switched to using \@gls@hyp@opt	120	\GLSname: switched to using \@gls@hyp@opt .....	130
\gls@defglossaryentry: added check for ignored glossary .....	80	\Glsname: switched to using \@gls@hyp@opt .....	130
\gls@istfilebase: new .....	36	\glsname: switched to using \@gls@hyp@opt .....	129
\glsaddkey: removed \@sGLS@user@<key> .....	74	\GLSpl: switched to using \@gls@hyp@opt .....	124
removed \@sGls@user@<key> .....	74	\Glspl: switched to using \@gls@hyp@opt .....	123
removed \@sgls@user@<key> .....	74	\glspl: switched to using \@gls@hyp@opt .....	123
switched to using \@gls@hyp@opt ...	74	\GLSplural: switched to using \@gls@hyp@opt .....	128
\GLSdesc: switched to using \@gls@hyp@opt .....	131	\Glsplural: switched to using \@gls@hyp@opt .....	128
\Glsdesc: switched to using \@gls@hyp@opt .....	130	\glsplural: switched to using \@gls@hyp@opt .....	128
\glsdesc: switched to using \@gls@hyp@opt .....	130	\glsspace: new .....	214
\GLSdescplural: switched to using \@gls@hyp@opt .....	132	\GLSsymbol: switched to using \@gls@hyp@opt .....	132
\Glsdescplural: switched to using \@gls@hyp@opt .....	131	\Glsymbol: switched to using \@gls@hyp@opt .....	132
\glsdescplural: switched to using \@gls@hyp@opt .....	131	\glsymbol: switched to using \@gls@hyp@opt .....	132
\glsdisablehyper: added \KV@glslink@hyperfalse to definition .....	120	\GLSsymbolplural: switched to using \@gls@hyp@opt .....	133
\glsdisp: switched to using \@gls@hyp@opt .....	125	\Glsymbolplural: switched to using \@gls@hyp@opt .....	133
\glsdohyperlink: new .....	119	\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 .....	126	\ns@newglossary: added \@glotype@<name>@log .....	59
\Glstext: switched to using \@gls@hyp@opt .....	127	new .....	58
\glstext: switched to using \@gls@hyp@opt .....	126	\p@gls@hyp@opt: new .....	107
\glstreenamefmt: new .....	306	\PGLS: changed to use \@gls@hyp@opt	261
\GLSuseri: switched to using \@gls@hyp@opt .....	134	\PglS: changed to use \@gls@hyp@opt	259
\Glsuseri: switched to using \@gls@hyp@opt .....	134	\pglS: changed to use \@gls@hyp@opt	258
\glSuseri: switched to using \@gls@hyp@opt .....	134	\PGLSpl: changed to use \@gls@hyp@opt .....	261
\GLSuserii: switched to using \@gls@hyp@opt .....	135	\PglSpl: changed to use \@gls@hyp@opt .....	260
\Glsuserii: switched to using \@gls@hyp@opt .....	135	\pglSpl: changed to use \@gls@hyp@opt .....	259
\glSuserii: switched to using \@gls@hyp@opt .....	135	\s@gls@hyp@opt: new .....	107
\glSuserii: switched to using \@gls@hyp@opt .....	134	\s@newglossary: new .....	58
\GLSuseriii: switched to using \@gls@hyp@opt .....	136	automake: new .....	27
\Glsuseriii: switched to using \@gls@hyp@opt .....	135	4.09 (2014-08-12)	
\glSuseriii: switched to using \@gls@hyp@opt .....	135	\glsaddkey: fixed bug in user commands	74
\GLSuseriv: switched to using \@gls@hyp@opt .....	136	4.10 (2014-08-27)	
\Glsuseriv: switched to using \@gls@hyp@opt .....	136	\@Gls@acrentryname: new .....	147
\glSuseriv: switched to using \@gls@hyp@opt .....	136	\@Gls@entryname: new .....	147
\GLSuserv: switched to using \@gls@hyp@opt .....	137	\@gls@glossary: Renamed \@glossary to \@gls@glossary .....	176
\Glsuserv: switched to using \@gls@hyp@opt .....	137	\glSpercentchar: new .....	157
\glSuserv: switched to using \@gls@hyp@opt .....	137	\glStildechar: new .....	157
\GLSuservi: switched to using \@gls@hyp@opt .....	138	alttree: moved space after symbol	313, 314
\Glsuservi: switched to using \@gls@hyp@opt .....	138	4.11 (2014-09-01)	
\glSuservi: switched to using \@gls@hyp@opt .....	138	\@do@wrglossary: added hook .....	179
\ifignoredglossary: new .....	61	sanitize: none option .....	23
altlongragged4col: fixed bug that displayed description instead of symbol .....	286	\glS@wrglossary: renamed from \@wrglossary to \glS@wrglossary	176
\newglossary: added starred version ..	58	\glSaddprotectedpagefmt: new ....	178
\newignoredglossary: new .....	60	\glSbackslash: new .....	156
		4.12 (2014-11-22)	
		\@glS@addpredefinedattributes: Added glSignore attribute .....	44
		\@glS@adjustmode: new .....	155
		\@glS@notranslatorhook: removed ...	23
		\@glS@toc: added \protect to \numberline .....	41
		\@glS@usetranslator: new .....	23
		\glSacrpluralsuffix: new .....	32
		\glSadd: added check for vertical mode	155
		\glSaddallunused: replaced @gobble with glSignore .....	156
		\glSifusedtranslatordict: new ....	23
		\glSignore: new .....	156
		\glSupacrpluralsuffix: new .....	32
		\ProvidesGlossariesLang: new .....	33

\RequireGlossariesLang: new	33	4.16 (2015-07-08)	
4.13 (2015-02-03)			\@ACRlong: added \glspostlinkhook 356
\indexspace: new	268, 287, 306		\@ACRshort: added \glspostlinkhook 355
4.14 (2015-02-28)			\@Acrlong: added \glspostlinkhook 355
\@glslocalreset: new	89		\@Acrshort: added \glspostlinkhook 354
\@glslocalunset: new	88		\@GLS@: added \glspostlinkhook ... 123
\@glsreset: new	89		\@GLSpl: added \glspostlinkhook .. 125
\@glsunset: new	88		\@Gls@: added \glspostlinkhook ... 122
\@newglossaryentry@defcounters:			\@GLspl@: added \glspostlinkhook . 124
new	90		\@acrlong: added \glspostlinkhook 355
\cGls: new	93		\@acrshort: added \glspostlinkhook 354
\cGls@: new	93		\@gls@: added \glspostlinkhook ... 121
\cGLspl@: new	94		\@gls@@link: added
\cgls: new	93		\glspostlinkhook ..... 108
\cgls@: new	93		\@gls@field@link: added
\cglspl: new	94		\glspostlinkhook ..... 126
\cglspl@: new	94		\@gls@link: moved definition of
\@gls@entry@count: new	93		\glsifhyperon outside of this
\@gls@increment@currcount: new	92		macro ..... 110
\@gls@local@increment@currcount:			\@glsdisp: added \glspostlinkhook 126
new	92		\@glspl@: added \glspostlinkhook . 123
\@gls@write@entrycounts: new	92		General: added \glspostlinkhook 139-146
\@glslocalreset: new	89		\glsacspace: new ..... 221
\@glslocalunset: new	88		\glsadd: changed \@do@wrglossary to
\@glsreset: new	89		\@do@wrglossary ..... 155
\@glsunset: new	88		\glsfielddef: new ..... 76
\@newglossaryentry@defcounters:			\glsfieldedef: new ..... 75
new	84		\glsfieldfetch: new ..... 76
\cGls: new	93		\glsfieldgdef: new ..... 76
\cgls: new	93		\glsfieldxdef: new ..... 75
\cGlsformat: new	94		\glsifhyperon: moved definition of
\cglsformat: new	93		\glsifhyperon ..... 109
\cGLspl: new	94		\glslinkpostsetkeys: new ..... 109
\cglspl: new	94		\glspostlinkhook: new ..... 108
\cGLsplformat: new	95		\glswriteentry: new ..... 177
\cglsplformat: new	94		\ifglsfieldcseq: new ..... 77
\@gls@defdocnewglossaryentry: new	69		\ifglsfielddefeq: new ..... 77
\@glsenableentrycount: new	90		\ifglsfieldefeq: new ..... 77
\glslocalreset: switched to			long-sp-short: new ..... 220
\@glslocalreset	87		\showglofield: new ..... 253
\glslocalunset: switched to			
\@glslocalunset	88	4.18 (2015-09-09)	
\glsreset: switched to \@glsreset	87		General: split mfirstuc into separate
\glsunset: switched to \@glsunset	88		bundle ..... 4
4.15 (2015-03-16)		4.19 (2015-10-31)	
General: bug fix replaced \@glo@type			\glstreenamibox: new ..... 312
with \glstype	145	4.19 (2015-11-22)	
4.16 (2015-06-18)			\@gls@link@nocheckfirsthyper: new 126
\glsaddstoragekey: new	72		\@gls@preglossaryhook: new ..... 184

\@printglossary: added		\glslistgroupheaderfmt: new	268
\@gls@preglossaryhook	186	\glslistnavigationitem: new	268
\do@glsglisablehyperinlist: new	109	\glstreegroupheaderfmt: new	306
\doifglossarynoexistsordo: new	53	\glstreenavigationfmt: new	306
\gls@gobbleopt: new	58	\ifglswrallowprimitivemods: new	178
\glsdoifexistsordo: new	53	list: fixed missing space before	
4.20 (2015-11-30)		description	268
\@gls@link: added		long: fixed typo in \glossentrydesc	272
\@gls@setdefault@glslink@opts	109	super4col: fixed bug in \glossentry	297
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
\gls@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	64	nogroupskip changed	286
\glsdonohyperlink: new	119	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	304
\@printglossary: warn if no style has		long: check for nogroupskip changed	272
been set	185	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	278
assignment	139–145	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed	273
already set	209	long3col-booktabs: check for	
\glsLTpenaltycheck: new	280	nogroupskip changed	278
\glspatchLToutput: new	281	long4col: check for nogroupskip	
\glspenaltygroupskip: new	281	changed	275
altlong4col-booktabs: new	279	long4col-booktabs: check for	
altlongragged4col-booktabs: new	280	nogroupskip changed	279
long-booktabs: new	277	longragged: check for nogroupskip	
long3col-booktabs: new	278	changed	283
long4col-booktabs: new	278	longragged3col: check for nogroupskip	
longragged-booktabs: new	279	changed	284
longragged3col-booktabs: new	280	super: check for nogroupskip changed	294
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set	208	changed	296
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@wrglossary: added check for		changed	297
\@arabic	179	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	301
modifications and added arabic case	179	superragged3col: check for	
mcolalttreespannav: new	292	nogroupskip changed	303
mcolindexspannav: new	288	4.24 (2016-05-27)	
mcoltreenonamespannav: new	291	\@gls@extramakeindexopts: new	166
mcoltreespannav: new	290	\@gls@glossary: added check for debug	
\gls@arabicpage: new	178	mode	176
\gls@protected@pagefmts: added		\@gls@see@noindex: new	5
arabic to list	177	debug: new	5
\glstrytitlecase: new	150	seenoindex: new	6
\glsfindwidesttoplevelname: new	311	\glsnomakeindexwarning: new	41

\GlsSetQuote: new .....	163	\@gls@currentlettergroup outside of the glossary environment .....	193
\GlsSetWriteIstHook: new .....	163	General: added check for	
4.25 (2016-06-09)		\@glsxtr@doaccsupp .....	336
\@gls@enablesavenonumberlist: new	64	\glsnavhyperlinkname: new .....	263
\@gls@initnonumberlist: new .....	64	4.30 (2017-06-11)	
\@gls@savenonumberlist: new .....	64	\@glo@autosee: new .....	84
4.26 (2016-10-12)		\@glo@autoseehook: new .....	84
\@glossary@default@style: added		\@glo@check@sortallowed: new .....	11
check for classicthesis .....	7	\@gls@noidx@do: letter group	
mcolindex: replaced \@idxitem with		assignment made global .....	195
\glstreeitem .....	288	\@gls@setupsort@def: added check for	
mcolindexspannav: replaced \@idxitem		register .....	12
with \glstreeitem .....	289	\@gls@setupsort@none: new .....	13
\glstreechildpredesc: new .....	307	\@xdycrossrefhook: new .....	47
\glstreeitem: new .....	306	\@xdylocationclassorder: bug fix:	
\glstreepredesc: new .....	306	changed \edef to \def .....	47
\glstreesubitem: new .....	306	\glosortentrieswarning: new .....	18
\glstreesubsubitem: new .....	306	\gls@set@xr@key: new .....	63
4.28 (2017-01-07)		\gls@xr@key: new .....	63
\glspatchtabularx: new .....	87	\GlsAddXdyLocation: bug fix: changed	
4.29 (2017-01-19)		#1 to #2 .....	47
\@gls@noidx@do: current letter group		\glsnoidxstripaccents: added \a ...	21
assignment made global .....	195	added \TH, \dh and \DH .....	21
\@print@noidx@glossary: moved			
definition of			

# 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	
\!	115, 116
\"	20, 113–116, 118
\#	160
\%	157, 162, 320, 321
\&	32, 154
\'	20
\.	9, 21
\=	21
\?	113–115, 165
\@@delimN	211
\@@do@wrglossary	171, 180
\@@do@wrglossary	155, 177
\@@glo@assign@sortkey	172
\@@glo@list	51
\@@glo@sort	20
\@@glo@type	184
\@@glossarysec	6, 40
\@@glossaryseclabel	7, 40, 198
\@@glossarysecstar	7, 40, 198
\@@gls@checkactual	117
\@@gls@checkbar	116
\@@gls@checkescactual	115
\@@gls@checkescbar	115
\@@gls@checkesclevel	116
\@@gls@checkescquote	114, 165, 166
\@@gls@checklevel	117
\@@gls@checkquote	113, 114, 164
\@@gls@default@entryfmt	96, 105
\@@gls@expand@field	19, 68, 72, 73, 232, 234–236, 239, 241, 244–246, 367–371
\@@gls@extramakeindexopts	163, 169
\@@gls@fixbraces	182
\@@gls@noexpand@field	19, 67
\@@gls@noidx@no@sanitizesort	20
\@@gls@noidx@nosanitizesort	173
\@@gls@nosanitizesort	19, 173
\@@gls@sanitizesort	19, 173
\@@gls@xdycheckbackslash	118, 119
\@@gls@xdycheckquote	118
\@@gls@localreset	89, 91
\@@gls@localunset	88, 91
\@@gls@reset	89, 91
\@@gls@unset	88, 91
\@@newglossaryentry@defcounters	90
\@@this@glo@	51
\@ACRfull	215
\@ACRfullpl	216
\@ACRlong	143, 215
\@ACRlongpl	145, 216
\@ACRshort	140, 215
\@ACRshortpl	141, 216
\@Acrfull	215
\@Acrfullpl	216
\@Acrlong	143, 215
\@Acrlongpl	145, 216
\@Acrshort	139
\@Acrshortpl	141
\@Alph	178, 179
\@GLS	122
\@GLS@	122, 261
\@GLSdesc	131
\@GLSdesc@	131
\@GLSdescplural	132
\@GLSdescplural@	132
\@GLSfirst	127, 128
\@GLSfirst@	128
\@GLSfirstplural	129
\@GLSfirstplural@	129
\@GLSname	130
\@GLSname@	130
\@GLSpl	124
\@GLSpl@	124, 262
\@GLSplural	128



\@GLSplural@	128	\@Glsuseriii@	135, 136
\@GLSsymbol	132	\@Glsuseriv	136
\@GLSsymbol@	132, 133	\@Glsuseriv@	136
\@GLSsymbolplural	133	\@Glsuserv	137
\@GLSsymbolplural@	133	\@Glsuserv@	137
\@GLStext	126	\@Glsuservi	138
\@GLStext@	126, 127	\@Glsuservi@	138
\@GLSuseri	134	\@Mi	281
\@GLSuseri@	134	\@PGLS	261
\@GLSuserii	135	\@PGLS@	261
\@GLSuserii@	135	\@PGLSpl	261
\@GLSuseriii	136	\@PGLSpl@	261
\@GLSuseriii@	136	\@Pgls	259
\@GLSuseriv	136, 137	\@Pgls@	259
\@GLSuseriv@	137	\@Pglspl	260
\@GLSuserv	137	\@Pglspl@	260
\@GLSuserv@	137	\@Roman	178, 179
\@GLSuservi	138	\@acrfull	214
\@GLSuservi@	138	\@acrfullpl	215, 216
\@Gls	121	\@acrlong	142, 214
\@Gls@	91, 93, 121, 260	\@acrlongpl	144, 216
\@Gls@acrentryname	217	\@acrshort	138, 214, 215
\@Gls@entry@field	73, 147–152	\@acrshortpl	140, 216
\@Gls@entryname	147, 217	\@addtoacronymlists	15, 16
\@Glsdesc	130, 131	\@after	16
\@Glsdesc@	131	\@afterheading	269, 324
\@Glsdescplural	131	\@alph	177, 179
\@Glsdescplural@	131	\@arabic	178, 179
\@Glsfirst	127	\@auxout	57, 59, 92, 169, 171, 174, 183, 187, 263
\@Glsfirst@	127	\@backslashchar	112, 118, 119
\@Glsfirstplural	129	\@before	16
\@Glsfirstplural@	129	\@bsphack	176
\@Glsname	130	\@cGls	93
\@Glsname@	130	\@cGls@	91, 93
\@Glspl	123, 124	\@cGlspl	94
\@Glspl@	92, 94, 124, 260, 261	\@cGlspl@	92, 94
\@Glsplural	128	\@cclv	281
\@Glsplural@	128	\@cgls	93
\@Glsymbol	132	\@cgls@	91, 93
\@Glsymbol@	132	\@cglspl	94
\@Glsymbolplural	133	\@cglspl@	91, 94
\@Glsymbolplural@	133	\@chapter	31
\@Glstext	127	\@classoptionslist	29
\@Glstext@	127	\@closegls	167, 168
\@Glsuseri	134	\@colht	281
\@Glsuseri@	134	\@colroom	281
\@Glsuserii	135	\@currentlabelname	7, 198
\@Glsuserii@	135	\@curroptions	29
\@Glsuseriii	135	\@declaredoptions	29

\@delimN .....	211	\@glo@counterprefix	174, 179–181, 208, 211
\@delimR .....	210	\@glo@default@sorttype ..	10, 172, 191, 192
\@disable@onlypremakeg .....	169	\@glo@defaultcounter .....	82
\@disable@premakecs .....	31	\@glo@desc .....	61, 78–80, 83
\@disabled@gl\$addxdycounters .....	44	\@glo@descaccess .....	338–340
\@do@addcounter .....	42	\@glo@descplural .....	62, 78, 79
\@do@auxoutstuff .....	187	\@glo@descpluralaccess .....	338–340
\@do@glossentry .....	204, 336, 337	\@glo@do@sortentries .....	188
\@do@gl\$@getcounterprefix .....	180	\@glo@entry .....	155, 156
\@do@gl\$@islistofacronyms .....	16	\@glo@entryprefix .....	256
\@do@gl\$see .....	84	\@glo@entryprefixfirst .....	256
\@do@ifinlist .....	42	\@glo@entryprefixfirstplural ..	256, 257
\@do@newglossaryentry .....	218, 232, 234–236, 238–241, 243–248, 367–371	\@glo@entryprefixplural .....	256
\@do@seeglossary .....	171, 182	\@glo@esclabel .....	85, 86
\@do@subglossentry .....	205, 337	\@glo@etext .....	96–98
\@do@wrglossary .....	110	\@glo@first .....	62, 79, 82, 83, 243, 371
\@do@writeaux@info .....	183	\@glo@firstaccess .....	337, 339, 340
\@ehc .....	281	\@glo@firstplural .....	62, 79, 82, 371
\@empty ...	13, 15, 28, 29, 31, 42, 43, 47, 50, 51, 80, 81, 86, 110, 111, 121–125, 139– 145, 158, 161, 163, 167, 168, 175, 176, 181, 198, 201, 208, 233, 235, 237, 239– 242, 244, 246, 248, 318, 320, 322, 354–356	\@glo@firstpluralaccess .....	338–340
\@end@fixbraces .....	182	\@glo@grabfirst .....	194
\@endfortrue .....	25, 54, 71, 264	\@glo@label .....	66, 72, 73, 75–84, 90, 154, 256, 257, 311, 312, 340
\@esphack .....	177	\@glo@list .....	84
\@expandtwoargs .....	29	\@glo@long .....	55, 66, 80, 82
\@firstofone .....	20, 21	\@glo@longaccess .....	338–340
\@firstofthree .....	107, 120, 121, 123, 125, 139, 140, 142, 144, 354–356	\@glo@longpl .....	66, 80, 82, 232, 234, 236, 238, 241, 243, 245, 367
\@firstoftwo .....	23– 25, 70, 71, 107, 123–125, 140–142, 144, 145	\@glo@longpluralaccess .....	338–340
\@for .....	25, 29, 31, 42, 44, 50, 51, 70, 71, 112, 158, 160, 169, 170, 177, 182, 188, 219, 233, 235, 237, 239, 241, 244, 246, 248, 264, 265, 318	\@glo@name .....	11, 61, 67, 79, 81–83
\@glo@desc .....	83	\@glo@no@assign@sortkey .....	170
\@glo@\$ymbol .....	83	\@glo@nonumberlist .....	64, 65
\@glo@access .....	337, 339, 340, 342	\@glo@numfmt .....	180, 318
\@glo@addchildren .....	188, 193	\@glo@parent .....	13, 64, 79, 81, 82, 86, 189
\@glo@assign@sortkey .....	170, 172, 200	\@glo@plural .....	62, 79, 81, 82, 369
\@glo@autosee .....	84	\@glo@pluralaccess .....	338–340
\@glo@autoseehook .....	84	\@glo@prefix .....	8, 64, 79, 86, 111, 112, 180, 317, 318
\@glo@check@m\$kidxrangechar .....	111, 112, 180, 317, 318	\@glo@range .....	180, 317, 318
\@glo@check@sortallowed ..	11–13, 170, 174	\@glo@see .....	63, 79, 84
\@glo@childlist .....	188	\@glo@seeautonumberlist .....	8, 63
\@glo@counter .....	63, 79, 82	\@glo@short .....	55, 65, 80, 82, 370
		\@glo@shortaccess .....	338–340, 367–370
		\@glo@shortpl .....	66, 80, 82, 232, 234, 236, 238, 240, 243, 245, 367, 370
		\@glo@shortpluralaccess .....	338–340
		\@glo@sort .....	11, 13, 20, 62, 79, 82, 86
		\@glo@sortedinsert .....	189
		\@glo@sortentries .....	191, 192
		\@glo@sorthandler@case .....	192

\@glo@sorthandler@letter	191	\@gls@automake	170
\@glo@sorthandler@nocase	192	\@gls@between	264, 265
\@glo@sorthandler@word	191	\@gls@body	147
\@glo@sortinghandler	188, 190	\@gls@checkactual	113, 165
\@glo@sortinglist	188, 189, 192	\@gls@checkbar	113, 165
\@glo@sorttype	172, 193, 194, 200	\@gls@checkedmkidx	112–118, 164–166
\@glo@storeentry	11–13	\@gls@checkescactual	113, 165
\@glo@suffix	111, 112, 180, 318	\@gls@checkescbar	113, 165
\@glo@symbol	54, 63, 79, 83, 238, 243, 339	\@gls@checkescquote	113, 164–166
\@glo@symbolaccess	338–340, 370	\@gls@checklevel	113, 165
\@glo@symbolplural	63, 79, 83	\@gls@checkmkidxchars	.. 85, 86, 111, 164, 171, 179, 181, 317, 318
\@glo@symbolpluralaccess	338–340	\@gls@checkquote	113, 164
\@glo@text	62, 79, 82, 83, 121–125, 146–148, 238, 257, 369	\@gls@classI	158, 159
\@glo@textaccess	337, 339, 340, 367–370	\@gls@classII	158, 159
\@glo@thislabel	85	\@gls@codepage	187
\@glo@thislettergrp	194, 195	\@gls@counter	.... 106, 109–111, 155, 174, 180, 181, 318
\@glo@thisvalue	55, 56	\@gls@counterwithin	10, 198, 199, 201
\@glo@tmp	72, 73, 181	\@gls@ctr	42
\@glo@type	7, 13, 63, 79, 80, 82–84, 155, 169, 175, 185–189, 193, 194, 197, 198, 217, 233, 235, 237, 239, 241, 244, 246, 248, 263–265	\@gls@currentlettergroup	193, 195
\@glo@types	50, 51, 58, 89, 90, 155, 156, 169, 170, 254, 311	\@gls@declareoption	8, 9, 14, 15, 18, 24, 26–29
\@glo@useri	65, 79, 82	\@gls@default	95
\@glo@userii	65, 79, 82	\@gls@default@value	54–56, 67, 68, 79, 81–83, 242, 256
\@glo@useriii	65, 79, 82	\@gls@deffile	69, 70
\@glo@useriv	65, 79, 82	\@gls@defsort	11–13, 83
\@glo@userv	65, 79, 82	\@gls@defsortcount	11–13, 59
\@glo@uservi	65, 79, 82	\@gls@do@acronymsdef	14, 15, 30, 60
\@glodesc	83	\@gls@do@indexdef	29, 30, 60
\@glolist@	80	\@gls@do@numbersdef	29, 30, 60
\@gloname	83	\@gls@do@symbolsdef	28, 60
\@glossary@default@style	7, 9, 185, 208, 209, 249	\@gls@do@symbolssdef	30
\@glossaryentryfield	86	\@gls@doautomake	27, 170
\@glossarysection	38	\@gls@docheckquotedef	164–166
\@glossarystyle	185, 186, 198	\@gls@docloadedfalse	4
\@glossarysubentryfield	86	\@gls@docloadedtrue	4
\@gls	120	\@gls@dodeflistparser	170
\@gls@	91, 93, 120, 259, 260	\@gls@doentrydef	105
\@gls@@link	108	\@gls@dolast	182, 183
\@gls@Hcounter	110, 111	\@gls@donext	182, 183
\@gls@ReturnAfterFi	211	\@gls@donext@def	154
\@gls@access@display	342, 343	\@gls@dothiswrite	167, 168
\@gls@actualchar	86, 115, 117, 162, 321	\@gls@elem	264
\@gls@addpredefinedattributes	157, 166	\@gls@enablesavenonumberlist	70
\@gls@adjustmode	155	\@gls@encapchar	115, 116, 162, 180, 182, 318, 321
		\@gls@entry@count	92

<code>\@gls@entry@field</code> .....	<code>\@gls@noaccess</code> .....	342
..... 72, 73, 90, 91, 147–153, 340–342	<code>\@gls@noexpand@fields</code> .....	69
<code>\@gls@escbsdq</code> .....	<code>\@gls@nohyperlist</code> .....	17, 61, 109
113, 163, 322	<code>\@gls@noidx@do</code> .....	194
<code>\@gls@expand@fields</code> .....	<code>\@gls@noidx@getgrouptitle</code> .....	171
68, 69	<code>\@gls@noidx@sanitizesort</code> .....	20, 173
<code>\@gls@expandonce</code> .....	<code>\@gls@noidx@setsanitizesort</code> ....	22, 173
68	<code>\@gls@noidx@loclist@finalsep</code> .....	173
<code>\@gls@extramakeindexopts</code> .....	<code>\@gls@noidx@loclist@prev</code> .....	173, 196
169	<code>\@gls@noidx@loclist@sep</code> .....	173, 196
<code>\@gls@fetchfield</code> .....	<code>\@gls@noref@warn</code> .....	171, 194
56	<code>\@gls@numberlink</code> .....	211
<code>\@gls@field@link</code> .....	<code>\@gls@numbersdef</code> .....	29
74, 75, 126–138	<code>\@gls@numlist@lastsep</code> .....	154
<code>\@gls@firsttok</code> .....	<code>\@gls@numlist@nextsep</code> .....	154
194	<code>\@gls@numlist@sep</code> .....	154
<code>\@gls@fixbraces</code> .....	<code>\@gls@old@chapter</code> .....	31
84	<code>\@gls@oldnewglossaryentryposthook</code> .	339
<code>\@gls@forbidtexext</code> .....	<code>\@gls@oldnewglossaryentryprehook</code> ..	339
59	<code>\@gls@onlypremakeg</code> .....	31
<code>\@gls@get@counterprefix</code> .....	<code>\@gls@order</code> .....	167, 168
181	<code>\@gls@org@LT@output</code> .....	281
<code>\@gls@getbody</code> .....	<code>\@gls@org@glsnoidxdisplayloc</code> .....	173
147	<code>\@gls@org@glssseeformat</code> .....	173
<code>\@gls@getcounterprefix</code> .....	<code>\@gls@patchtabularx</code> .....	87
180	<code>\@gls@preglossaryhook</code> .....	186
<code>\@gls@getgrouptitle</code> .....	<code>\@gls@prevlevel</code> . 291–293, 312–315, 330, 331	
171, 207, 265	<code>\@gls@provide@newglossary</code> .....	59
<code>\@gls@glossary</code> .....	<code>\@gls@quotechar</code> .... 114–117, 162–166, 321	
176	<code>\@gls@reference</code> .....	171, 174
<code>\@gls@gobbleopt</code> .....	<code>\@gls@removespaces</code> .....	211
58	<code>\@gls@renewglossary</code> .....	167
<code>\@gls@grptitle</code> .....	<code>\@gls@replacementtext</code> .....	342
207, 263, 265	<code>\@gls@rest</code> .....	147
<code>\@gls@hyp@opt</code> .....	<code>\@gls@roman</code> .....	45, 46, 318, 319
74,	<code>\@gls@sanitized@tmp</code> .....	112
75, 93, 94, 108, 120–145, 214–216, 258–261	<code>\@gls@sanitizedesc</code> .....	25
<code>\@gls@hyp@opt@cs</code> .....	<code>\@gls@sanitizesort</code> .....	11
107	<code>\@gls@sanitizesymbol</code> .....	25, 26
<code>\@gls@hypergroup</code> .....	<code>\@gls@saveentrycounter</code> .....	110, 155
263	<code>\@gls@savenonumberlist</code> .....	64
<code>\@gls@ifinlist</code> .....	<code>\@gls@see@noindex</code> .....	6, 64
42	<code>\@gls@setacrstyle</code> .....	25, 26, 30
<code>\@gls@ifnotmeasuring</code> .....	<code>\@gls@setcounter</code> .....	59
87	<code>\@gls@setdefault@glslink@opts</code> ....	109
<code>\@gls@igtype</code> .....	<code>\@gls@setsort</code> .....	11–13, 110
61	<code>\@gls@setupshortcuts</code> .....	30
<code>\@gls@increment@currcount</code> .....	<code>\@gls@sort</code> .....	195
91	<code>\@gls@sort@A</code> .....	190, 191
<code>\@gls@indexdef</code> .....		
29		
<code>\@gls@initnonumberlist</code> .....		
64, 79		
<code>\@gls@islistofacronyms</code> .....		
16		
<code>\@gls@keylist</code> .....		
366		
<code>\@gls@keymap</code> .....		
65, 70–73, 256, 339		
<code>\@gls@label</code> .....		
171, 174, 180		
<code>\@gls@langmod</code> .....		
167, 168		
<code>\@gls@levelchar</code> .... 86, 116, 117, 162, 321		
<code>\@gls@link</code> .. 108, 121–126, 139–146, 354–356		
<code>\@gls@link@checkfirsthyper</code> .... 120–125		
<code>\@gls@link@label</code> .....		
109, 234, 240		
<code>\@gls@link@nocheckfirsthyper</code> 126, 139–145		
<code>\@gls@link@opts</code> .....		
109, 234, 240		
<code>\@gls@list</code> .....		
264, 265		
<code>\@gls@listsuffix</code> .....		
42		
<code>\@gls@loadlist</code> .....		
9, 248		
<code>\@gls@loadlong</code> .....		
8, 9, 249		
<code>\@gls@loadsuper</code> .....		
9, 249		
<code>\@gls@loadtree</code> .....		
9, 249		
<code>\@gls@local@increment@currcount</code> .... 91		
<code>\@gls@loclist</code> .....		
172, 173, 195, 196		
<code>\@gls@map</code> .....		
70, 71		
<code>\@gls@missingnumberlist</code> .....		
83		

\@gls@sort@B .....	190, 191	\@glshypernumber .....	210
\@gls@startswithexpandonce .....	68	\@glsisacronymlistfalse .....	16
\@gls@storenonumberlist .....	64, 65, 83	\@glsisacronymlisttrue .....	16
\@gls@symbolsdef .....	28	\@glslink .....	110, 120, 154, 263
\@gls@this .....	177	\@glslocalreset .....	88, 91
\@gls@thisHloc .....	181	\@glslocalunset .....	88, 91
\@gls@thisfield .....	56	\@glslocref .....	174, 179, 180, 317, 318
\@gls@thislabel .....	53, 54, 182, 183, 192	\@glsminrange .....	157–159, 319
\@gls@thislist .....	154	\@glsname .....	129
\@gls@thisloc .....	181	\@glsname@ .....	129, 130
\@gls@thisval .....	71	\@glsnextpages .....	186
\@gls@title .....	38	\@glsnodels .....	79, 80, 83
\@gls@tmp 13, 33, 47, 68, 112, 176, 177, 264, 265		\@glsnoname .....	79, 81, 83
\@gls@tmpb .....	113–118, 164–166	\@glsnonextpages .....	185
\@gls@toc .....	40	\@glsnumberformat .....	
\@gls@type .....	170, 219,	.....	106, 109, 155, 174, 180, 317, 318
233, 235, 237, 239, 241, 244, 246, 248, 311		\@glsopenfile .....	167, 175
\@gls@updatechecked ....	112, 113, 164, 165	\@glsorder .....	169
\@gls@usetranslator .....	24, 33	\@glspl .....	123
\@gls@value .....	67, 68, 150	\@glspl@ .....	91, 94, 123, 259–261
\@gls@warnonglossdefined .....	18, 184	\@glsplural .....	128
\@gls@warnonthehglossdefined ....	18, 203	\@glsplural@ .....	128
\@gls@write@entrycounts .....	92	\@glsreset .....	87, 91
\@gls@writedef .....	70	\@glssee .....	84, 182
\@gls@writeisthook .....	161, 163	\@glssymbol .....	132
\@gls@xdy@locationlist .....	158	\@glssymbol@ .....	132
\@gls@xdycheckbackslash .....	112	\@glssymbolplural .....	133
\@gls@xdycheckquote .....	112	\@glssymbolplural@ .....	133
\@gls@xref .....	181, 182	\@glstarget .....	120, 203, 263
\@gls@Alphacompositor .....	36, 46, 319	\@glstext .....	126
\@gls@Hlocref .....	179, 180	\@glstext@ .....	126
\@gls@acronymlists ..	15, 16, 51, 217, 219,	\@glsunset .....	88, 91
233, 235, 237, 239, 241, 244, 246, 248, 253		\@glsuseri .....	134
\@gls@addkey .....	73	\@glsuseri@ .....	134
\@gls@addstoragekey .....	72	\@glsuserii .....	134
\@gls@addxdyattribute .....	43, 44	\@glsuseriii .....	134
\@gls@defaultsort .....	11	\@glsuseriii .....	135
\@gls@desc .....	130	\@glsuseriii@ .....	135
\@gls@desc@ .....	130	\@glsuseriv .....	136
\@gls@descplural .....	131	\@glsuseriv@ .....	136
\@gls@descplural@ .....	131	\@glsuserv .....	137
\@gls@disp .....	125	\@glsuserv@ .....	137
\@gls@entry .....	89, 90, 92	\@glsuservi .....	138
\@gls@entrytitlecase .....	150	\@glsuservi@ .....	138
\@gls@first .....	127	\@gls@widestname .....	312, 313, 330
\@gls@first@ .....	127	\@gls@writefiles .....	28
\@gls@firstletter .....	50, 157	\@gls@xtr@doaccsupp .....	336
\@gls@firstplural .....	129	\@gobble .....	11–13,
\@gls@firstplural@ .....	129	70, 87, 112, 156, 157, 160, 171, 316, 320, 321	

\@idxitem .....	306	\@print@glossary .....	184
\@ifclassloaded .....	4, 10, 39	\@print@noidx@glossary .....	184
\@ifnextchar .....	59, 107	\@printgloss@setsort .....	170, 172, 185
\@ifpackageloaded .....		\@printglossary .....	184
..... 4, 7, 23–25, 33, 50, 87, 153, 163, 336		\@roman .....	45, 318
\@ifstar .....	58, 72, 73, 107, 213	\@secondofthree .....	
\@ifundefined .....		107, 120, 121, 124, 139, 141, 143, 145, 354	
. 33, 264, 271, 282, 293, 300, 313, 330, 344		\@secondoftwo . 21, 23, 24, 33, 70, 71, 119–	
\@ignored@glossaries .....	60, 61	122, 125, 139, 140, 142, 143, 354–356, 374	
\@input@ .....	186	\@set@glo@numformat .....	180, 318
\@istfilename .....	169	\@sglsaddkey .....	73
\@makecol .....	281	\@sglsaddstoragekey .....	72
\@makeglossary .....	169	\@thirdofthree .....	
\@minus .....	268, 287, 306	.... 107, 122, 125, 140, 142, 143, 145, 354	
\@mkboth .....	39	\@this@attr .....	160
\@newglossary .....	57, 59	\@this@childlabel .....	188, 189
\@newglossaryentry@defcounters ..	84, 90	\@this@counter .....	44
\@newglossaryentryposthook .....		\@this@ctr .....	160
..... 72, 73, 84, 256, 339		\@this@key .....	71
\@newglossaryentryprehook .....		\@this@label .....	188
..... 72, 73, 78, 80, 256, 339		\@thiscs .....	31
\@nil .....	16, 84, 111–113, 147, 164,	\@tmp .....	45, 319
165, 180, 182, 194, 195, 210, 211, 317, 318		\@use@option .....	29
\@nnil .....	16, 182	\@warn@nomakeglossaries .....	168, 188
\@no@makeglossaries .....	170, 172	\@wrglossary@pageformat .....	178
\@no@post@desc .....	323	\@wrglossarynumberhook .....	178, 179
\@nopostdesc .....	186	\@xdy@main@language .....	27, 167, 187
\@onelevel@sanitize .....	20,	\@xdy@attributelist .....	43, 160
45, 70, 86, 112, 161, 181, 183, 194, 319, 320		\@xdy@attributes .....	43, 158, 316, 318
\@onlypreamble ... 59, 69, 78, 92, 95, 170, 174		\@xdy@counters .....	42, 44, 160
\@onlypremakeg .... 35–37, 43, 44, 47, 59, 163		\@xdy@crossrefhook .....	159
\@org@glossaryentrynumbers .... 185, 186		\@xdy@language .....	187
\@org@gls@assign@descplural .....		\@xdy@lettergroups .....	50, 161, 321
..... 232, 241, 243–246, 367, 370, 371		\@xdy@locationclassorder .... 48, 159, 320	
\@org@gls@assign@firstpl .....	232,	\@xdy@locref .....	43, 161, 316, 320
234–236, 238, 239, 241, 243–246, 367–371		\@xdy@requiredstyles .....	48, 49, 158, 318
\@org@gls@assign@plural .....	232,	\@xdy@sortrules .....	48, 161, 321
234–236, 238, 239, 241, 243–246, 367–371		\@xdystyle .....	158, 318
\@org@gls@assign@symbolplural .. 232,		\@xdy@useralphabets .....	45, 158, 318
234–236, 238, 239, 244–246, 368, 369, 371		\@xdy@userlocationdefs ... 47, 159, 317, 319	
\@org@gls@numberformat .....	154	\@xdy@userlocationnames .....	47, 48, 317
\@org@newglossaryentryprehook .....	78	\@xfor@nextelement .....	182
\@outputpage .....	281	\\ .....	85, 112, 156, 162,
\@p@glossarysection .....	38	163, 210, 211, 321, 322, 324–326, 334, 335	
\@pgls .....	258	\{ .....	70, 156, 162, 163, 316, 321, 322
\@pgls@ .....	258	\} .....	70, 156, 162, 163, 316, 322
\@pglspl .....	259	\^ .....	20
\@pglspl@ .....	259	\‘ .....	20
\@plus .....	268, 287, 306	\  .....	113, 115, 165

\~ .....	21	\advance .....	12, 13, 81, 110, 281
<b>A</b>		\AE .....	21
\a .....	21	\ae .....	21
\AA .....	21	amsgen package .....	4, 106
\aa .....	21	amsmath package .....	87
accsupp package .....	336	\andname .....	183
\accsuppglossaryentryfield .....	336	\AnyTrackedLanguages .....	34, 374
\accsuppglossarysubentryfield .....	337	\appto .....	17, 65, 72, 73, 256, 339
\acrfootnote .....	234, 240	array package .....	277, 282, 300
\Acrfull .....	231	article class .....	181
\acrfull .....	231	\AtBeginDocument ....	15, 49, 70, 87, 155, 171
\ACRfullfmt ....	215, 218, 226, 228, 362, 364	\AtEndDocument .	28, 70, 92, 171, 175, 187, 264
\Acrfullfmt ....	215, 218, 226, 228, 362, 364	<b>B</b>	
\acrfullfmt ....	214, 218, 226, 228, 362, 364	\b .....	21
\acrfullformat .....	153, 214, 232, 247	babel package .....	23, 31, 33, 49
\Acrfullpl .....	231	\begin .....	160, 193, 268, 271–274, 276, 277, 279, 280, 282–305, 320
\acrfullpl .....	231	\BeginAccSupp .....	342
\ACRfullplfmt ...	216, 218, 226, 228, 362, 364	\begingroup .....	5, 176, 179, 211
\Acrfullplfmt ...	216, 218, 226, 228, 362, 364	\bfseries .....	272– 276, 278, 279, 283–287, 295–299, 301–305
\acrfullplfmt ...	216, 218, 226, 228, 362, 364	\bgroup .....	20, 78, 153, 185, 188
\acrlinkfootnote .....	233	booktabs package .....	277–280
\acrlinkfullformat .....	214–216	\boolean .....	247
\Acrlong .....	231	\boolfalse .....	28
\acrlong .....	230	\booltrue .....	28
\Acrlongpl .....	231	\bottomrule .....	278, 279
\acrlongpl .....	230	\box .....	281
\acrnameformat .....	238, 369	<b>C</b>	
\acronymentry .....		\c .....	21
	218, 220–225, 227–230, 358–360, 363–366	\c@equation .....	110
\acronymfont .....	102, 139–142, 147, 153, 217, 218, 220–230, 234, 235, 237, 239, 240, 242–244, 351, 352, 354–356, 358–360, 362–366, 368–370	\c@glossarysubentry .....	199
\acronymname .....	14, 15, 34	\c@page .....	177–179
\acronymsort .....	218, 220–225, 227–230, 358–360, 363, 364, 366	\cGls .....	93
\acronymtype .....	14, 15, 217–219, 232–241, 243–246, 248, 367–370	\cglS .....	93
\acrpluralsuffix .....		\cGlsformat .....	91
	218, 220–223, 227–229, 232–236, 238– 245, 247, 248, 358, 359, 363–365, 367–371	\cglSformat .....	91
\Acrshort .....	230	\cGlspl .....	94
\acrshort .....	230	\cglSpl .....	94
\Acrshortpl .....	230	\cGlsplformat .....	92
\acrshortpl .....	230	\cglSplformat .....	91
\addcontentsline .....	41	\char .....	207
\addglossarytocaptions .....	33	classicthesis package .....	7
\addtolength .....	313, 330	\cleardoublepage .....	41
		\clearpage .....	40, 41
		\closeout .....	70, 161, 163, 167, 175
		\compatglossarystyle .....	323–335
		\compatibleglossentry .....	205



<code>\compatiblesubglossentry</code> .....	205	<code>\def@gl@xdycheckbackslash</code> ....	118, 119
<code>\copy</code> .....	281	<code>\DefaultNewAcronymDef</code> .....	233
<code>\count@</code> .....	194	<code>\defgl@entryfmt</code> .....	59, 61, 105, 219, 232, 233, 236, 237, 240, 242, 245, 247
<code>\csdef</code> .....	19, 72–74, 84, 85, 90, 91, 188, 189, 209, 219, 220, 322	<code>\define@boolkey</code> .....	..... 5, 6, 8–10, 14, 21, 22, 25–28, 106, 200
<code>\csedef</code> .....	92, 178	<code>\define@choicekey</code> .....	..... 6, 7, 10, 22, 24, 26, 64, 198, 199
<code>\csgdef</code> .....	38, 57, 60, 91, 92, 183, 197	<code>\define@key</code> .....	7, 10, 17, 23, 27, 61–66, 72, 73, 106, 155, 197, 198, 200, 256, 337, 338
<code>\cslet</code> .....	65, 78, 85, 192	<code>\DefineAcronymSynonyms</code> .....	30, 231
<code>\csname</code> .....	10–13, 29, 32–34, 40, 43, 45, 46, 49, 51, 53, 58–60, 67, 72– 77, 80–86, 88, 89, 105, 109–111, 121– 125, 139–146, 154, 155, 158–160, 164– 167, 171, 174–178, 180–182, 185–187, 189, 198, 204, 205, 208, 209, 213, 249– 257, 264, 265, 312, 313, 316–318, 330, 336, 337, 340, 341, 344, 354–356, 372, 373	<code>\delimN</code> .....	160, 170, 196, 211, 320
<code>\csshow</code> .....	253	<code>\delimR</code> .....	160, 210, 211, 320
<code>\csuse</code> .....	34, 38, 57, 67, 68, 74, 105, 168, 189, 191, 193, 195, 197–199, 209, 220, 257, 323–335	<code>\DescriptionDUANewAcronymDef</code> .....	237
<code>\csxdef</code> .....	83, 92	<code>\DescriptionFootnoteNewAcronymDef</code> .	235
<code>\currentglossary</code> .....	38, 185, 199, 201	<code>\descriptionname</code> .....	34, 272– 276, 278, 279, 283–287, 295–299, 301–305
<code>\currentglssubentry</code> .....	199, 201, 202	<code>\DescriptionNewAcronymDef</code> .....	239
<code>\CurrentOption</code> .....	29, 256, 336	<code>\DH</code> .....	21
<code>\CurrentTrackedLanguage</code> .....	34, 374, 375	<code>\dh</code> .....	21
<code>\CurrentTrackedTag</code> .....	34, 374, 375	<code>\dimen@</code> .....	221, 281, 311
<code>\CustomAcronymFields</code> .....	248	<code>\disable@keys</code> .....	30
<code>\CustomNewAcronymDef</code> .....	248	<code>\do</code> .....	25, 29, 31, 42, 44, 50, 51, 70, 71, 112, 154, 158, 160, 169, 170, 177, 182, 188, 219, 233, 235, 237, 239, 241, 244, 246, 248, 264, 265, 318
<b>D</b>		<code>\do@glo@storeentry</code> .....	11–13, 84
<code>\d</code> .....	21	<code>\do@gl@link@checkfirsthyper</code> .....	..... 108, 109, 120–126, 139–145, 354, 355
<code>datatool package</code> .....	190	<code>\do@gl@xdycheckbackslash</code> .....	112
<code>\day</code> .....	158, 162, 318, 321	<code>\do@gl@disablehyperinlist</code> .....	109
<code>\DeclareAcronymList</code> ....	14, 15, 17, 217, 219, 233, 235, 237, 239, 241, 244, 246, 248	<code>\do@gl@shaschildren</code> .....	53, 54
<code>\DeclareListParser</code> .....	170	<code>doc package</code> .....	4, 5, 13, 14
<code>\DeclareOption</code> .....	8, 256, 336	<code>\doifglossarynoexistsordo</code> .....	58
<code>\DeclareOptionX</code> .....	8	<code>\dtl@ifsingle</code> .....	207
<code>\DeclareRobustCommand</code> .....	..... 35, 182, 183, 242, 342–344	<code>\dtl@insertinto</code> .....	190
<code>\def</code> .....	8, 11–13, 16, 20, 21, 26, 27, 30– 32, 34, 35, 38, 42, 45–50, 53, 54, 57–66, 68, 76, 78–83, 85, 87, 91–95, 105, 106, 109–146, 154, 155, 157, 162–169, 172, 173, 175, 178–182, 185, 188, 192–194, 196–201, 207–211, 213–217, 232–239, 241, 243–246, 248, 256, 258–262, 264– 267, 291–293, 312–315, 317, 318, 321, 323, 330, 331, 336–339, 353–356, 367–371	<code>\dtl@sortresult</code> .....	190, 191
		<code>\dtlcompare</code> .....	190
		<code>\dtlicompare</code> .....	191
		<code>\DTLifinlist</code> .....	61, 109
		<code>\DTLifint</code> .....	207
		<code>\dtlletterindexcompare</code> .....	190
		<code>\DTLsubstituteall</code> .....	112
		<code>\dtlwordindexcompare</code> .....	190
		<code>\DUANewAcronymDef</code> .....	246
<b>E</b>		<code>\eappto</code> .....	60, 61, 85, 178
		<code>\edef</code> .....	13, 16, 31, 34, 42, 43, 45–49, 51, 53, 58–61, 67,



68, 71, 75–77, 79, 80, 85, 86, 105, 109–118, 154–157, 162–168, 170, 171, 175, 180, 181, 183, 187–191, 194, 199, 202, 207, 211, 213, 232, 234, 236, 238, 240, 243, 245, 263, 316, 317, 319, 321, 366–370	\expandonce ..... 67, 68, 112, 164–166, 178, 190, 191, 204, 205, 218, 232, 234, 236, 238, 240, 241, 243, 245, 336, 337
<b>F</b>	
\egroup ..... 20, 78, 154, 186, 189	\fi ..... 5–7, 9, 11, 13–16, 18, 19, 21, 22, 24, 27–31, 35, 36, 39, 41–51, 60, 64, 67, 80–83, 85–87, 91, 92, 108–119, 121–126, 148, 155, 157, 158, 161, 163–168, 170, 175–183, 185, 187, 194, 198–203, 208–211, 221, 231, 233, 235–237, 239, 241, 242, 244–249, 255, 264, 268, 272, 273, 275, 278–281, 283, 284, 286, 294, 296, 297, 301, 303, 304, 308–314, 316–320, 322, 323, 328–331, 336, 342
\else . 5, 9, 13–16, 18, 19, 21, 22, 27–31, 35, 36, 39, 41–45, 47–51, 64, 66, 81, 82, 85–87, 91, 92, 108–110, 112–119, 121–126, 147, 157, 158, 161, 163–168, 175–182, 185, 194, 198–203, 208, 210, 211, 221, 235, 237, 239, 242, 244–247, 249, 264, 268, 272, 273, 275, 278, 279, 281, 283, 284, 286, 294, 296, 297, 301, 303, 304, 307–309, 311–314, 317–323, 328–331, 342	file types
\emph ..... 182, 212	.aux ..... 187
\empty ..... 211, 336	.glo ..... 85
\end ..... 160, 194, 268, 271–274, 276, 277, 279, 280, 282–306, 320	.ist ..... 156, 166
\end@doifinlist ..... 42	.toc ..... 41
\end@getprefix ..... 181	.xdy ..... 36
\end@glis@islistofacronyms ..... 16	glo ..... 254
\EndAccSupp ..... 342	\firstacronymfont 104, 220–222, 227, 228, 234, 238, 240, 243, 353, 357–359, 363, 364
\endcsname ..... 10–13, 29, 32–34, 40, 43, 45, 46, 49, 51, 53, 58–60, 67, 72–77, 80–86, 88, 89, 105, 109–111, 121–125, 139–146, 154, 155, 158–160, 164–167, 171, 174–178, 180–182, 185–187, 189, 198, 204, 205, 208, 209, 213, 249–257, 264, 265, 312, 313, 316–318, 330, 336, 337, 340, 341, 344, 354–356, 372, 373	\footnote ..... 227, 228, 233, 364
\endfoot . 272–274, 276, 278, 279, 283–285, 287	\FootnoteNewAcronymDef ..... 241
\endgroup ..... 5, 177, 179, 211	\forall glossaries ..... 51, 175, 184, 311
\endhead ..... 272–276, 278, 279, 283–287	\forall glsentries ..... 89, 90, 92, 155, 156
\endtheglossary ..... 5	\ForEachTrackedDialect ..... 34, 374, 375
\entryname ..... 34, 272–276, 278, 279, 283–287, 295–299, 301–305	\for glsentries ..... 51, 53, 85, 192, 311
\equal ..... 23, 30, 40, 110, 169, 208, 264	\forlistcsloop ..... 188, 194
equation (counter) ..... 110, 111	\forlistloop ..... 173, 196
etoolbox package ..... 4	
\expandafter ..... 11–13, 20, 29, 31, 33, 34, 43, 45, 46, 48–51, 53, 58–61, 67, 68, 70–77, 80–84, 86–89, 105, 109–118, 147, 154, 156, 157, 160, 164, 165, 167, 175–177, 179, 180, 183, 185, 189, 194, 195, 204, 205, 209, 211, 213, 234, 240, 249–254, 256, 257, 264, 265, 312, 316–318, 320, 321, 336, 337, 340, 342, 366, 372, 373	<b>G</b>
	garamondx package ..... 213
	\gdef ..... 12, 43, 58, 76, 81, 82, 176, 200, 264
	\Genacrfullformat ..... 103, 218, 220–222, 227, 353, 357, 358, 364
	\genacrfullformat ..... 103, 104, 218, 220, 221, 227, 352, 353, 357, 358, 363
	\GenericAcronymFields . . 218, 220, 221, 223–227, 229, 230, 357–360, 362, 363, 366
	\Genplacrfullformat ..... 103, 218, 220–222, 228, 352, 358, 364
	\genplacrfullformat ..... 103, 104, 218, 220–222, 228, 352, 358, 364
	\glo@desc ..... 323
	\glo@do@compare ..... 190, 191
	\glo@grabfirst ..... 195
	\glo@label ..... 53, 54, 85

<code>\glo@list</code> .....	85	<code>index</code> .....	7, 288, 306–308, 328
<code>\glo@name</code> .....	204	<code>indexgroup</code> .....	308, 328
<code>\glo@parent</code> .....	54	<code>indexhypergroup</code> .....	308, 328
<code>\glo@type</code> .....	85	<code>inline</code> .....	323
<code>\glo@value</code> .....	70	<code>list</code> .....	7, 268–270, 323
<code>\global</code> .....	12, 13, 67, 69, 78, 83, 88, 89, 176, 186, 195, 201, 281	<code>listdotted</code> .....	270, 271, 324
<code>\glo@linkprefix</code> .....	110, 154, 203	<code>listgroup</code> .....	269, 323
<code>\glosortentrieswarning</code> .....	18, 188	<code>listhypergroup</code> .....	269, 324
<code>glossareentry (counter)</code> .....	201	<code>long</code> .....	271–273, 278, 282, 324, 326
<code>glossaries package</code> .....	29, 49, 157, 248, 256, 268, 316, 336	<code>long-booktabs</code> .....	277, 280
<code>glossaries-accsupp package</code> .....	85, 336	<code>long3col</code> .....	273, 274, 278, 325
<code>glossaries-extra package</code> .....	159, 336	<code>long3col-booktabs</code> .....	278, 280
<code>\GlossariesWarning</code> .....	5, 6, 18, 21–23, 38, 41, 53, 56, 64, 66, 93, 94, 105, 107, 153, 168, 171– 174, 177, 181, 185, 205, 206, 208, 316, 336	<code>long3colborder</code> .....	274, 325
<code>\GlossariesWarningNoLine</code> .....	5, 18, 169, 170, 172, 175, 188, 264	<code>long3colheader</code> .....	274, 278, 325
<code>\glossary</code> .....	317, 318	<code>long3colheaderborder</code> .....	274, 325
<code>glossary package</code> .....	1, 212	<code>long4col</code> .....	274–276, 279, 325
<code>glossary styles:</code>		<code>long4col-booktabs</code> .....	278, 279
<code>altlist</code> .....	269, 270, 324	<code>long4colborder</code> .....	275, 326
<code>altlistgroup</code> .....	270, 324	<code>long4colheader</code> .....	275, 278, 326
<code>altlisthypergroup</code> .....	270, 324	<code>long4colheaderborder</code> .....	276, 326
<code>altlong4col</code> .....	276, 277, 285, 326	<code>longborder</code> .....	272, 325
<code>altlong4col-booktabs</code> .....	279, 280	<code>longheader</code> .....	272, 277, 278, 325
<code>altlong4colborder</code> .....	277, 326	<code>longheaderborder</code> .....	273, 325
<code>altlong4colheader</code> .....	276, 279, 326	<code>longragged</code> .....	279, 282–284
<code>altlong4colheaderborder</code> ....	277, 326	<code>longragged-booktabs</code> .....	279
<code>altlongragged4col</code> ...	280, 285, 286, 327	<code>longragged3col</code> .....	280, 284, 285, 327
<code>altlongragged4col-booktabs</code> ....	280	<code>longragged3col-booktabs</code> .....	280
<code>altlongragged4colborder</code> ....	286, 327	<code>longragged3colborder</code> .....	284, 327
<code>altlongragged4colheader</code> ....	286, 327	<code>longragged3colheader</code> .....	285, 327
<code>altlongragged4colheaderborder</code>	287, 328	<code>longragged3colheaderborder</code> .	285, 327
<code>altsuper4col</code> .....	298, 299, 304, 335	<code>longraggedborder</code> .....	283, 326
<code>altsuper4colborder</code> .....	299, 335	<code>longraggedheader</code> .....	283, 327
<code>altsuper4colheader</code> .....	299, 335	<code>longraggedheaderborder</code> ....	283, 327
<code>altsuper4colheaderborder</code> ...	299, 335	<code>mcolalttree</code> .....	292, 332
<code>altsuperragged4col</code> .....	304, 305, 333	<code>mcolalttreegroup</code> .....	292, 332
<code>altsuperragged4colborder</code> ...	305, 333	<code>mcolalttreehypergroup</code> .....	292, 332
<code>altsuperragged4colheader</code> ...	305, 333	<code>mcolindex</code> .....	288, 331
<code>altsuperragged4colheaderborder</code> .	305, 333	<code>mcolindexgroup</code> .....	288, 331
<code>alttree</code> .....	291, 306, 307, 312, 314, 330	<code>mcolindexhypergroup</code> .....	288, 331
<code>alttreegroup</code> .....	314, 331	<code>mcoltree</code> .....	289, 331
<code>alttreehypergroup</code> .....	314, 331	<code>mcoltreegroup</code> .....	331
		<code>mcoltreehypergroup</code> .....	289, 290, 331
		<code>mcoltreenoname</code> .....	290, 332
		<code>mcoltreenonamegroup</code> .....	291, 332
		<code>mcoltreenonamehypergroup</code> ...	291, 332
		<code>sublistdotted</code> .....	324
		<code>super</code> .....	293–295, 301, 334
		<code>super3col</code> .....	295, 296, 334

super3colborder .....	296, 334	\glossarytitle .....	159, 185, 193, 197, 320
super3colheader .....	296, 334	\glossarytoctitle .....	7, 14, 15, 28,
super3colheaderborder .....	296, 334		29, 32, 34, 39, 159, 185, 193, 197, 198, 320
super4col .....	297, 298, 335	\glossentry .....	85, 186, 195, 196, 205,
super4colborder .....	298, 335		266, 268–273, 275, 282, 284, 286, 294,
super4colheader .....	297, 335		295, 297, 301, 302, 304, 307, 309, 310, 312
super4colheaderborder .....	298, 335	\Glossentrydesc .....	356
superborder .....	294, 334	\glossentrydesc .....	266, 268–
superheader .....	294, 334		270, 272, 273, 275, 282–284, 286, 294–
superheaderborder .....	295, 334		297, 301, 302, 304, 307–310, 313, 314, 356
superragged .....	300, 302, 332	\glossentryname .....	266,
superragged3col .....	302, 303, 333		268–273, 275, 282, 284, 286, 294, 295,
superragged3colborder .....	303, 333		297, 301, 302, 304, 307–310, 312, 314, 356
superragged3colheader .....	303, 333	\Glossentrysymbol .....	357
superragged3colheaderborder .....	303, 333	\glossentrysymbol .....	266,
superraggedborder .....	301, 332		275, 286, 297, 304, 307–310, 313, 314, 357
superraggedheader .....	301, 332	\Gls .....	93, 213, 231
superraggedheaderborder .....	302, 333	\gls .....	93, 171, 202, 213, 231
tree .....	289, 308–310, 312, 328	\gls@Alphpage .....	177, 179
treegroup .....	289, 309, 329	\gls@alphpage .....	177, 179
treehypergroup .....	309, 329	\gls@arabicpage .....	177, 179
treenoname .....	290, 307, 310, 311, 329	\gls@assign@desc .....	78, 83
treenonamegroup .....	311, 330	\gls@assign@descplural .....	
treenonamehypergroup .....	311, 330		232, 241, 243–246, 367, 370, 371
glossary-hypernav package .....	156	\gls@assign@field .....	
glossary-list package .....	7, 9, 267		69, 72, 73, 78, 80, 82, 83, 256, 257
glossary-long package .....	8, 271, 285, 293	\gls@assign@firstpl .....	232,
glossary-longragged package .....	282		234–236, 238, 239, 241, 243–246, 367–371
glossary-mcols package .....	287	\gls@assign@plural .....	232,
glossary-super package ...	9, 271, 293, 300, 304		234–236, 238, 239, 241, 243–246, 367–371
glossary-superragged package .....	300	\gls@assign@symbolplural .....	232,
glossary-tree package .....	9, 306		234–236, 238, 239, 244–246, 368, 369, 371
\glossaryentry .....	180, 182, 318	\gls@checkisacronymlist .....	108
glossaryentry (counter) .....	10, 202, 203	\gls@checkseeallowed .....	63, 69, 169, 171
\glossaryentryfield .....		\gls@checkseeallowed@preambleonly ..	69
	204, 323–330, 332–335, 357	\gls@codepage .....	49, 50, 168, 187
\glossaryentrynumbers .....		\gls@defdocnewglossaryentry .....	70, 90
	8, 160, 185, 186, 195, 196, 200, 201, 320	\gls@defglossaryentry .....	69, 70, 78
\glossaryheader ....	160, 193, 266, 268–	\gls@disablepagerefexpansion ..	177, 179
	276, 278, 279, 282–288, 290–292, 294,	\gls@do@addxdyattribute .....	44
	295, 297, 300, 302, 304, 307–312, 314, 320	\gls@doclearpage .....	41
\glossarymark .....	39	\gls@dosubst .....	112
\glossaryname .....	14, 34	\gls@dotocitle .....	185, 186, 197
\glossarypostamble .....	160, 194, 320	\gls@end@sanitizesort .....	20
\glossarypreamble .....	159, 193, 320	\gls@endcheck .....	68
\glossarysection .....	159, 193, 320	\gls@glossary .....	176, 180–182
glossarysubentry (counter) ....	10, 201–203	\gls@gobbleopt .....	59
\glossarysubentryfield .....		\gls@grplabel .....	263
	205, 323–330, 332–335, 357	\gls@hypergroup prerun .....	264

<code>\gls@ifnotmeasuring</code> .....	87, 88	<code>\glsautomakefalse</code> .....	27
<code>\gls@inlinepostchild</code> .....	265–267, 323	<code>\glsautoprefix</code> .....	7, 198
<code>\gls@inlinessep</code> .....	265, 266, 323	<code>\glscapscase</code> .....	96, 97, 100–103, 121–125, 139–145, 225, 226, 238, 243, 344, 346, 348, 349, 351, 352, 354–356, 361
<code>\gls@inlinesubsep</code> .....	265, 266, 323	<code>\glsclearpage</code> .....	40
<code>\gls@islistofacronyms</code> .....	16	<code>\glsclosebrace</code> .....	47, 160, 161, 320, 321
<code>\gls@istfilebase</code> .....	35, 168	<code>\glscompositor</code> .....	36, 46, 163, 319, 322
<code>\gls@label</code> .....	213	<code>\glscounter</code> .....	17, 30, 42, 59, 82, 110, 316
<code>\gls@level</code> .....	81, 82, 195	<code>\glscurrententrylabel</code> .....	183, 186
<code>\gls@noidxglossary</code> .....	171	<code>\glscurrentfieldvalue</code> .....	56
<code>\gls@nosetquote</code> .....	79, 162, 163, 166	<code>\glscustomtext</code> .....	96, 99, 101, 102, 104, 121–125, 139–146, 225, 226, 233, 234, 237, 238, 240, 242, 243, 344, 347, 348, 350, 351, 353–356, 361, 362
<code>\gls@numberpage</code> .....	177, 179	<code>\GlsDeclareNoHyperList</code> .....	17
<code>\gls@org@glossaryentryfield</code> .....	186	<code>\glsdefaulttype</code> .....	14, 38, 49, 51, 57, 58, 80, 95, 105, 175, 184, 185
<code>\gls@org@glossarysubentryfield</code> ....	186	<code>\glsdefmain</code> .....	14, 60
<code>\gls@org@insert</code> .....	237, 240, 242	<code>\glsdescriptionaccessdisplay</code> .....	346–348, 356, 357
<code>\gls@protected@pagefmts</code> ....	112, 177, 178	<code>\glsdescriptionpluralaccessdisplay</code> .....	344, 345
<code>\gls@Romanpage</code> .....	177, 179	<code>\glsdescwidth</code> .....	271–274, 276, 277, 279, 280, 282–287, 293–297, 299–305
<code>\gls@romanpage</code> .....	177, 179	<code>\glsdetoklabel</code> ....	52–56, 65, 70, 75–79, 85, 88–92, 109, 146, 147, 154, 155, 171– 173, 180, 186, 189–191, 195, 197, 199, 202, 204, 249–253, 312, 336, 337, 372, 373
<code>\gls@save@numberlist</code> .....	8	<code>\glsdisplay</code> .....	96, 105
<code>\gls@set@xr@key</code> .....	63	<code>\glsdisplayfirst</code> .....	96, 105
<code>\gls@suffixF</code> .....	37, 161, 163, 320, 322	<code>\glsdisplaynumberlist</code> .....	172
<code>\gls@suffixFF</code> .....	37, 161, 163, 320, 322	<code>\glsdohyperlink</code> .....	119, 120
<code>\gls@text</code> .....	104	<code>\glsdohypertarget</code> .....	119, 120
<code>\gls@thissty</code> .....	25	<code>\glsdoifexists</code> .....	53–55, 75–77, 87, 88, 120–126, 138– 145, 153, 155, 172, 173, 258–262, 353–357
<code>\gls@tmp</code> .....	175, 242	<code>\glsdoifexistsordo</code> .....	108, 146
<code>\gls@tmplen</code> ....	119, 311, 313, 314, 330, 331	<code>\glsdoifexistsorwarn</code> .....	197, 204, 205
<code>\gls@tr@set@acronym@toctitle</code> ....	14, 15	<code>\glsdoifnoexists</code> .....	69, 78
<code>\gls@tr@set@main@toctitle</code> .....	14	<code>\glsdonohyperlink</code> .....	110, 119, 120
<code>\gls@tr@set@numbers@toctitle</code> .....	29	<code>\glsdosanitizesort</code> .....	11
<code>\gls@tr@set@symbols@toctitle</code> .....	28	<code>\glsentryaccess</code> .....	342
<code>\gls@wrglossary</code> .....	176	<code>\glsentrycounter</code> .....	208, 211
<code>\gls@xdystring</code> .....	112	<code>\glsentrycounterfalse</code> .....	10
<code>\gls@xindy@glsnumbersfalse</code> .....	27	<code>\glsentrycounterlabel</code> .....	199, 203
<code>\gls@xindy@glsnumberstrue</code> .....	26	<code>\glsentrycountertrue</code> .....	10
<code>\gls@xr@key</code> .....	6, 63, 64	<code>\glsentrycurrcount</code> .....	90, 92
<code>\glsaccsupp</code> .....	342	<code>\Glsentrydesc</code> .....	131, 204, 356
<code>\glsacronymtrue</code> .....	15		
<code>\glsacrpluralsuffix</code> .....	32, 213, 222, 223, 227–229, 233		
<code>\glsacrshortcutsfalse</code> .....	30		
<code>\glsacrshortcutstrue</code> .....	30		
<code>\glsacspace</code> .....	221, 223		
<code>\glsadd</code> .....	156		
<code>\glsadd options</code>			
counter .....	155		
format .....	155, 210		
<code>\glsaddall options</code>			
types .....	155		
<code>\GlsAddXdyAttribute</code> .....	43, 44, 316, 317		
<code>\GlsAddXdyCounters</code> .....	43, 44, 60		

<code>\glentrydesc</code> .....	<code>\glentryprevcount</code> ..... 90–92
..... 98, 99, 130, 131, 204, 346–348, 356	<code>\Glentryshort</code> ..... 102,
<code>\glentrydescaccess</code> ..... 343	139, 147, 222, 228, 352–354, 358, 364, 365
<code>\Glentrydescplural</code> ..... 131	<code>\glentryshort</code> 102, 104, 139, 140, 147, 153,
<code>\glentrydescplural</code> 96, 97, 131, 132, 344, 345	218, 220–230, 351–354, 357–360, 362–366
<code>\glentrydescpluralaccess</code> ..... 343	<code>\glentryshortaccess</code> ..... 343
<code>\Glentryfirst</code> .... 94, 98, 101, 127, 347, 350	<code>\Glentryshortpl</code> .....
<code>\glentryfirst</code> .....	.... 102, 141, 222, 228, 351, 358, 364, 365
... 93, 98, 99, 101, 127, 128, 346, 347, 350	<code>\glentryshortpl</code> .....
<code>\glentryfirstaccess</code> ..... 343	..... 102, 104, 140, 142, 153, 220–
<code>\Glentryfirstplural</code> .....	222, 226–228, 247, 351, 353, 358, 362–365
..... 95, 97, 100, 129, 345, 349	<code>\glentryshortpluralaccess</code> ..... 343
<code>\glentryfirstplural</code> .....	<code>\Glentrysymbol</code> ..... 132, 205, 357
... 94, 96, 97, 100, 129, 344, 345, 348, 349	<code>\glentrysymbol</code> ..... 98, 99,
<code>\glentryfirstpluralaccess</code> ..... 343	132, 133, 205, 234, 238, 243, 346–348, 357
<code>\glentryfmt</code> ..... 59, 61	<code>\glentrysymbolaccess</code> ..... 343
<code>\Glentryfull</code> ..... 218, 227, 228, 363, 365	<code>\Glentrysymbolplural</code> ..... 133
<code>\glentryfull</code> ..... 218, 227, 228, 362, 365	<code>\glentrysymbolplural</code> .....
<code>\Glentryfullpl</code> .... 218, 227, 228, 363, 365	..... 96, 97, 133, 234, 238, 243, 344, 345
<code>\glentryfullpl</code> .... 218, 227, 228, 363, 365	<code>\glentrysymbolpluralaccess</code> ..... 343
<code>\glentryitem</code> ... 199, 266, 268–273, 275,	<code>\Glentrytext</code> ..... 98, 101, 127, 346, 350
282, 284, 286, 294, 295, 297, 301, 302,	<code>\glentrytext</code> ..... 98, 99,
304, 307, 309, 310, 312, 323–330, 332–335	101, 126, 127, 154, 183, 346, 347, 349, 350
<code>\Glentrylong</code> ..... 94, 143, 147, 153,	<code>\glentrytextaccess</code> ..... 342
220, 221, 226, 227, 353, 355, 358, 361–363	<code>\glentrytype</code> ..... 80
<code>\glentrylong</code> ..... 93, 104, 142,	<code>\Glentryuseri</code> ..... 134
144, 147, 153, 220–230, 240, 353, 355–366	<code>\glentryuseri</code> ..... 134
<code>\glentrylongaccess</code> ..... 343	<code>\Glentryuserii</code> ..... 135
<code>\Glentrylongpl</code> ..... 95, 145,	<code>\glentryuserii</code> ..... 134, 135
153, 220, 221, 225–227, 353, 358, 361–363	<code>\Glentryuseriii</code> ..... 136
<code>\glentrylongpl</code> .....	<code>\glentryuseriii</code> ..... 135, 136
..... 94, 104, 144, 146, 153, 220–222,	<code>\Glentryuseriv</code> ..... 136
225–228, 240, 247, 353, 358, 359, 361–365	<code>\glentryuseriv</code> ..... 136, 137
<code>\glentrylongpluralaccess</code> ..... 343	<code>\Glentryuserv</code> ..... 137
<code>\Glentryname</code> ..... 130, 204, 356	<code>\glentryuserv</code> ..... 137
<code>\glentryname</code> ..... 130, 311, 356	<code>\Glentryuservi</code> ..... 138
<code>\glentrynumberlist</code> ..... 153, 172	<code>\glentryuservi</code> ..... 138
<code>\Glentryplural</code> .... 97, 100, 128, 345, 349	<code>\glsfieldfetch</code> ..... 150
<code>\glentryplural</code> .....	<code>\glsfirstaccessdisplay</code> .... 346, 347, 350
..... 96, 97, 100, 128, 344, 345, 348, 349	<code>\glsfirstpluralaccessdisplay</code> .....
<code>\glentrypluralaccess</code> ..... 342	..... 344, 345, 348, 349
<code>\Glentryprefix</code> ..... 260	<code>\glsfirstpluralacessdisplay</code> ..... 349
<code>\glentryprefix</code> ..... 258, 261	<code>\glsgenacfmt</code> .... 220, 221, 227, 357, 358, 363
<code>\Glentryprefixfirst</code> ..... 260	<code>\glsgenentryfmt</code> .....
<code>\glentryprefixfirst</code> ..... 259, 261	.... 220, 221, 226, 227, 232, 234, 236–
<code>\Glentryprefixfirstplural</code> ..... 261	238, 240, 242, 245, 247, 357, 358, 362, 363
<code>\glentryprefixfirstplural</code> .... 259, 262	<code>\glsgetgrouptitle</code> .....
<code>\Glentryprefixplural</code> ..... 260	265, 269, 270, 288–293, 308–311, 314, 315
<code>\glentryprefixplural</code> ..... 259, 262	<code>\gls glossarymark</code> ..... 39

<code>\glsgroupheading</code>	161, 195, 266, 268–270, 272, 273, 275, 282, 284, 286, 288–295, 297, 300, 302, 304, 307–312, 314, 315, 321
<code>\glsgroupskip</code>	..... 160, 161, 195, 267, 268, 272, 273, 275, 278, 279, 283, 284, 286, 294, 296, 297, 301, 303, 304, 308, 309, 311, 314, 320
<code>\glshyperfirstfalse</code>	..... 227, 363
<code>\glshyperfirsttrue</code>	..... 25
<code>\glshyperlink</code>	..... 183
<code>\glshypernavsep</code>	..... 265
<code>\glshypernumber</code>	..... 37, 212
<code>\glsifhyperon</code>	..... 107
<code>\glsIfListOfAcronyms</code>	..... 16
<code>\glsifplural</code>	..... 96, 99, 102, 103, 121–125, 139–145, 225, 234, 238, 240, 242, 344, 348, 351, 352, 354–356, 361
<code>\glsifusetranslator</code>	.... 23–25, 33, 34, 374
<code>\glsindexonlyfirstfalse</code>	..... 25
<code>\glsinlinedescformat</code>	..... 266, 323
<code>\glsinlinedopostchild</code>	..... 266, 323
<code>\glsinlineemptydescformat</code>	..... 266, 323
<code>\glsinlinenameformat</code>	..... 266, 323
<code>\glsinlineparentchildseparator</code>	..... 266, 323
<code>\glsinlinepostchild</code>	..... 266, 323
<code>\glsinlineseparator</code>	..... 266, 323
<code>\glsinlinesubdescformat</code>	..... 266, 323
<code>\glsinlinesubnameformat</code>	..... 266, 323
<code>\glsinlinesubseparator</code>	..... 266, 323
<code>\glsinsert</code>	..... 96– 103, 121–125, 139–145, 225, 226, 234, 237, 238, 240, 242, 243, 344–356, 361, 362
<code>\glskeylisttok</code>	..... 217, 218, 232–239, 241, 243–246, 248, 366–371
<code>\glslabel</code>	.. 79, 96–103, 108–110, 139–145, 220, 221, 225–227, 233, 234, 237, 238, 240, 242, 243, 344–353, 357, 358, 361–363
<code>\glslabeltok</code>	..... 217, 218, 232–241, 243–248, 367–370
<code>\glslink</code>	..... 218, 226, 228, 233, 362, 364
<code>\glslink options</code>	
counter	..... 106, 120, 255
format	..... 106, 120, 210
hyper	..... 106, 108, 109, 120
local	..... 106
<code>\glslinkcheckfirsthyperhook</code>	..... 108
<code>\glslinkpostsetkeys</code>	..... 109
<code>\glslinkvar</code>	..... 107
<code>\glslistdottedwidth</code>	..... 270, 324
<code>\glslistgroupheaderfmt</code>	..... 269, 270
<code>\glslistnavigationitem</code>	..... 269, 270
<code>\glslocalreset</code>	..... 89
<code>\glslocalunset</code>	..... 90, 121–126
<code>\glslongaccessdisplay</code>	..... 353, 355–367
<code>\glslongkey</code>	..... 371
<code>\glslongpluralaccessdisplay</code>	..... 353, 358, 359, 361–365, 367
<code>\glslongpluralkey</code>	..... 371
<code>\glslongtok</code>	..... 217, 218, 220, 221, 226, 227, 232– 241, 243–248, 357, 358, 362, 363, 366–371
<code>\glsLTpenaltycheck</code>	..... 281
<code>\glsncols</code>	..... 288–292
<code>\glsnameaccessdisplay</code>	..... 356, 357
<code>\glsnamefont</code>	..... 204, 205, 336, 337, 356
<code>\glsnavhyperlink</code>	..... 265
<code>\glsnavhyperlinkname</code>	..... 263
<code>\glsnavhypertarget</code>	..... 269, 270, 288–293, 308, 310, 311, 315
<code>\glsnavigation</code>	..... 269, 270, 288–292, 308, 309, 311, 314
<code>\glsnextpages</code>	..... 8, 64, 186
<code>\glsnogroupskipfalse</code>	..... 9
<code>\glsnoidxdisplayloc</code>	..... 173, 174
<code>\glsnoidxdisplaylocclishandler</code>	.... 173
<code>\glsnoidxloclist</code>	..... 172, 195, 196
<code>\glsnoidxloclisthandler</code>	..... 196
<code>\glsnoidxnumberlistloophandler</code>	.... 173
<code>\glsnoidxstripaccents</code>	..... 20
<code>\glsnomakeindexwarning</code>	..... 163
<code>\glsnonextpages</code>	..... 64, 185
<code>\glsnopostdotfalse</code>	..... 9
<code>\glsnoxindywarning</code>	.. 36, 43–45, 47–50, 157
<code>\glsnumberformat</code>	..... 154
<code>\glsnumberlistloop</code>	..... 173
<code>\glsnumbersgroupname</code>	..... 29, 35, 208
<code>\glsnumlistlastsep</code>	..... 154, 173
<code>\glsnumlistparser</code>	..... 154, 170
<code>\glsnumlistsep</code>	..... 154, 173
<code>\glsopenbrace</code>	..... 47, 160, 161, 320, 321
<code>\glsorder</code>	..... 26, 167–169, 191
<code>\glsorg@endtheglossary</code>	..... 5
<code>\glsorg@PrintChanges</code>	..... 5
<code>\glsorg@theglossary</code>	..... 5
<code>\glspagelistwidth</code>	273, 274, 276, 277, 279, 280, 284–287, 295–297, 299, 300, 302–305
<code>\glspatchLToutput</code>	..... 277–280
<code>\glspenaltygroupskip</code>	..... 278, 279



<code>\glsperscentchar</code> .....	70, 160, 162	
<code>\Glspl</code> .....	94, 231	283, 284, 286, 294, 295, 297, 301, 302, 304, 307, 309, 310, 313, 323–330, 332–335
<code>\glspl</code> .....	94, 231	<code>\glssymbolaccessdisplay</code> ....
<code>\glspluralaccessdisplay</code> .....	344, 345, 348, 349	346–348, 357
<code>\glspluralsuffix</code> .....		<code>\glssymbolpluralaccessdisplay</code> .
.....	32, 82, 220–222, 358, 359, 363–365	344, 345
<code>\glspostdescription</code> .....		<code>\glssymbolsgroupname</code> .....
..	35, 267–269, 272, 282, 283, 294, 301, 307–310, 313, 314, 323–326, 328–332, 334	28, 35, 208
<code>\glspostinline</code> .....	265	<code>\glstarget</code> .....
<code>\glspostlinkhook</code> .....		206, 267–273, 275, 282–284, 286, 294–297, 301, 302, 304, 307–310, 312, 314, 323–335
.....	108, 121–126, 139–146, 354–356	<code>\glstextaccessdisplay</code> ..
<code>\glsprestandardsort</code> .....	11	346, 347, 349, 350
<code>\glreset</code> .....	89	<code>\glstextformat</code> .....
<code>\glresetentrycounter</code> .....	199, 201	108, 110
<code>\glresetentrylist</code> .....	160, 193, 200, 320	<code>\glstextup</code> .....
<code>\glresetsubentrycounter</code> .....		32, 365
.....	199, 200, 203, 266, 323	<code>\glstildechar</code> .....
<code>\glssanitizesortfalse</code> .....	22	43, 160, 161
<code>\glssanitizesorttrue</code> .....	22	<code>\glstranslatefalse</code> .....
<code>\glssavenumberlistfalse</code> .....	8	24
<code>\glssavewritesfalse</code> .....	28	<code>\glstranslatetrue</code> .....
<code>\glseeformat</code> .....	159, 171, 173, 320	24, 25
<code>\glseeitem</code> .....	183	<code>\glstreechildpredesc</code> .....
<code>\glseeitemformat</code> .....	183	308, 309
<code>\glseeelastsep</code> .....	183	<code>\glstreegroupheaderfmt</code> .....
<code>\glseeelist</code> .....	182	.....
<code>\glseeesep</code> .....	183	288–293, 308–311, 314, 315
<code>\glssetexpandfield</code> .....	19, 21–23	<code>\glstreeindent</code> ..
<code>\glssetnoexpandfield</code> .....	19, 21, 22	309, 310, 312–314, 329–331
<code>\GlsSetQuote</code> .....	79, 162	<code>\glstreeitem</code> .....
<code>\glsettocitle</code> .....	34, 185	288, 289, 306, 307
<code>\glsshortaccessdisplay</code> .....		<code>\glstreenamebox</code> .....
.....	351–354, 357–360, 362–367	312, 314
<code>\glsshortkey</code> .....	371	<code>\glstreenamefmt</code> .....
<code>\glsshortpluralaccessdisplay</code> .....		306–314
.....	351, 353, 358, 362–365, 367	<code>\glstreenavigationfmt</code> .....
<code>\glsshortpluralkey</code> .....	371	.....
<code>\glsshorttok</code> .....		288–292, 308, 309, 311, 314
.....	217, 218, 232–241, 243–248, 367–371	<code>\glstreepredesc</code> .....
<code>\glssortnumberfmt</code> .....	12, 13	307, 309, 310
<code>\glsspace</code> .....	214	<code>\glstreesubitem</code> .....
<code>\glsstepeentry</code> .....	199, 203	288, 307
<code>\glsstepeentry</code> .....	199, 200, 203	<code>\glstreesubsubitem</code> .....
<code>\glssubentrycounterfalse</code> .....	10	288, 307
<code>\glssubentrycounterlabel</code> ..	199, 200, 203	<code>\glstype</code> .
<code>\glssubentryitem</code> .....		108, 109, 121–125, 139–146, 354–356
. 199, 200, 266, 268–270, 272, 273, 275,		<code>\glsucmarkfalse</code> .....
		10
		<code>\glsucmarktrue</code> .....
		10
		<code>\glunset</code> .....
		87, 89, 91, 92, 121–126
		<code>\glsupacrpluralsuffix</code> .....
		.....
		222, 223, 229, 235, 239, 242, 244
		<code>\GlsUseAcrEntryDispStyle</code> ...
		219, 222–
		225, 227, 229, 230, 359, 360, 363, 365, 366
		<code>\GlsUseAcrStyleDefs</code> .....
		219, 222–
		225, 227, 229, 230, 359, 360, 363, 365, 366
		<code>\glswrallowprimitivemodstrue</code> .....
		178
		<code>\glswrite</code> .....
		157–163, 169, 175, 318–322
		<code>\glswritedefhook</code> .....
		70
		<code>\glswriteentry</code> .....
		177
		<code>\glswritefiles</code> .....
		28, 175
		<code>\glsxindyfalse</code> .....
		26
		<code>\glsxindytrue</code> .....
		27
<b>H</b>		
<code>\H</code> .....	21	
<code>\hangindent</code> .....	292,	
	293, 306, 309, 310, 312, 314, 315, 328–331	
<code>\hbox</code> .....	87, 270, 324	

<code>\hfill</code> .....	270, 324	<code>\ifglssacrdescription</code> .....	246
<code>\hline</code> ...	272–274, 276, 283–285, 287, 294–305	<code>\ifglssacrdua</code> .....	236, 242, 245–247
<code>\hsize</code> .....	271, 282, 293, 300	<code>\ifglssacrfootnote</code> .....	108, 246, 247
<code>\hspace</code> .....	306	<code>\ifglssacrronym</code> .....	14, 15
<code>\hss</code> .....	270, 324	<code>\ifglssacrshortcuts</code> .....	30, 231
<code>\ht</code> .....	281	<code>\ifglssacrsmallcaps</code> .....	235, 237, 239, 241, 244, 245
<code>\hyperdef</code> .....	31	<code>\ifglssacrsmlaller</code> .....	235, 237, 239, 242
<code>\hyperlink</code> .....	106, 119, 211	<code>\ifglssautomake</code> .....	27, 170
<code>hyperref</code> package .....	181, 184, 210, 255	<code>\ifglssdescsuppressed</code> .....	266
<code>\hypertarget</code> .....	119	<code>\ifglssentrycounter</code> ....	198, 199, 201–203
<b>I</b>			
<code>\IeC</code> .....	20	<code>\ifglssentryexists</code> ....	52, 53, 69, 70, 78, 81
<code>\if</code> .....	112, 180, 317	<code>\ifglsshaschildren</code> .....	266, 323
<code>\if@endfor</code> .....	264	<code>\ifglsshasdesc</code> .....	266
<code>\if@gl@debug</code> .....	5, 18, 176	<code>\ifglsshaslong</code> .....	93–95, 147, 220, 221, 225, 227, 240, 357, 358, 361, 363
<code>\if@gl@docloaded</code> .....	4, 14, 176	<code>\ifglsshasparent</code> .....	189, 195, 311
<code>\if@gl@sisacronymlist</code> .....	108	<code>\ifglsshasprefix</code> .....	260
<code>\if@openright</code> .....	40	<code>\ifglsshasprefixfirst</code> .....	260
<code>\ifbool</code> .....	15, 25, 28, 52, 96–98	<code>\ifglsshasprefixfirstplural</code> .....	260
<code>\ifboolexpr</code> .....	33, 57, 207	<code>\ifglsshasprefixplural</code> .....	260
<code>\ifcase</code> .....	6, 7, 24, 64, 198, 307, 328	<code>\ifglsshasymbol</code> .....	234, 238, 242, 307–310, 313, 314
<code>\ifcsdef</code> .....	23, 34, 40, 67, 68, 73–78, 105, 177, 188, 189, 191, 193, 209, 219	<code>\ifglsshyperfirst</code> .....	108
<code>\ifcsempty</code> .....	54, 258	<code>\ifglssindexonlyfirst</code> .....	177
<code>\ifcsequal</code> .....	54	<code>\ifglssnogroupskip</code> .....	268, 272, 273, 275, 278, 279, 283, 284, 286, 294, 296, 297, 301, 303, 304, 308, 309, 311, 314
<code>\ifcsstrequal</code> .....	78	<code>\ifglssnonnumberlist</code> .....	200
<code>\ifcsstring</code> .....	77	<code>\ifglssnopostdot</code> .....	9
<code>\ifcsundef</code> ....	6, 12, 27, 30, 31, 33, 37–40, 51, 52, 59, 60, 63, 80, 82, 83, 90, 91, 106, 110, 111, 119, 167, 175, 184, 187, 189, 197, 198, 203, 207–210, 213, 219, 264, 322	<code>\ifglssnumberline</code> .....	41
<code>\ifdef</code> .....	55, 56, 65, 70, 87, 106, 146, 150, 172, 173, 213, 306	<code>\ifglsssanitizesort</code> .....	19, 22
<code>\ifdefempty</code> .....	17, 40, 51, 54–56, 60, 61, 96, 99, 102, 170, 194, 195, 217, 219, 225, 233, 237, 240, 242, 344, 348, 351, 361	<code>\ifglssavenumberlist</code> .....	66, 170, 183
<code>\ifdefequal</code> ....	54–56, 67, 68, 71, 81, 85, 195	<code>\ifglssavewrites</code> .....	28, 167, 176
<code>\ifdefstrequal</code> .....	77	<code>\ifglssubentrycounter</code> ....	199, 201–203
<code>\ifdefstring</code> .....	33, 57, 167–169, 191, 192, 194, 196	<code>\ifglstoc</code> .....	41
<code>\ifdefvoid</code> .....	20, 84, 195	<code>\ifglstranslate</code> .....	33
<code>\ifdim</code> .....	221, 281, 311	<code>\ifglssucmark</code> .....	39
<code>\iffalse</code> .....	83, 89	<code>\ifglssused</code> .....	92, 96–102, 108, 156, 177, 233, 237, 240, 242, 258–262, 344–351
<code>\IfFileExists</code> .....	9, 23, 24, 187	<code>\ifglsswrallowprimitivemods</code> .....	179
<code>\ifglossaryexists</code> ....	38, 49, 53, 167, 168	<code>\ifglssxindy</code> .....	35, 36, 42–45, 47–50, 60, 85, 86, 113, 157, 163, 167, 180, 181, 187, 316–318
<code>\ifgl@sanitize@description</code> .....	21	<code>\ifignoredglossary</code> .....	80, 84, 177
<code>\ifgl@sanitize@name</code> .....	21	<code>\ifin@</code> .....	29
<code>\ifgl@sanitize@symbol</code> .....	22	<code>\ifinlistcs</code> .....	193, 197
<code>\ifgl@xindy@gl@numbers</code> .....	50	<code>\ifKV@gl@link@hyper</code> .....	109, 110
		<code>\ifKV@gl@link@local</code> .....	121–126
		<code>\ifmeasuring@</code> .....	87



<code>\ifnum</code> .....	11, 91, 92, 194, 280, 281, 309, 310, 312, 313, 329, 330	234–246, 256, 264, 265, 281, 288, 289, 306, 307, 322, 339, 354–356, 367–371, 374
<code>\ifstrempy</code> .....	323	
<code>\ifstrequal</code> .....	207	
<code>\ifthenelse</code> .....	23, 30, 40, 110, 169, 208, 247, 264	
<code>\IfTrackedLanguage</code> .....	163	
<code>\IfTrackedLanguageFileExists</code> .....	34, 374, 375	
<code>\iftrue</code> .....	84, 88	
<code>\ifundef</code> .....	58, 69, 80, 157, 162, 169, 199	
<code>\ifvmode</code> .....	155	
<code>\ifvoid</code> .....	281	
<code>\ifx</code> .....	11, 13, 15, 16, 29, 31, 42, 43, 45, 47, 50, 51, 80–83, 86, 110, 111, 113–118, 147, 158, 161, 163–166, 175, 178, 179, 181, 182, 185, 198, 201, 208–211, 233, 235, 237, 239, 241, 242, 244, 246, 248, 249, 318–320, 322, 323, 328–331, 336, 342	
<code>\immediate</code> .....	69, 70, 92, 167, 175, 187	
<code>\in@</code> .....	29	
<code>\index</code> .....	176	
<code>\indexname</code> .....	29	
<code>\indexspace</code> .....	268, 288–293, 308–311, 314, 315	
<code>\input</code> .....	33, 95	
<code>\inputencodingname</code> .....	27	
<code>\InputIfFileExists</code> .....	70	
<code>\istfilename</code> .....	35, 158, 162, 168, 169, 318, 321	
<code>\item</code> .....	268–271, 288, 289, 307, 308, 323, 324, 328	
<b>J</b>		
<code>\jobname</code> .....	36, 69, 70, 158, 162, 167, 168, 186, 187, 318, 321	
<b>K</b>		
<code>\key@ifundefined</code> .....	72, 73	
<code>\KV@glslink@hyperfalse</code> .....	106, 108, 109, 120	
<code>\KV@glslink@hypertrue</code> .....	106, 120	
<b>L</b>		
<code>\L</code> .....	21	
<code>\l</code> .....	21	
<code>\label</code> .....	7, 198, 199, 202	
<code>\language</code> .....	26	
<code>\leaders</code> .....	270, 324	
<code>\leavevmode</code> .....	78, 109	
<code>\let</code> .....	5, 9, 11–15, 20, 21, 23–25, 28–31, 33, 35, 44, 56, 57, 67–69, 78–81, 83, 84, 87–90, 92, 95, 107–110, 112, 119–126, 139–145, 147, 154, 161– 163, 166–173, 176–179, 182, 183, 185, 186, 195, 197, 201, 205, 217, 230–232,	
		234–246, 256, 264, 265, 281, 288, 289, 306, 307, 322, 339, 354–356, 367–371, 374
<code>\letcs</code> .....	54–56, 70, 72, 73, 76, 82, 146, 147, 167, 172, 173, 188, 190, 191, 195, 204, 207, 312	
link text .....	95	
<code>\listcsadd</code> .....	193	
<code>\listcsgadd</code> .....	197	
<code>\listcsxadd</code> .....	188, 189	
<code>\listead</code> .....	192	
<code>\loadglsentries</code> .....	95	
<code>\long</code> .....	78, 211	
<code>\longnewglossaryentry</code> .....	78	
longtable package .....	271, 277, 282	
<code>\LT@end@open</code> .....	281	
<code>\LT@err</code> .....	281	
<code>\LT@foot</code> .....	281	
<code>\LT@head</code> .....	281	
<code>\LT@lastfoot</code> .....	281	
<code>\LT@output</code> .....	281	
<b>M</b>		
<code>\makeatletter</code> .....	70, 186	
<code>\makeatother</code> .....	70	
<code>\makebox</code> .....	270, 312, 314, 324, 330, 331	
makeglossaries ..	26, 36, 49, 58, 163, 169, 187	
<code>\makeglossaries</code> ..	6, 27, 31, 64, 170, 172, 174, 188	
<code>\makeglossary</code> .....	167, 169	
makeindex .....	376	
makeindex .....	10, 26, 27, 32, 35–37, 41, 58, 59, 62, 86, 111, 114, 156, 159, 162, 163, 166, 175, 180, 206, 317, 318	
<code>delim_n</code> .....	37	
<code>delim_r</code> .....	37	
<code>page_compositor</code> .....	36	
special characters .....	113, 156	
<code>\makenoidxglossaries</code> .....	6, 64, 170, 174	
<code>\MakeTextUppercase</code> .....	4	
<code>\MakeUppercase</code> .....	345, 347, 354, 356	
<code>\markboth</code> .....	39	
<code>\mbox</code> .....	155, 269, 292, 293, 312, 324	
memoir class .....	176	
<code>\memUHead</code> .....	39	
<code>\MessageBreak</code> .....	34, 57, 185, 336, 374, 375	
mfirstuc package .....	1	
<code>\mfirstucMakeUppercase</code> .....	4, 39, 75, 97, 99–103, 127–138, 140, 142, 144, 146, 218, 225–228, 238, 243, 261, 262, 349–353, 361, 362, 364, 365	
<code>\midrule</code> .....	278, 279	

<code>\month</code> .....	158, 162, 318, 321	<code>pluralaccess</code> .....	341, 342
<code>multicol</code> package .....	287	<code>prefix</code> .....	256
<b>N</b>			
<code>\n</code> .....	162, 321	<code>prefixfirst</code> .....	256
<code>\NeedsTeXFormat</code> ...	4, 256, 316, 322, 336, 374	<code>prefixfirstplural</code> .....	257
<code>\new@glossaryentry</code> .....	69, 172	<code>prefixplural</code> .....	257
<code>\new@ifnextchar</code> .....	58, 74, 93, 94, 120–124, 126–145, 214–216, 258–261	<code>see</code> .....	5, 8, 63, 69, 169, 171
<code>\newacronym</code> .....	213, 217, 233, 235, 237, 239, 241, 244, 246, 248	<code>short</code> .....	102, 152, 338
<code>\newacronymhook</code> .....	218, 233, 235, 237, 239, 241, 244, 246, 248, 366	<code>shortaccess</code> .....	341, 343
<code>\newacronymstyle</code> ...	220–225, 227, 229, 230	<code>shortplural</code> .....	152, 338
<code>\newcommand</code> .....	5–20, 22, 23, 25, 26, 28–33, 35–45, 47–55, 57–64, 66–79, 84, 85, 87–90, 92–96, 99, 102, 104, 105, 107–110, 112, 113, 119–157, 163, 166– 168, 171, 174–186, 188–193, 195–197, 200–210, 212–221, 230, 232–254, 257– 261, 263–265, 267, 268, 280, 281, 288, 306, 307, 312, 322, 340–342, 357, 371–373	<code>shortpluralaccess</code> .....	341, 343
<code>\newcount</code> .....	12, 67	<code>sort</code> .....	62, 150, 206
<code>\newcounter</code> .....	198, 199, 201	<code>symbol</code> .....	61, 62, 132, 235, 237, 239, 244, 274, 297, 337–339
<code>\newenvironment</code> .....	203	<code>symbolaccess</code> .....	341, 343
<code>\newglossary</code> .....	14, 15, 28, 29, 59, 169	<code>symbolplural</code> .....	133, 338
<code>\newglossaryentry</code> .....	29, 66, 69, 90, 218, 232, 234, 236, 238, 240, 243, 245, 247, 367–370	<code>symbolpluralaccess</code> .....	341, 343
<code>\newglossaryentry</code> options		<code>text</code> .....	62, 120, 126, 148, 235, 239, 337
<code>access</code> .....	339, 340	<code>textaccess</code> .....	341, 342
<code>counter</code> .....	63	<code>type</code> .....	14, 63, 95, 150
<code>description</code> .....	25, 61, 66, 69, 79, 130, 148, 213, 241, 338	<code>user1</code> .....	133, 150, 339
<code>descriptionaccess</code> .....	341, 343	<code>user2</code> .....	134, 151
<code>descriptionplural</code> .....	131, 338	<code>user3</code> .....	135, 151
<code>descriptionpluralaccess</code> .....	341, 343	<code>user4</code> .....	136, 151
<code>first</code> ....	62, 82, 120, 127, 149, 239, 244, 337	<code>user5</code> .....	137, 151
<code>firstaccess</code> .....	341, 343	<code>user6</code> .....	137, 152, 339
<code>firstplural</code> .....	62, 128, 149, 338	<code>\newglossarystyle</code> .....	265, 268–280, 282–305, 307–312, 314
<code>firstpluralaccess</code> .....	341, 343	<code>\newif</code> .....	4, 16, 23, 26, 178
<code>format</code> .....	158	<code>\newlength</code> .....	119, 271, 282, 293, 300, 310
<code>long</code> .....	102, 152, 338	<code>\newrobustcmd</code> .....	69, 74, 75, 93, 94, 108, 120–145, 147–153, 155, 156, 213–216, 257–261, 311
<code>longaccess</code> .....	342, 343	<code>\newterm</code> .....	29
<code>longplural</code> .....	152, 338	<code>\newtoks</code> .....	113, 167, 217
<code>longpluralaccess</code> .....	342, 343	<code>\newwrite</code> .....	69, 157, 162, 167, 169
<code>name</code> ...	61, 62, 66, 69, 79, 129, 147, 183, 337	<code>ngerman</code> package .....	163
<code>nonumberlist</code> .....	64	<code>\noalign</code> .....	281
<code>parent</code> .....	64, 69	<code>\nobreak</code> .....	269, 281, 324
<code>plural</code> .....	62, 82, 128, 338	<code>\noexpand</code> ....	16, 31, 42, 44, 83, 84, 105, 110–112, 118, 119, 154, 164–168, 170, 178, 180, 183, 187, 190, 191, 204, 205, 213, 218, 232, 234, 236, 238, 240, 241, 243, 245, 247, 248, 316, 336, 337, 367–371
		<code>\nohyperpage</code> .....	210
		<code>\noindent</code> .....	206, 289–292, 309–311
		<code>\noist</code> .....	321, 322
		<code>\nopostdesc</code> .....	29, 35, 78, 186, 323
		<code>\normalbaselineskip</code> .....	280, 281
		<code>\nr</code> .....	6, 7, 24, 64, 198

<code>\ns@ACRfull</code>	215
<code>\ns@Acrfull</code>	215
<code>\ns@acrfull</code>	214
<code>\ns@ACRfullpl</code>	216
<code>\ns@Acrfullpl</code>	216
<code>\ns@acrfullpl</code>	215
<code>\ns@ACRlong</code>	143
<code>\ns@Acrlong</code>	142, 143
<code>\ns@acrlong</code>	142
<code>\ns@ACRlongpl</code>	145
<code>\ns@Acrlongpl</code>	144, 145
<code>\ns@acrlongpl</code>	144
<code>\ns@ACRshort</code>	139, 140
<code>\ns@Acrshort</code>	139
<code>\ns@acrshort</code>	138
<code>\ns@ACRshortpl</code>	141
<code>\ns@Acrshortpl</code>	141
<code>\ns@acrshortpl</code>	140
<code>\ns@newglossary</code>	58
<code>\null</code>	112–119, 164–166, 187
<code>\number</code>	11, 82, 92, 178, 179, 205, 337
<code>\numberline</code>	41
<code>\numexpr</code>	92

## O

<code>\O</code>	21
<code>\o</code>	21
<code>\OE</code>	21
<code>\oe</code>	21
<code>\openout</code>	69, 158, 162, 167, 318, 321
<code>\OR</code>	247
<code>\or</code>	6, 7, 24, 198, 307, 328
<code>\org@glossaryentrynumbers</code>	185, 201
<code>\org@glossarytitle</code>	185
<code>\org@glspostdescription</code>	35
<code>\org@ifKV@glslink@hyper</code>	109, 110
<code>\orgAlph</code>	179
<code>\orgalph</code>	179
<code>\orgarabic</code>	179
<code>\orgnumber</code>	179
<code>\orgRoman</code>	179
<code>\orgromannumeral</code>	179
<code>\orgthe</code>	179
<code>\outputpenalty</code>	280, 281

## P

<code>\p@</code>	268, 287, 306
<code>\p@glshyp@opt</code>	107
package options:	
acronym	14, 15, 31, 184, 213

true	15
counter	17
description	239, 240
dua	237, 239, 240
entrycounter	199, 201
true	10
footnote	121–125, 235, 237, 239, 241
hyperfirst	
false	121–125
indexonlyfirst	383
makeindex	160, 255
nogroupskip	272, 273, 275, 278, 279, 283, 284, 286, 294, 296, 297, 301, 303, 304
nolist	248
nolong	249, 271
nomain	14
nonumberlist	8
nosuper	249
notree	249
nowarn	5
numberline	6
sanitize	21, 61, 147, 148
sanitizesort	18
savewrites	28, 380
false	167
true	168, 175
section	6, 39
sort	
def	10, 11
standard	10
use	10, 11
style	7, 248, 249
subentrycounter	199, 201
toc	6
true	6
translate	24
false	24
translator	23
xindy	26, 27, 160, 255
<code>\PackageError</code>	5, 6, 13, 27, 31, 43, 49, 52, 53, 57, 63, 66, 72–78, 80, 81, 90, 106, 146, 166, 167, 169, 172, 174, 191–193, 198, 200, 208, 209, 219, 220, 236, 237, 242, 245, 322, 344
<code>\PackageInfo</code>	5, 167, 176
<code>\PackageWarning</code>	5, 17
<code>\PackageWarningNoLine</code>	5, 17, 34, 374, 375
<code>\pagegoal</code>	281

`\pagelistname` ..... 34, 274–  
                   276, 278, 279, 285–287, 296–299, 303–305  
`\par` ..... 35, 206, 268, 269, 287,  
                   289–293, 306, 307, 309–315, 324, 329–331  
`\parindent` 288–293, 307–310, 312–315, 329–331  
`\parskip` ..... 288–291, 307, 308, 310  
`\PassOptionsToPackage` ..... 256, 336  
`\penalty` ..... 281  
`\phantomsection` ..... 40  
polyglossia package ..... 23, 33  
`\printglossaries` ..... 170  
`\printglossary` .. 15, 18, 28, 29, 170, 184, 200  
`\printglossary options`  
   `entrycounter` ..... 198  
   `nogroupskip` ..... 198  
   `nonumberlist` ..... 200  
   `nopostdot` ..... 198  
   `numberedsection` ..... 198  
   `style` ..... 198  
   `subentrycounter` ..... 199  
   `title` ..... 197  
   `toctitle` ..... 197  
   `type` ..... 14, 183, 197  
`\printindex` ..... 29  
`\printnoidxglossaries` ..... 172  
`\printnoidxglossary` .....  
                   ..... 171, 172, 174, 184, 191, 192, 200  
`\printnoidxglossary options`  
   `sort` ..... 200  
`\printnumbers` ..... 29  
`\printsymbols` ..... 28  
`\ProcessOptions` ..... 256, 336  
`\ProcessOptionsX` ..... 30  
`\protect` ..... 41, 104, 220–222, 227,  
                   228, 234, 238, 240, 353, 357, 358, 363, 364  
`\protected@edef` ..... 7, 44, 45,  
                   48, 50, 80, 84, 86, 96–98, 104, 111, 154,  
                   176, 179, 198, 204, 205, 208, 209, 218,  
                   242, 247, 257, 263, 317, 318, 336, 337, 342  
`\protected@write` ..... 57, 59,  
                   158, 160, 169, 171, 174, 177, 183, 263, 318  
`\protected@xdef` 11–13, 15, 20, 67, 86, 179, 340  
`\providecommand` 15, 31, 32, 39, 57, 92, 120,  
                   160, 169, 171, 187, 204, 205, 268, 287, 306  
`\ProvidesFile` ..... 33  
`\ProvidesPackage` .....  
                   ..... 4, 256, 263, 265, 267, 271, 277,  
                   282, 287, 293, 300, 306, 316, 322, 336, 374

## R

`\r` ..... 21  
`\raggedright` ..... 280, 282–287, 300–305  
`\raisebox` ..... 119  
`\ref` ..... 202  
`\refstepcounter` ..... 199, 202  
`\relax` 7, 9, 12–15, 24, 30, 44, 57, 62, 64, 68,  
                   81, 83, 87, 91, 92, 107, 108, 110, 112–  
                   118, 147, 160–166, 169, 171–173, 177,  
                   179, 180, 182, 185, 186, 194, 195, 197,  
                   198, 207–209, 249, 264, 268, 280, 287,  
                   292, 293, 306, 307, 309–315, 317, 320–  
                   322, 328–331, 339, 354, 355, 367–369, 371  
`\renewacronymstyle` . 357–361, 363, 365, 366  
`\renewcommand` ..... 4–10, 13–18,  
                   22, 24–28, 30, 33–37, 49, 61, 63–65, 78,  
                   90–92, 154, 155, 157, 163, 164, 169–173,  
                   176, 186, 188, 198–200, 217, 218, 220–  
                   230, 233, 235, 237, 239, 241, 242, 244,  
                   246, 248, 266–276, 278, 279, 281–297,  
                   300–304, 307–317, 322–330, 332–335,  
                   339, 344, 348, 351, 353, 356–360, 362–370  
`\renewenvironment` .. 203, 265, 268, 271–  
                   277, 279, 280, 282–305, 307, 308, 310, 312  
`\RequireGlossariesLang` ..... 34, 374, 375  
`\RequirePackage` .....  
                   ..... 4, 8, 9, 23, 24, 30, 33, 248, 255,  
                   256, 271, 277, 282, 287, 293, 300, 337, 374  
`\restorecounters@` ..... 111  
`\romannumeral` ..... 178, 179, 312, 313, 330

## S

`\s@glshyp@opt` ..... 107  
`\s@newglossary` ..... 58  
`\savecounters@` ..... 110  
`\seename` ..... 182  
`\SetAcronymStyle` ..... 25, 26  
`\setbool` ..... 22  
`\setbox` ..... 281  
`\setcounter` ..... 199, 201  
`\SetCustomDisplayStyle` ..... 248  
`\SetDefaultAcronymDisplayStyle` .... 233  
`\SetDefaultAcronymStyle` ..... 246  
`\SetDescriptionAcronymDisplayStyle` 239  
`\SetDescriptionAcronymStyle` ..... 246  
`\SetDescriptionDUAAcronymDisplayStyle`  
                   ..... 237  
`\SetDescriptionDUAAcronymStyle` .... 246  
`\SetDescriptionFootnoteAcronymDisplayStyle`  
                   ..... 235



<code>\translate</code> .....	34, 35	<code>\warn@noprintglossary</code> ..	169, 171, 172, 186
<code>\translatelet</code> .....	14, 15, 28, 29	<code>\write</code> .....	70, 92,
<code>translator package</code> .	14, 15, 23, 28, 29, 33, 34, 184		158–163, 167, 168, 171, 175, 187, 318–322
<code>\TX@trial</code> .....	87	<code>\writeist</code> .....	166, 167, 322
<code>\typeout</code> .....	18		
<b>U</b>		<b>X</b>	
<code>\u</code> .....	20	<code>\x</code> .....	211
<code>\uccode</code> .....	194	<code>\xatlevel@</code> .....	110
<code>\undef</code> .....	64, 184	<code>\xcapitalisewords</code> .....	150
<code>\unskip</code> .....	78, 270, 324	<code>\xdef</code> .....	75, 80–83, 186, 264
<code>\unvbox</code> .....	281	<code>\xglaccsupp</code> .....	342
<code>\usedictionary</code> .....	33	<code>\xifinlistcs</code> .....	188, 189, 192
<code>\usepackage</code> .....	191, 192	<code>xindy</code> .....	376
<b>V</b>		<code>xindy</code> 10, 26, 27, 35, 36, 41, 45, 47–50, 86, 117,	
<code>\v</code> .....	21		118, 157, 159, 175, 180, 187, 206, 255, 317
<code>\val</code> .....	6, 7, 24, 64, 198	<code>\xmakefirstuc</code> .....	96–98, 104, 146, 148, 257
<code>\vbox</code> .....	281	<code>\xspace</code> .....	213
<code>\vsize</code> .....	281	<code>xspace package</code> .....	4, 212, 213
<code>\vskip</code> .....	268, 280, 281, 287, 306	<b>Y</b>	
<code>\vss</code> .....	281	<code>\year</code> .....	158, 162, 318, 321
<b>W</b>		<b>Z</b>	
<code>\warn@nomakeglossaries</code> .....	169, 171, 172	<code>\z@</code> .....	281